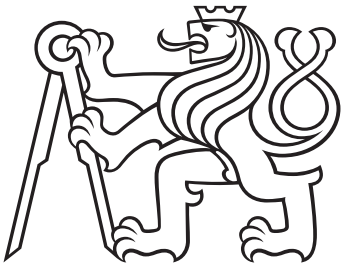
České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce
Obor: Počítačová grafika



Procedurální generování budoucí výstavby měst dle pravidel

Procedural generation of future urban construction according to rules

DIPLOMOVÁ PRÁCE

Vypracoval: Petr Lhota
Vedoucí práce: Ing. David Sedláček, Ph.D.
Rok: 2025

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Lhota** Jméno: **Petr** Osobní číslo: **483655**

Fakulta/ústav: **Fakulta elektrotechnická**

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

Studijní program: **Otevřená informatika**

Specializace: **Počítačová grafika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Procedurální generování budoucí výstavby měst dle pravidel

Název diplomové práce anglicky:

Procedural generation of future urban construction according to rules

Pokyny pro vypracování:

Části města postupně přechází z jedné formy užití v jinou. Po schválení této změny je potřeba komunikovat s obyvateli, jakým způsobem se tato změna projeví (řeší Institut plánování a rozvoje Prahy, zkráceně IPR). Např. převodem oblasti z tzv. brown fieldu na rezidenční čtvrť dojde k významným změnám v oblasti, ale tato změna se musí komunikovat dříve, než se začnou zadávat projekty a soutěže na realizaci změny. Proto je vhodné pro tento budoucí stav vygenerovat možnou budoucí podobu a její varianty, aby byla podpořena občanská povědomost o této změně.
1) Seznamte se s odbornou literaturou a také nástroji vhodnými pro procedurální generování geometrie částí měst, např. CityEngine, Houdiny, Blender.
2) Seznamte se a v práci popište mapové podklady Prahy [1] (strukturu, dostupné datové formáty) vhodné pro vaši práci. A také se vstupně výstupními datovými procesy IPRu.
3) Seznamte se s urbanistickými pravidly pro nové výstavby (Metropolitní plán) [2].
4) Po dohodě s vedoucím vyberte vhodnou podmnožinu pravidel a realizujte je ve vámi vybraném nástroji. Důležitým prvkem je realistické zastavění území a hmota budov včetně dalších přiměřených detailů.
5) Vytvořte náhledy (rendery) porovnávající původní a budoucí výstavbu. Demonstrujte škálu variability minimálně na třech vybraných územích, které vyberete po konzultaci s vedoucím z množiny dodané IPRem. Porovnejte s podobnými, dříve vzniklými, vizualizacemi na IPR.
6) V průběhu práce a při bodu posouzení bodu (5) komunikujte s pověřenou osobou za IPR.

Seznam doporučené literatury:

1] Otevřená data hlavního města Prahy, https://opendata.praha.eu/
2] Metropolitní plán Prahy (textová část + mapy), https://plan.praha.eu/
3] Ondřej Kyzr, Procedural generation of outdoor scenes. BP ČVUT FEL, 2023.
4] Jana Kejvalová, Procedural model generation from real maps. DP ČVUT FEL, 2019.
5] Alena Mikushina, Creation of modular 3D assets for videogames. BP ČVUT FEL, 2020.
6] Jan Kutálek, Procedural generation of videogame environments. BP ČVUT FEL, 2021.
7] Jiří Zemko, Procedural generation of city models. BP ČVUT FEL, 2022.

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. David Sedláček, Ph.D. katedra počítačové grafiky a interakce FEL

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

Datum zadání diplomové práce: **16.02.2024** Termín odevzdání diplomové práce: **07.01.2025**

Platnost zadání diplomové práce: **21.09.2025**

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

III. PŘEVZETÍ ZADÁNÍ

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

.	
Datum převzetí zadání	Podpis studenta

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne .....

.....
Petr Lhota

Poděkování

Děkuji Ing. Davidovi Sedláčkovi, Ph.D. za vedení mé diplomové práce a za podnětné návrhy, které ji obohatily. Dále děkuji Mgr. Jiřímu Čtyrokému, Ph.D. a také Ing. arch. Lukášovi Soukupovi nejen za kvalitní náhled na řešený problém z hlediska urbanistiky, ale i za pomoc při obstarání dat pro moji práci. V neposlední řadě děkuji mé rodině za podporu při studiu.

Petr Lhota

Název práce:
Procedurální generování budoucí výstavby měst dle pravidel

Autor: Petr Lhota

Studijní program: Otevřená informatika
Obor: Počítačová grafika
Druh práce: Diplomová práce

Vedoucí práce: Ing. David Sedláček, Ph.D.
České vysoké učení technické v Praze

Abstrakt: Tato diplomová práce se zaměřuje na procedurální generování budoucí výstavby měst na základě definovaných pravidel. Hlavním cílem je prozkoumat způsoby jak vytvořit sofistikovaný systém, který umožní automatické generování realistických městských prostředí s ohledem na urbanistické normy a požadavky. Práce kombinuje principy urbanismu s pokročilými technikami procedurálního modelování a simulace za účelem dosažení vysoké úrovně autenticity a variability vytvořených scénářů. Implementace je provedena v softwarovém prostředí SideFX Houdini za použití mocných nástrojů pro procedurální generování terénu, budov a infrastruktury. Výsledkem této práce je generátor městských prostředí.

Klíčová slova: grafika, město, procedurální, generace, Houdini

Title:
Procedural generation of future urban construction according to rules

Author: Petr Lhota

Abstract: This thesis focuses on the procedural generation of future cityscapes based on defined rules. The primary objective is to create a sophisticated system capable of automatically generating realistic urban environments while adhering to urban planning standards and requirements. The work integrates principles of urbanism with advanced procedural modeling and simulation techniques to achieve a high level of authenticity and variability in the generated scenarios. The implementation is carried out in the Houdini environment, leveraging its powerful tools for procedural generation of terrain, buildings, and infrastructure. The outcome of this thesis aims to be a generator of city structures.

Key words: graphics, city, procedural, generation, Houdini

Obsah

Seznam použitých zkratek **xi**

Seznam obrázků **xii**

Úvod **1**

1 Procedurální generování v urbanismu **3**
1.1 Procedurální přístup k datům . . . 3
1.1.1 Výhody procedurálního přístupu a jeho využití . . . 3
1.2 Inverzní procedurální modelování . . . 4
1.3 Techniky procedurálního modelování městských struktur . . . 4
1.3.1 L-systémy . . . 5
1.3.2 Gramatiky . . . 5
1.3.3 Tvarové gramatiky . . . 5
1.3.4 Rozdělovací gramatiky . . . 6
1.3.5 Ostatní techniky . . . 6

2 Analýza **9**
2.1 Data . . . 9
2.1.1 Geografický informační systém . . . 9
2.1.2 Použité geografické termíny . . . 9
2.1.3 Souřadnicové systémy . . . 10
2.1.4 Světový geodetický systém 1984 . . . 10
2.1.5 Systém jednotné trigonometrické sítě katastrální . . . 11
2.2 Pravidla pro výstavbu města Prahy . . . 11
2.2.1 Pravidla zástavby rozvojových ploch . . . 12
2.2.2 Stavební bloky . . . 13
2.2.3 Posouzení zastavitelnosti . . . 14
2.2.4 Vymezení komunikací . . . 14
2.3 Existující nástroje procedurálního generování . . . 14
2.4 Houdini . . . 16
2.5 Cíl . . . 17

3 Návrh řešení a realizace **19**
3.1 Návrh algoritmického postupu . . . 19
3.2 Zpracování dat . . . 19
3.3 Terén . . . 21
3.4 Tvorba sítě ulic . . . 21
3.4.1 Tvorba tenzorových polí . . . 22
3.4.2 Tvorba mřížkových vzorů . . . 23
3.4.3 Tvorba radiálních vzorů . . . 23
3.4.4 Vliv hranice na tenzorové pole . . . 24
3.4.5 Vliv terénu na tenzorové pole . . . 25

3.4.6 Vliv existujících ulic . . . 25
3.4.7 Trasování tenzorovým polem . . . 26
3.4.8 Graf ulic . . . 29
3.5 Tvorba geometrie ulic . . . 29
3.5.1 Geometrie křižovatky . . . 30
3.5.2 Geometrie silnic . . . 31
3.6 Tvorba bloků zastavitelných ploch . . . 33
3.6.1 Rozdělení stavebních bloků . . . 33
3.6.2 Příprava parcel pro geometrie budov . . . 35
3.7 Tvorba budov . . . 35
3.7.1 Tvorba střech . . . 35
3.8 Projekce geometrie na terén . . . 36
3.8.1 Projekce budov na terén . . . 37

4 Výsledky a zhodnocení **39**
4.1 Výsledky . . . 39
4.1.1 Vizuální výsledky . . . 42
4.1.2 Dodržení pravidel Metropolitního plánu . . . 44
4.1.3 Statistiky . . . 44
4.1.4 Možná zlepšení . . . 45

Závěr **51**

Bibliografie **53**

Přílohy **55**
A Seznam použitých uzlů . . . 55
B Obrázky sítí geometrie . . . 57

Seznam použitých zkratek

CGA Počítačově generovaná architektura (*Computer Generated Architecture*)

GIS Geografický informační systém (*Geographic Information System*)

HDA Houdini Digital Asset

IPR Institut plánování a rozvoje hlavního města Prahy

RPB Regulovaná plocha budovy

S-JTSK Systém jednotné trigonometrické sítě katastrální

SOP Povrchový operátor (*surface operator*)

WGS84 World Geodetic System 1984

Seznam obrázků

1.1 Nové budovy vybudované inverzním procedurálním generováním . . . 4
1.2 Příklad vygenerovaného půdorysu města pomocí L-systémů [7]. 5
1.3 Tvarový strom gramatiky a výsledná geometrie 6
1.4 Příklad rozdělovací gramatiky [7]. 7
1.5 Výsledek dané rozdělovací gramatiky [7]. 7

2.1 Česká republika ve Křovákově zobrazení (černě) v porovnání s WGS84 (oranžově). 11
2.2 Příklad kompaktní blokové struktury. 12
2.3 Příklad rostlé struktury. 12
2.4 Příklad volné specifické struktury [15]. 12
2.5 CGA editor ArcGIS CityEngine 15
2.6 Vzniklý artefakt při tvorbě silniční soustavy v CityEngine 2023 [18]. . 15
2.7 Vygenerovaná městská struktura v CityEngine 2023 [18]. 15
2.8 Příklad SOP sítě v Houdini 18

3.1 Oblast Velkého Háje z prostředí QGIS. 20
3.2 Maskování terénu 21
3.3 Vyčištěný polygon terénu 21
3.4 Diagram tvorby uliční sítě. 22
3.5 Vizualizace tenzorového pole s mřížkovým prvkem 23
3.6 Vygenerovaný graf cest kombinovaného tenzoru mřížkového typu . . . 23
3.7 Vizualizace kombinovaného tenzorového pole s radiálním prvkem . . . 24
3.8 Vygenerovaný graf cest kombinovaného tenzoru radiálního 24
3.9 Vliv hranice polygonu na generovanou síť. 24
3.10 Vizualizace tenzorového pole závislého na výškovém poli 25
3.11 Vygenerovaný graf cest pomocí tenzoru z obrázku 3.10. 25
3.12 Generovaný graf ulic s překryvnou ulicí 26
3.13 Existující ulice upravující napojování a tvar ulic generovaných 26
3.14 Artefakty LABS Road Generator 30
3.15 Seřazená primitiva geometrie křižovatky 31
3.16 Vodící geometrie křižovatek 31
3.17 Artefakt způsobený blízkým napojením dvou silnic na křižovatku. . . 32
3.18 Opravený artefakt. 32
3.19 Vygenerovaná geometrie křižovatky 32
3.20 Geometrie silnic a křižovatek 33
3.21 Rozdělené bloky pomocí mřížky 34
3.22 Rozdělené bloky pomocí konvexní obálky 34
3.23 Pětipodlažní budova s rovnou střechou 36
3.24 Čtyřpodlažní budova s valbovou střechou 36

3.25 Třípodlažní budova s rovnou střechou 37
3.26 Zarovnání ploch terénu 37

4.1 Výstup algoritmu na vygenerovaném terénu 40
4.2 Původní zástavba lokality Na Dědině. 41
4.3 Návrh zástavba lokality Na Dědině 41
4.4 Vizualizace Dukelské kasárny, Jindřichův Hradec 42
4.5 Studie kasáren, Nové Město nad Metují 43
4.6 Vizualizace Smíchov City - Sever, Kuba & Pilař architekti [23]. 43
4.7 Vizualizace splnění kritérií zástavby bloků 44
4.8 Vizualizace podlažnosti budov 45
4.9 Testování časové náročnosti 45
4.10 Geometrie urbanistického návrhu vytvořená pomocí ChatGPT. 46
4.11 Vygenerovaný detail jako dotaz pro umělou inteligenci 46
4.12 Stylizace obrázku 4.11, realisticky, zdroj *Dzine.ai*. 47
4.13 Jiná stylizace obrázku 4.11, realisticky, zdroj *Dzine.ai*. 47
4.14 Stylizovaný obrázek 4.11, umělecky, zdroj *Dzine.ai*. 47
4.15 Návrh zástavby Dolních Počernic - jih 48
4.16 Návrh zástavby Dolních Počernic - jih s těsnou zástavbou 48
4.17 Návrh zástavby Velkého Háje 49
4.18 Část návrhu zástavby Velkého Háje 49
19 Objekty v síti. 58
20 Hlavní síť uzlů. 59
21 Hlavní ovládací panel. 60
22 Ovládání tvorby tenzorů. 60
23 Ovládání hlavních silnic. 61
24 Ovládání vedlejších silnic. 61
25 Ovladač chování polygonu. 61
26 Ovladač chování bloků. 62
27 Ovladač chování parcel a budov. 62
28 Síť generátoru tenzorových polí a grafu ulic. 62
29 Síť s přepínači na ovládání tenzorového pole. 63
30 Trasovací cyklus. 64
31 Příprava geometrie polygonu s odebraným průmětem geometrie ulic. . 65
32 Subdivize bloků s vyrovnáním k ose. 66
33 Příprava základů budovy. 67
34 Logika geometrie budovy. 68
35 Logika geometrie střech. 69

Úvod

Procedurální generování je disciplína ve výpočetních technikách, která využívá algoritmické tvorby dat na základě parametrizace. Častokrát se využívá prvků vytvořených člověkem spojených s algoritmicky generovanými objekty s případnou náhodností k dosažení realismu a funkčnosti v daném oboru.

V minulosti se procedurální techniky využívaly pro definování jednoduchých geometrií a barev. Od těchto základů se časem techniky posunuly k pokročilému modelování a hlubšímu texturování objektů. Nejedná se pouze o statické objekty. Mezi nejčastěji procedurálně ovládanou geometrii patří například voda, oheň nebo pára. Ovládat vizualizaci takových objektů a jejich chování s sebou nese potřebu matematické zdatnosti programátorů.

Kromě vytváření dat se tato disciplína zabývá i optimalizací geometrií objektů. V současné době existuje spousta metod zabývajících se teselací geometrie či její vizuální reprezentací. Optimalizace se často řeší pomocí decimace samotné geometrie nebo vykreslováním pomocí statických či adaptivních LOD modelů.

S momentálním rapidním vývojem strojového učení, umělé inteligence a právě počítačové grafiky se procedurální techniky stávají velmi zajímavým a živým nástrojem pro tvorbu těch nejrealističtějších, účelných či uměleckých výsledků, a to i v reálném čase. Nejenže jsou moderní nástroje vytvářející procedurální obsah efektivní, jsou také relativně dostupné i na komerčních počítačích. S grafikou vytvářenou procedurálními algoritmy se můžeme setkat v počítačových hrách, ve filmu, reklamě či dokonce při fotografování mobilními telefony s nejnovější technologií. Kromě vytváření grafického obsahu je přínosné pro disciplíny, jako je právě urbanismus.

Urbanisté v rámci své práce musí manuálně vytvářet návrhy, které mohou nebo nemusí korelovat s potenciálním vývojem zástavby kvůli neurčitostem architektonického plánování. Musí vytvořit několik vizualizací jak pro veřejnost, tak i pro technické, průzkumné a statistické účely. Všechny tyto vizualizace ale ukazují, jak vyvíjená oblast bude v nějakém smyslu vypadat. Procedurální techniky se pro urbanisty jeví jako velice vhodné pro usnadnění této práce z hlediska parametrizace daných oblastí.

V následujících kapitolách se práce bude zabývat principy procedurálního generování a již existujícími metodami pro jeho použití v praxi, konkrétně v jejich aplikaci pro tvoření modelů a systémů pro potenciální výstavbu měst. Na základě těchto poznatků byl v prostředí Houdini vyvinut systém pro tvorbu městských struktur, který je založený na vybraných pravidlech výstavby daných Metropolitním plánem města Prahy a jiných předpisech. Měl by být schopný generovat věrohodné reprezentace budoucí zástavby. Projekt byl vytvářen primárně na podněty Institutu plánování a rozvoje hlavního města Prahy (IPR Praha), který zpřístupnil data pro tento projekt.

Kapitola 1

Procedurální generování v urbanismu

Kapitola pojednává o hlavních charakteristikách procedurálního generování vzhledem k urbanismu. Je zde popsáno, v jakých případech je vhodné procedurální generování používat a jak maximalizovat jeho účinnost. Dále představuje již vyvinuté aplikace moderní doby týkající se problematiky tvorby městských struktur.

1.1 Procedurální přístup k datům

Podle profesora I. Parberryho musí procedurální generování mít tři důležité vlastnosti. První z nich je nenáročnost provozu, neboli nezávislost na nejnovějších technologiích. Poté je nutné, aby generování bylo náhodné a strukturované tak, aby vytvářelo obsah, který je různorodý a zajímavý. Nakonec by se s vygenerovanými daty mělo zacházet jednoduše a intuitivně [1].

Jiný pohled na procedurální generování uvádí, že může být lehce abstrahované, což znamená, že uživatel nepotřebuje znát procedury, ale stále může vytvořit algoritmus, který by systém měl zvládnout pochopit a zakomponovat do výsledku. Následně by mělo splňovat možnost kontrolovat parametry, podle kterých se data generují. Nakonec je důležité, aby bylo flexibilní a vytvářelo zajímavý a různorodý obsah bez potřeby podřídit se limitacím reálného světa [2].

To však neznamená, že by tento přístup generování objektů měl být intuitivní i pro nejběžnějšího uživatele. V dnešní době se procedurální modelování používá při tvorbě velice komplexních modelů. Nejčastěji to bývají terény, stromy, mraky, ale také třeba člověkem vytvořené objekty, jako například stavby nebo právě celé komplexní městské struktury.

1.1.1 Výhody procedurálního přístupu a jeho využití

Pro tvorbu těchto objektů potřebujeme poměrně malý objem dat, který popisuje charakteristiky daného vzoru. Model je následně vytvořen procedurou, která využívá parametrizace v kombinaci s pseudonáhodnými čísly. Tímto způsobem můžeme vytvářet objekty, ve kterých se vyskytují podobné modely, ale žádný z nich není nutně stejný. Procedurální metody jistým způsobem amplifikují původní data [3].

Hlavní motivací pro procedurální generování je produkce velkého množství různorodých dat, což ale vyžaduje znovupoužitelnost algoritmů. To od vytvořených

systémů vynucuje jejich promyšlenost a perfekcionismus. Nemluvíme nutně pouze o recyklovatelnosti v rámci jednoho projektu, ale můžeme tak mluvit i o reprodukci algoritmů. Například simulátor ohně může v jiném projektu simulovat roznášení nákazy.

Další motivací je unikátnost založená na individuálních zkušenostech uživatelů. Díky procedurálnímu generování vznikají nové interakce díky jeho nepředvídatelnosti a živosti systémů. To například u některých videoher prohlubuje faktor zábavy.

Na druhou stranu programy založené na procedurálním generování nutně nemusí vždy pracovat tak, jak bylo původně zamýšleno. Je prakticky nemožné pro testery procházet každou iteraci procedurálního obsahu. Chyba se může vyskytnout u 0.01% případů, se kterými se setká jenom hrstka uživatelů, ale chybovost může být i vyšší [4].

1.2 Inverzní procedurální modelování

Procedurální modelování prokazuje přesvědčivé výsledky co se týče výstavby architektonických staveb. Nicméně vytváření detailních a realistických budov vyžaduje velké množství času a rozsáhlé kódování. Naproti dopřednému procedurálnímu přístupu inverzní procedurální modelování konvertuje již existující modely do lehce syntetizovatelného procedurálního tvaru a dovoluje rychlé přetvoření na velmi podobné geometrie. Většina takových algoritmů se soustředila primárně na plochy, mračna bodů, stavby nebo městské struktury. Tyto metody ovšem až do nedávna vyžadovaly dobře segmentované a pojmenované modely se strukturálními omezujícími podmínkami [5].

Obrázek 1.1: Nové budovy vybudované inverzním procedurálním generováním, originální vstupní model nalevo [5].

1.3 Techniky procedurálního modelování městských struktur

Existuje již několik sofistikovaných technik, které dovolují generovat nejrůznější data. Nejobecnějšími tvary, které s procedurálním modelováním úzce souvisí, jsou fraktály. Ty jsou soběpodobné, což znamená, že jejich část je stejná nebo přibližně podobná, jako některé z jejích podčástí [6]. Ačkoliv je možné pomocí fraktálů v určitých momentech (například u texturování) dosáhnout nekonečného detailu, je to oblast, kterou urbanismus při návrhu městských struktur nebo geometrií budov nevyžaduje, a proto se soustředí na několik jiných funkčních struktur a algoritmů, které tuto disciplínu podporují.

1.3.1 L-systémy

Lindenmayerovy systémy (dále L-systémy) se tradičně používají pro přírodní úkazy, jako jsou rostliny, stromy, ale jsou vhodné i pro generování městských struktur pro jejich stručnost a výpočetní efektivitu. Jejich možným výstupem jsou rozvětvené struktury, které by v našem případě mohly připomínat sítě komunikací. Například pro CityEngine jsou L-systémy jednou z klíčových technik, a to nejen pro tvorbu komunikací, ale i budov. Využívá totiž síť ulic pro tvorbu půdorysů jednotlivých bloků a následně budov. Tradičně se jedná o geometrie měst manhattanovského typu, souběžných silnic a kolmých křižovatek. Ostatní městské struktury s nepravidelným tvarem je složitější pro tyto systémy vytvářet [7]. Nicméně v poslední době se i v těchto zemích snaží ubírat k maximální efektivitě zástavby.

Obrázek 1.2: Příklad vygenerovaného půdorysu města pomocí L-systémů [7].

1.3.2 Gramatiky

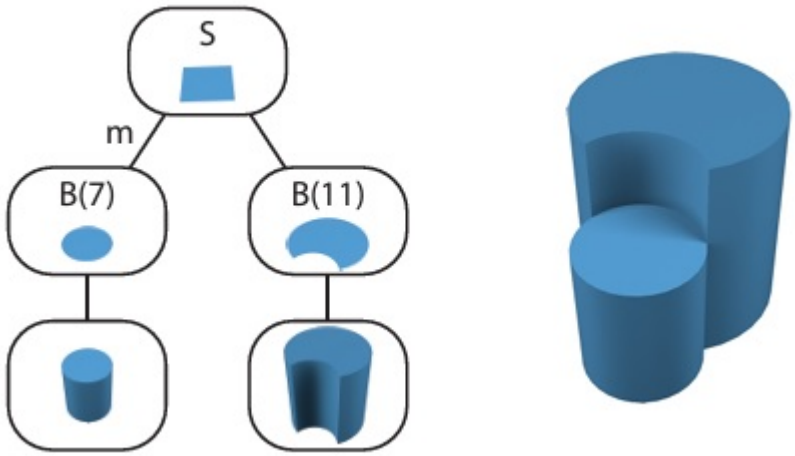
Formálně, gramatiky (anglicky grammars) jsou čtveřice (N, Σ, P, S), kde N je konečná množina neterminačních symbolů, Σ je konečná množina terminačních symbolů, která je různá od N, P je konečná množina produkčních pravidel, která mají formu $\alpha \rightarrow \beta$, kde α a β jsou prvky z $N \cup \Sigma$ a S je startovní symbol, $S \in N$. Neformálně jsou nástrojem, který popisuje, jak tvořit prvky z abecedy formálního jazyka, které jsou validní k syntaxi toho stejného jazyka. Pro účel této práce se zaměříme na dva typy slovníků.

1.3.3 Tvarové gramatiky

Tento typ gramatiky se zaměřuje na fundamentální primitiva tvarů spíše než symbolů. Sestává z tvarových pravidel a generačního enginu, který vybírá a zpracovává pravidla na geometrii. Pravidla definují, jak se existující části nebo tvary mají transformovat. Minimálně je tvoří tři pravidla: startovní pravidlo, alespoň jedno transformační pravidlo a terminační pravidlo. Engine potom může pravidla aplikovat buď sériově, paralelně nebo vybrat určitý počet shod a na ty pravidlo aplikovat [8]. Nejčastější je však iterativní způsob (sériový). Příkladem může být změna svislé úsečky na úsečku s vychýleným kloubem. Příklad tvarové gramatiky:

$$S \rightarrow i(\text{"}circle\text{"}), m = B(7), t(3,0,0), s(10,0,10), minus(m), B(11) \tag{1.1}$$

$$B(h) \rightarrow extrude(h)$$



Obrázek 1.3: Nalevo tvarový strom gramatiky definovaný podle výrazu 1.1, napravo výsledná geometrie [9].

1.3.4 Rozdělovací gramatiky

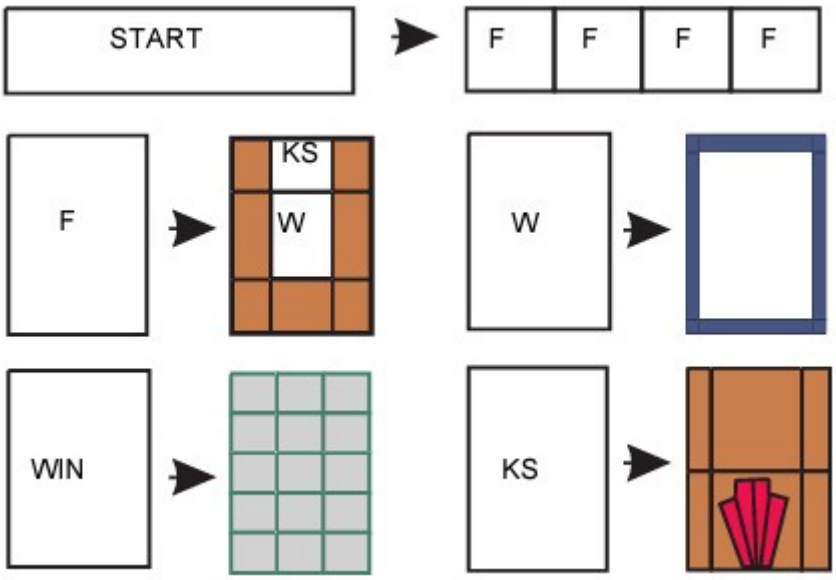
Rozdělovací gramatiky (split grammars) se taktéž zaměřují na koncept tvaru. Jsou jistým poddruhem tvarových gramatik. Zaměřují se na rozdělující operace, jejichž pravidlem je dekompozice tvaru na množinu menších objektů. Tyto gramatiky jsou vcelku jednoduché, ale poměrně omezené pro složitější objekty [10].

1.3.5 Ostatní techniky

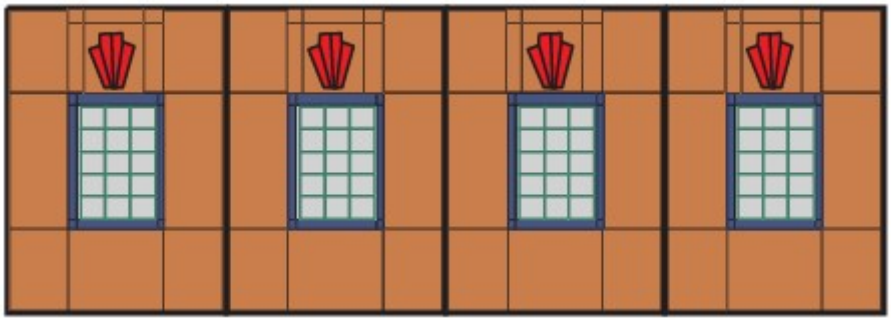
Mezi další používané procedurální techniky pro urbanismus patří například grafové gramatiky. Speciálně tato metoda, podobná inverzním procedurálním technikám, na základě vstupu automaticky tvoří množiny pravidel pro grafové gramatiky. Tato metoda byla prezentována na SIGGRAPH 2023. Není však pro svou použitelnost moc populární.

Co se týče ostatních technik, existuje simulace na bázi agentů. Tato metoda pracuje s parametrizovatelnou odchylkou, při hodnotě 0 vznikají rovné bloky, při hodnotě 1 organické městské struktury. Cesty generují pomocí dvou typů agentů - *rozšiřujícími* a *spojujícími*. Podobně vznikají i budovy, pro které je proces tvorby rozdělen pomocí *vývojářských* agentů, kteří rozdělí město na jednotlivé typy, například rezidenční, komerční, industriální. Tvorba měst tímto způsobem však zabírá značně delší dobu.

Dále se používají techniky na základě projektování již existujících městských vzorů na terén.



Obrázek 1.4: Příklad rozdělovací gramatiky [7].



Obrázek 1.5: Výsledek dané rozdělovací gramatiky [7].

Kapitola 2

Analýza

V této kapitole je čtenář seznámen s hlavní analýzou a se souvisejícími tématy týkajícími se této práce. Je obeznámen se základy geografických informačních systémů (GIS), využitým softwarem Houdini a s daty zařazenými do kontextu.

2.1 Data

Data pro tuto práci byla poskytnuta Institutem plánování a rozvoje hlavního města Prahy (dále IPR Praha). Byla poskytnuta ve formátu ESRI FileGDB, což je kolekce geoprostorových dat doprovázených tabulkovými atributy. Jedná se o data z území Prahy a jejího okolí, konkrétně z lokalit Na Radosti, Velký háj, Na Homoli, Dědina, Nová Ruzyně, Zálesí a Dolní Počernice jih. Příprava těchto dat nebo jim podobných vyžaduje znalost z následujících podkapitol. Z poskytnutých dat jsou důležité vrstvy potenciální zástavby, křivky již existujících ulic a vrstvy definující podlažnost oblastí.

2.1.1 Geografický informační systém

Pro práci byl zvolen software QGIS, protože je ke zpracování dat nutné zpracování v geografickém informačním systému (GIS[1]). Je to open source software určený primárně pro prohlížení, tvorbu a editaci rastrových a vektorových geodat a jejich exportování. Existuje pro něj řada zásuvných modulů (*pluginů*) a právě proto se jeví jako vhodný kandidát pro tuto práci. FileGDB formát dokáže konvertovat do ESRI Shapefile formátu, který je dobrým přestupním můstkem mezi geografickými formáty a formáty užívanými 3D vývojáři.

2.1.2 Použité geografické termíny

Pro základní orientaci v textu budou následně definovány následující pojmy týkající se zpracování dat v rámci GIS a souřadnicových systémů.

- **EN ISO 19111** je norma znovu účinná od 1. října 2020, která se zaměřuje na reprezentaci geografických informací. Definuje schéma prostorových referencí pomocí souřadnic, včetně popisu datových prvků, vztahů a souvisejících metadat.

[1] Geografický informační systém (*Geographic Information System*)

- **Datum** označuje parametr nebo soubor parametrů, který určuje polohu počátku, měřítko a orientaci souřadnicového systému.
- **Geodetické datum** je definováno jako datum, které popisuje vztah souřadnicového systému k Zemi.
- **Souřadnicový systém** je množina matematických pravidel pro přiřazení souřadnic k bodům. Ty obsahují obvykle pravidla o referenční ploše, orientaci sítě na ní, měřítku, referečním bodu a kartografickém zobrazení. Souřadnicový systém pak umožňuje jednoznačně určit polohu libovolného bodu na ploše nebo v prostoru.
- **Souřadnicový referenční systém** je souřadnicový systém vztahující se k reálnému světu datumem. Zahrnuje systémy založené na geodetických, kartézských či kartografických zobrazeních.
- **Kartografické zobrazení (projekce)** označuje transformaci souřadnic ze zakřiveného geodetického souřadnicového systému do roviny na základě jednoznačného vztahu, který používá stejné datum.
- **Složený souřadnicový referenční systém** je systém, který využívá dva nezávislé referenční systémy, jeden pro horizontální složku a druhý pro vertikální složku.
- **Geodetický souřadnicový systém** je systém, ve kterém je poloha určena pomocí geodetické šířky, geodetické délky a (v trojrozměrném případě) elipsoidické výšky.
- **EPSG kód** je unikátní nezáporné celé číslo (s výjimkou nuly), které odkazuje na databázi obsahující elipsoidy, geodetická data, geografické a kartografické souřadnicové systémy a měrné jednotky. Každé kartografické zobrazení má jedinečný kód [11] [12].

2.1.3 Souřadnicové systémy

Pro zpracování dat bylo třeba prozkoumat souřadnicové systémy vzhledem k souboru dat poskytnutých IPR. Protože se všechny tyto oblasti nacházejí na území České republiky, je nutné prozkoumat následující souřadnicové systémy, a to S-JTSK[2], ve kterém byla data poskytnuta, a WGS84[3], který je světově používaným standardem pro otevřené datové soubory.

2.1.4 Světový geodetický systém 1984

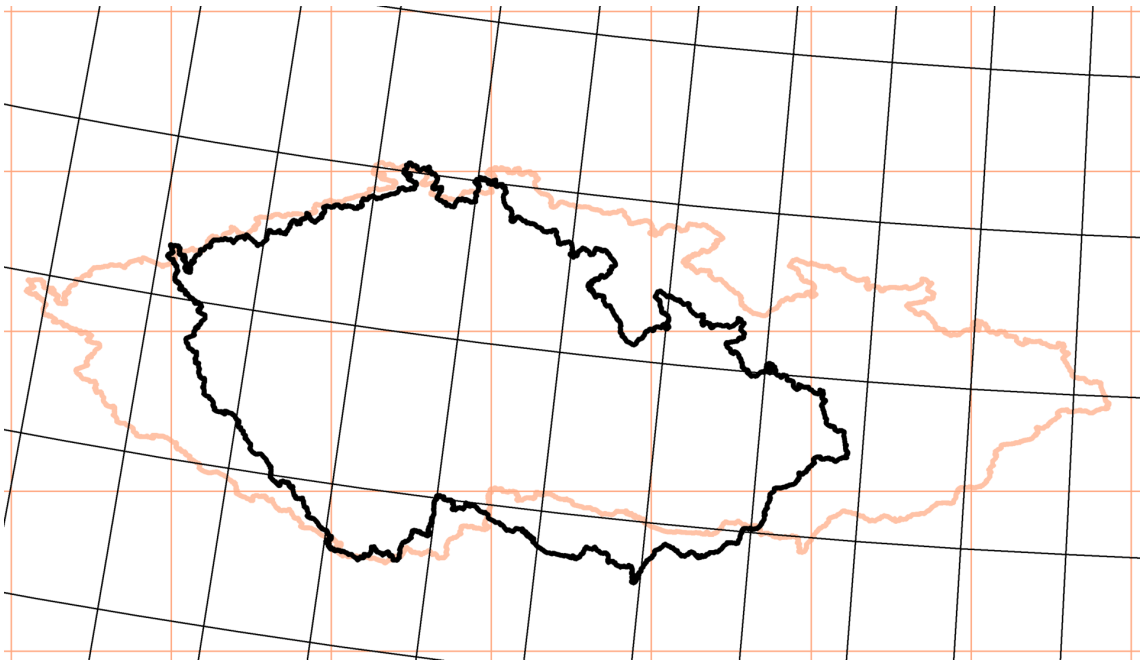
Tento souřadnicový systém (s kódem EPSG:4326 WGS84) je světově uznávaný geodetický standard vydaný USA roku 1984. Šířka nabývá 0°–90° na sever od rovníku a 0°–90° na jih od rovníku. Délka pak nabývá hodnot 0°–180° na západ od nultého poledníku a 0°–180° na východ od nultého poledníku. Nultým poledníkem ve WGS84 je mezinárodní nultý poledník (IERS), ležící 5,31 úhlových vteřin východně od Greenwichského nultého poledníku [13].

[2]Systém jednotné trigonometrické sítě katastrální
[3]World Geodetic System 1984

2.1.5 Systém jednotné trigonometrické sítě katastrální

S-JTSK je pravoúhlá souřadnicová síť používaná v geodézii na území České republiky a Slovenska. V České republice používá tento souřadnicový systém Zeměměřičský úřad pro většinu svých dat. Vychází z Křovákova zobrazení. Území Česka a Slovenska se tak nachází v prvním kvadrantu a má tedy obě souřadnice kladné, což usnadňuje souřadnicové výpočty. Kladná část osy X tohoto souřadného systému je směřována k severu, kladná část osy Y k východu. Pro každý libovolný bod na území České i Slovenské republiky navíc platí, že hodnota souřadnice Y je vždy menší než souřadnice X. Délkové zkreslení se pohybuje v hodnotách od +14 cm/km až do -10 cm/km [13].

Oproti WGS84 je S-JTSK KROVAK mírně natočený a zkosený (obr. 2.1, pokud je použita stejná projekce. Při konverzi vznikají poměrně velké nepřesnosti, navíc dostupná data ve WGS84 nemají na našem území tak detailní rozlišení. Data tedy nelze jednoduše kombinovat.



Obrázek 2.1: Česká republika ve Křovákově zobrazení (černě) v porovnání s WGS84 (oranžově).

2.2 Pravidla pro výstavbu města Prahy

Dalším souvisejícím tématem pro tuto práci je Metropolitní plán Prahy. Metropolitní plán jako Územní plán hlavního města Prahy tvoří základ uceleného, logického a vzájemně provázaného systému územně plánovací dokumentace města [14]. Určuje pravidla pro zastavitelnost a jiné parametry.

2.2.1 Pravidla zástavby rozvojových ploch

Po diskuzi s urbanisty na IPR[4] bylo rozhodnuto, že pro tuto práci jsou primárně důležité struktury rostlé, blokové, hybridní, heterogenní, vesnické, modernistické a lineární. Ty jsou společně s jinými strukturami definovány podle následujícího výčtu.

- Zastavitelné stavební lokality se dělí na:
  - **Rostlé struktury** vzniklé postupným historickým vývojem bez jednotného plánování, často s nepravidelným uspořádáním ulic a parcel. Příkladem rostlé struktury je Staré Město.
  - **Blokové struktury** charakterizované pravidelným uspořádáním ulic a uzavřených městských bloků, typické pro městskou zástavbu 19. a počátku 20. století. Příkladem rostlé struktury jsou Vinohrady.
  - **Hybridní struktury** kombinující prvky různých urbanistických struktur, často vzniklé postupnou transformací území nebo kombinací různých stavebních stylů. Příkladem rostlé struktury je Karlín.
  - **Heterogenní struktury** vyznačující se různorodostí ve velikosti a tvaru staveb, často bez jednotného urbanistického konceptu. Příkladem rostlé struktury je Smíchov.
  - **Vesnické struktury** typické pro okrajové části města, kde původní vesnická zástavba byla začleněna do městského organismu.
  - **Struktury zahradního města**
  - **Modernistické struktury** vzniklé v období modernismu, spíše s volným uspořádáním budov v zeleni, často s důrazem na funkčnost a oddělení jednotlivých funkcí.
  - **Struktury areálů produkce**
  - **Struktury areálů vybavenosti**
  - **Lineární struktury** tvořící zástavbu podél hlavních komunikačních tahů, často s komerčními nebo administrativními funkcemi [14].

Obrázek 2.2: Příklad kompaktní blokové struktury.

Obrázek 2.3: Příklad rostlé struktury.

Obrázek 2.4: Příklad volné specifické struktury [15].

[4]Institut plánování a rozvoje hlavního města Prahy

- Tři typy struktur pro zastavitelné nestavební lokality:
  - **Parkový areál**
  - **Parkové prostranství**
  - **Parkový les**
- Sedm typů struktur pro nezastavitelné lokality:
  - **Lesní krajina v rovině**
  - **Leso-zemědělská krajina**
  - **Zemědělsko-rybniční krajina**
  - **Zemědělská krajina v rovině**
  - **Krajina úzkých zaříznutých údolí v plošině**
  - **Krajina výrazných údolí**
  - **Krajina výrazných vrchů**

2.2.2 Stavební bloky

Stavební blok je vymezen v grafické části Z02[5] bílou plochou stejně jako zastavěné plochy v samotách a budovy v městských parcích [14].

Další důležitou informací je koeficient zastavění stavebního bloku (ZB), který stanovuje maximální podíl součtu RPB[6] všech budov a plochy stavebního bloku. Bloky se pro potřeby stanovení koeficientu rozlišují na:

- malé bloky o ploše PM do 2 000 m^2 včetně,
- střední bloky o ploše PS nad 2 000 m^2 do 12 000 m^2 včetně,
- velké bloky o ploše PV nad 12 000 m^2.

	ZB_M	ZB_{S1}	ZB_V	ZB_N
Rostlá struktura	95%	85%	60%	65%
Bloková struktura	85%	75%	50%	65%
Hybridní struktura	95%	85%	50%	65%
Heterogenní struktura	65%	55%	35%	40%
Vesnická struktura	30%	30%	20%	25%
Struktura zahradního města	35%	35%	20%	25%

Pro malé bloky se použije hodnota ZB_M, pro velké bloky se použije hodnota ZB_V, pro střední bloky se hodnota ZB_S odvodí výpočtem z hodnot ZB_{S1} a ZB_V dle následujícího vzorce, kde P_S značí plochu středního bloku.

$$ZB_S = ZB_V + (ZB_{S1} - ZB_V) \times \frac{(12000 - P_S)}{10000}$$

Pokud není stavební blok vymezen, použije se hodnota koeficientu zastavění stavebního bloku ZB_V. Ve zvlášť odůvodněných případech, kdy není možné vymezit stavební blok a míra využití území odpovídá cílovému charakteru, se použije ZB_N (pro neznámý blok).

[5] Vrstva Metropolitního Plánu města Prahy

[6] Regulovanou plochou budovy (RPB) je plocha, kterou vyjadřuje pravoúhlý průmět obvodových konstrukcí nadzemních podlaží budovy do vodorovné roviny vyjma prvků před stavební čárou

2.2.3 Posouzení zastavitelnosti

Zastavitelnost stavebního bloku se neposuzuje, jde-li o stavbu do 350 m^2 RPB v přímé návaznosti na uliční prostranství [14]. Zástavba je taktéž omezena výškovou regulací. Regulovaným počtem nadzemních podlaží (dále také „RNP“) je počet plnohodnotných nadzemních podlaží budov po jejich hlavní římsu[7].

2.2.4 Vymezení komunikací

Uliční prostranství vytvářejí základní síť obsluhy a prostupnosti zastavitelného území. Podle jejich významu se rozlišuje metropolitní, čtvrťová, lokalitní a místní úroveň uličních prostranství.

- Ulice metropolitní a ulice čtvrťové úrovně se souhrnně považují za městské třídy a jsou uvedeny v KLZ/200, novou zástavbu podél městských tříd je nutné umisťovat tak, aby parter přímo navazoval na uliční prostranství a byl využitelný pro veřejnou vybavenost,
- ulice lokalitní úrovně se považují za významné ulice,
- ulice místní úrovně se považují za lokální a přístupové ulice [14].

Při zakládání nových ulic musí být šířka uličního prostranství alespoň

- 24 m u městských tříd,
- 18 m u významných ulic,
- 12 m u lokálních ulic,
- 8 m u obslužných ulic,

nestanoví-li územní nebo regulační plán jinak [16].

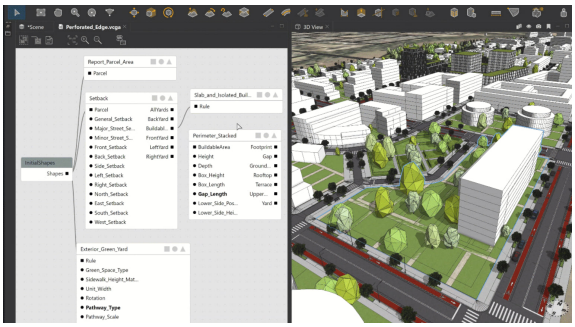
2.3 Existující nástroje procedurálního generování

K dnešnímu dni existuje poměrně široká škála programů zabývajících se touto tématikou. Pouze pár z nich má ale dostatečně silné nástroje a uživatelské rozhraní, aby byly používány komerčně.

Konkrétně pro urbanismus a tvorbu měst je asi nejznámějším programem ArcGIS CityEngine. Jeho síla spočívá v integraci reálných nebo syntetických GIS dat přímo do systému, v jejich prostorové analýze a procesování. CGA (počítačově generovaná architektura) je skriptovací jazyk unikátní pro CityEngine. Pomocí něho je uživatel schopný definovat, jak má geometrie budov, silnic a jiných struktur vypadat, čehož je docíleno na základě pravidel, parametrů a atributů dat.

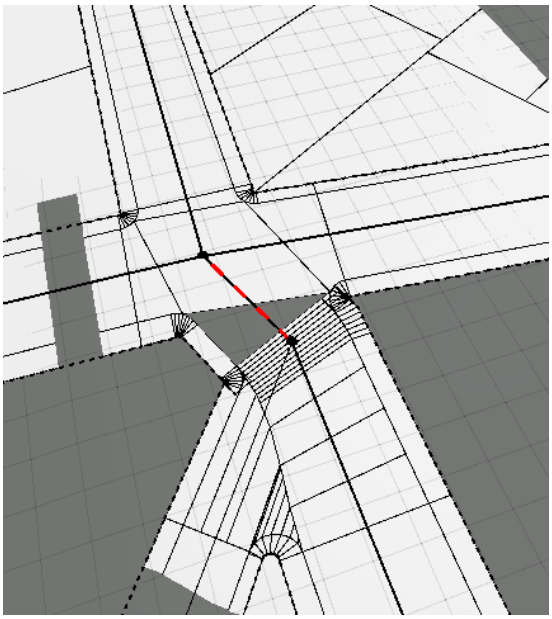
[7] Nad úrovní hlavní římsy je přípustná výstavba šikmých střech, ustupujících podlaží nebo jiných prostorových řešení střechy

Používá zejména algoritmy s L-systémy ve spojení s tvarovými gramatikami, které slouží pro rekurzivní tvorbu staveb. Je schopný vytvářet projekce geometrií na výšková pole a případně tento prostor dělit za pomocí algoritmů spojených s Voronoiovým diagramem. Stejně tak řeší dělení stavebních bloků pomocí subdivizních algoritmů. Řeší také detekci kolizí mezi vygenerovanou geometrií. Za pomocí kombinace zmíněných nástrojů generuje městské struktury a sítě sloužící nejen pro urbanisty (obr. 2.7), ale i pro tvorbu 3D obsahu, například pro průmysl automotive.

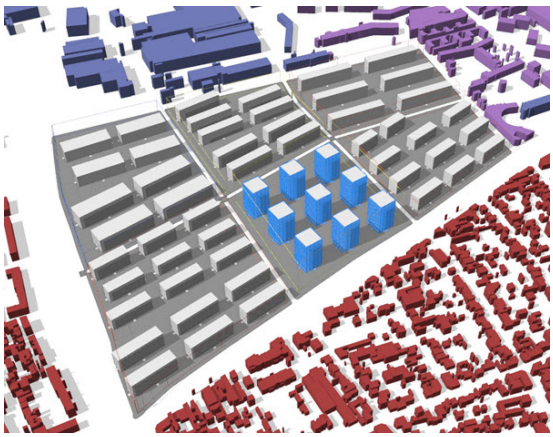


Obrázek 2.5: Nový vizuální CGA editor softwaru ArcGIS CityEngine dovoluje uživatelům vytvářet a modifikovat procedurální pravidla pro generování komplexních 3D urbanistických prostředí [17].

Přes všechny svoje kladné stránky CityEngine stále vyžaduje ruční manipulaci s konfliktními oblastmi vygenerovaných dat. Často se jedná o artefakty generovaných geometrií budov nebo geometrií silnic (obr. 2.6). Pro tyto případy má zabudované nástroje pro opravy chyb.



Obrázek 2.6: Vzniklý artefakt při tvorbě silniční soustavy v CityEngine 2023 [18].



Obrázek 2.7: Vygenerovaná městská struktura v CityEngine 2023 [18].

Všechny tyto nástroje jsou hojně používané a většinou dostupné i v obecnějších generativních softwarech. V rámci obecného generování jsou určitě nejznámější právě enginy Houdini, Maya, Cinema4D a Blender, ve kterých je práce s konkrétní geometrií možná složitější, ale zaručují obecnost a úplnou kontrolu. Existují i programy zabývající se jinou problematikou, jako je třeba tvorba terénů (Terragen, Bryce, World Machine aj.), které ve spojení právě s těmito softwary mohou vynášet lepší výsledky, než zmiňovaný CityEngine. Na rozdíl od CityEngine je třeba, aby vývojář v rámci enginu vytvořil prostředí dostatečně srozumitelné pro urbanistu neznalého algoritmů počítačové grafiky.

2.4 Houdini

Houdini je 3D animační software vyvíjený firmou SideFX. Je to software s nástroji pro produkci nejrůznějších efektů, geometrií a částicových systémů. Používá se také pro animaci filmů a to v celém měřítku. Tento engine se pro tuto práci jeví jako velice vhodný. Je velice silným nástrojem v oblasti procedurálního modelování a texturování. Navíc podporuje skriptování v Pythonu a ve vlastním skriptovacím jazyku VEX, který je velice podobný jazyku C++, ale ne vždy podporuje tvorbu vlastních struktur nebo tříd. Jazyky se v projektech dají kombinovat.

Mimo skriptovací jazyky je nejsilnější stránkou Houdini jeho semivizuální programování. Vývojář pracuje s uzly, které mají různou funkčnost, a tak vzniká nad objekty uzlová síť, která je orientovaným acyklickým grafem. Uzly v této síti označujeme jako *operátory*. Zároveň však uzlová síť vzniká na pozadí enginu samostatně, a to například při animování, rigování postav a podobných úkonech. V Houdini existuje následující výčet typů operátorů.

- **CHOP = Channel nodes** (kanálové uzly) vytvářejí, filtrují a manipulují s daty kanálů.
- **Compositing nodes** (kompoziční uzly) vytvářejí, filtrují a manipulují s obrazovými daty.
- **COP = Copernicus nodes** (COP uzly) umožňují manipulaci s obrazem v reálném čase v rámci 3D prostoru.
- **DOP = Dynamics nodes** (dynamické uzly) nastavují podmínky a pravidla pro dynamické simulace.
- **LOP** (LOP uzly) generují USD (formát Universal Scene Description) popisující postavy, rekvizity, osvětlení a rendering.
- **Object nodes** (uzly objektů) reprezentují objekty ve scéně, jako jsou části postav, geometrické objekty, světla, kamery apod.
- **Render nodes** (renderovací uzly) buď vykreslují scénu nebo nastavují sítě závislostí pro rendering.
- **SOP = Geometry nodes** (surface operator, geometrické uzly) se nacházejí uvnitř geo objektů a generují geometrii. Zároveň jsou nadmnožinou většiny typů operátorů pracující s geometrií.
- **TOP** definují pracovní postup, kde jsou data vkládána do sítě, měněna na pracovní položky a manipulována různými uzly. Mnoho uzlů představuje externí procesy, které lze spustit na místním stroji nebo na serverové farmě.
- **VEX networks** jsou sítě fungující jako kontejnery pro VOP sítě.
- **VOP** jsou uzly umožňující definovat program (například shader) propojením uzlů. Houdini poté kompiluje síť uzlů do spustitelného VEX kódu.
- **APEX** jsou ulzy poskytující operace pro tvorbu funkcionalit APEX grafů, které se používají v KineFX pro tvorbu kosterních rigů a další manipulaci s geometrií [19].

Geometrie je v Houdini reprezentována v pravorukém kartézském souřadnicovém systému. Základní jednotky jsou generické, takže se dají interpretovat jako libovolná měrná jednotka, v našem případě metry. Téměř všechny úkony s geometrií v rámci této práce jsou vykonávány pomocí SOP[8] sítě (obr. 2.8). Na základě typu SOP je výstup jednoho operátoru přijímán dalším operátorem, který výstupní geometrii přebere jako vlastní vstup, a tak vytváří SOP síť, která si postupně předává data pomocí ukládání dat do mezipaměti. Nad geometrií díky tomu může uživatel provádět nejrůznější typy operací, výběrů, seskupování i vizualizací. Do sítě je možné přidávat uzly s různými atributy a parametry. Mezi speciální typ SOP uzlů patří taktéž uzly s pojmenováním *Wrangle*, což je operátor a ovladač na nízké úrovni, který dovoluje expertům v Houdini a skriptovacím jazyku VEX upravovat geometrie a atributy psaním kódu (koresponduje s VOP SOP, ale namísto VOP sítě používá krátký VEX úryvek kódu). Právě pro tuto práci byly Wrangle uzly esenciální pro její tvorbu, protože algoritmy popsané v následující kapitole vyžadují kontrolu nad geometrií zejména v její nízké úrovni.

Jedním z nejprimárnějších nedostatků Houdini ovšem je, že na základě tvorby uzlových sítí, které mají vždy podobu orientovaného acyklického grafu, engine nepodporuje rekurzivní zanoření pro uzlové sítě ani VEX skripty, a tak je někdy získávání atributů, parametrů a předávání výstupů překážkou. Je tedy nutné převést rekurzivní algoritmy do iterativní formy, což je hlavní výzvou pro vývojáře.

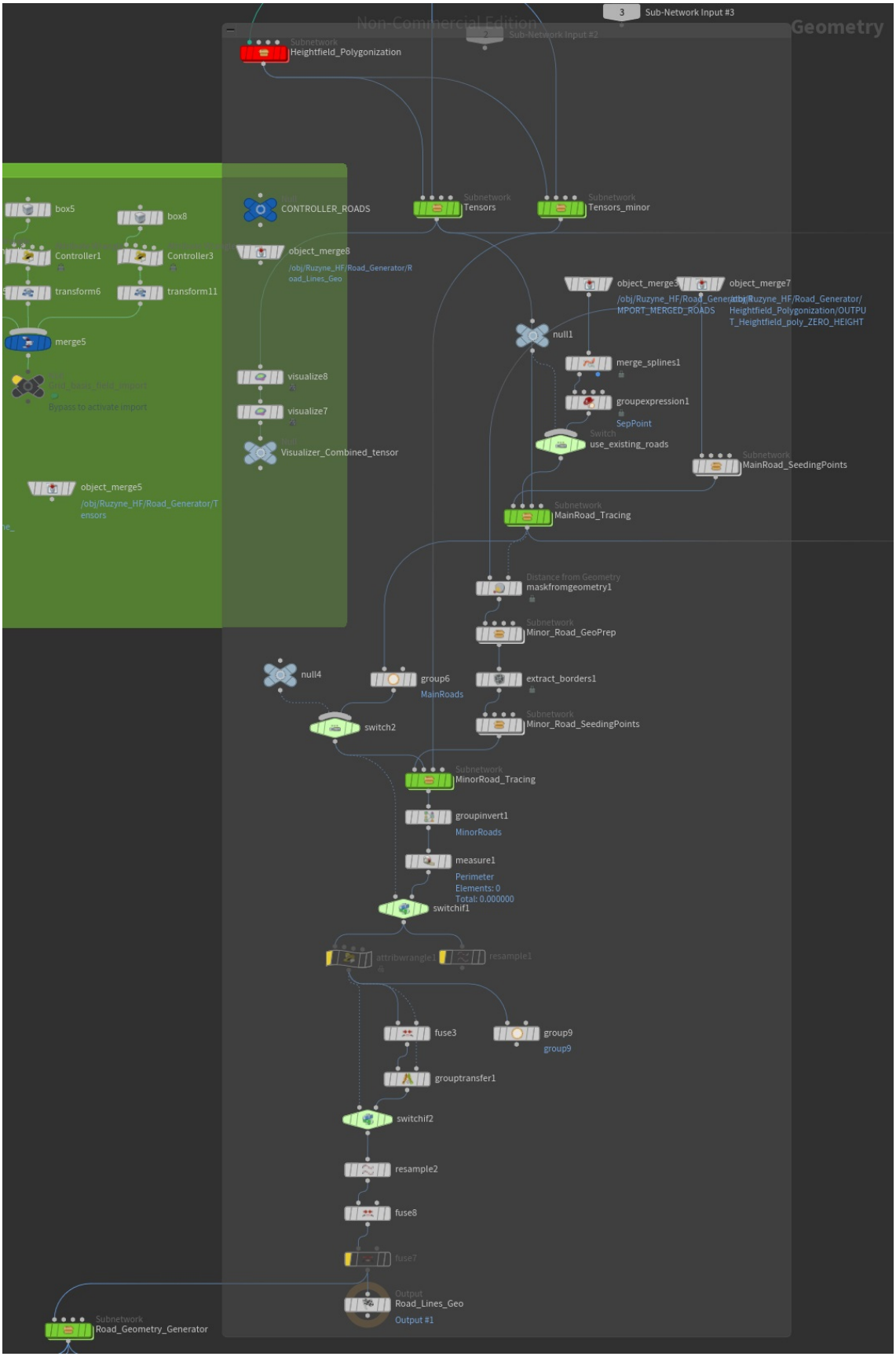
Na základě těchto poznatků bylo rozhodnuto, že projekt bude vypracován v Houdini s bezplatnou nekomerční licencí Apprentice, která zpřístupňuje všechny funkcionality, ale omezuje varianty formátů při exportu a rozlišení pro vykreslování obrázků a jejich sekvencí.

2.5 Cíl

Cílem této práce je prozkoumat metody a vytvořit sofistikovaný generátor urbanistických návrhů, který reaguje na vstupní parametry a data uživatele. Kromě toho musí generátor respektovat soubor definovaných pravidel promítnutých do určité formy jazyka prostředí, ve kterém je generátor vytvářen. Vzhledem k požadavkům urbanistického vývoje a nutnosti vytvářet různé variace by měl být generátor dostatečně jednoduchý na ovládání, aby uživatel nemusel přímo zasahovat do vnitřních parametrů grafických algoritmů. Zároveň je žádoucí, aby i při základní parametrizaci vynášel obstojné výsledky. V neposlední řadě by měl být schopný spočítat základní urbanistické statistiky pro prozkoumávanou oblast.

Výsledky generativních algoritmů by měly být geometrie, které přibližně nebo úplně reprezentují možnosti urbanistické zástavby. Umožní tak uživateli vygenerovat na základě parametrizace různé nebo naopak soběpodobné variace závisející na pseudonáhodných číslech některých z parametrů.

[8] Povrchový operátor (*surface operator*)



Obrázek 2.8: Příklad SOP sítě v Houdini, ve které jsou viditelné uzly podsítě, spojovací operátory, logické operátory, seskupovací uzly a jiné.

Kapitola 3

Návrh řešení a realizace

Následující kapitola pojednává o výběru vhodných metod a jejich realizaci k vytvoření generátoru urbanistických řešení. Algoritmy jsou nejdříve popsány obecně a pak promítnuty do prostředí Houdini. Každá konfigurace městského návrhu je založená na vstupních datech nebo jejich výběru poskytnutých Institutem plánování a rozvoje hlavního města Prahy a volně dostupnými daty. Generátor se zároveň podrobuje souhrnu vybraných pravidel z předchozí kapitoly.

3.1 Návrh algoritmického postupu

Důležitým krokem před zpracováním dat je stanovit algoritmický chod generátoru. Návrh řešení je následujícím způsobem:

1. zpracování dat,
2. tvorba silniční sítě,
3. vymezení zastavitelných ploch na základě silniční sítě,
4. rozdělení zastavitelných ploch na pozemky,
5. extruze geometrií budov,
6. projekce geometrie budov a silnic na terén a úprava terénu.

V průběhu těchto kroků je nutné průběžné výsledky čistit a zároveň na nich provádět statistiku, která určí důležité hodnoty z urbanistického hlediska. Díky průběhu uzlové sítě Houdini je možné téměř absolutně jednotlivé vrstvy separovat.

3.2 Zpracování dat

Pro implementaci algoritmů bylo nutné z poskytnutých dat použít následující vrstvy:

- vrstvu lokalit definujících ořezové polygony řešených oblastí (na obr. 3.1 šedivý polygon překrytý ostatními vrstvami kopírující jeho tvar),
- vrstvu potenciální zástavby (na obr. 3.1 růžově),

- vrstvu existujících (nebo zamýšlených) ulic a cest (na obr. 3.1 žlutě),
- vrstvu parků (na obr. 3.1 fialově),
- vrstvu výškové regulace (na obr. 3.1 modře).

Všechna následující data bylo nutné převést z FileGDB formátu do formátu Shapefile v prostředí QGIS a následně byly projektovány pomocí Web Mercator projekce, aby případně byly kompatibilní s OSM (Open Street Map) daty. V prostředí QGIS bylo tedy nutné vyexportovat do formátu Shapefile vrstvy jednotlivě. Následně byly zpracovány v programu Blender kvůli převodu do 3D dat za pomocí BlenderGIS pluginu. K tomu bylo v QGIS nutné vypočítat střed řešené oblasti a pro Web Mercator projekci posunout střed do souřadnic. Protože jsou oblasti relativně blízko u sebe, je možné použít jejich sdružený střed s opomenutím malé nepřesnosti. Střed je pro data v souřadnicích $(-741756, -1045551)$. Střed mapy byl afinní transformací posunut do počátku a modely byly vyexportovány ve formátu OBJ, aby je Houdini bylo schopné importovat[1]. To s sebou bohužel nese ztrátu dat ve formě atributů, které jsou v daných atributových tabulkách vrstev. Separátně byly taktéž vygenerovány ulice a síť silnic. Všechny geometrie a křivky by měly odpovídat naměřeným vzdálenostem, tudíž není nutné je škálovat. Vrstva výškové regulace je převedena do 3D polygonů a ve výškové souřadnici navíc extrudována na základě maximální podlažnosti.

Poslední vrstvou je výškové pole zmiňované oblasti (na obr. 3.1 odstínech šedivé v okolí polygonů). Celé výškové pole je objemný 32-bitový GeoTIFF rastr o jednom kanálu s rozměry 35000×26001 pixelů a velikostí pixelu 1×1 m, který bylo nutné nahrát do QGIS a vrstvou lokalit ořezat. Pro každou oblast tak vznikl samostatný rastr definující výšku terénu. Pro celý rastr to jsou výšky od 170 m do 421 m nad mořem. Data jsou pod S-JTSK projekcí.

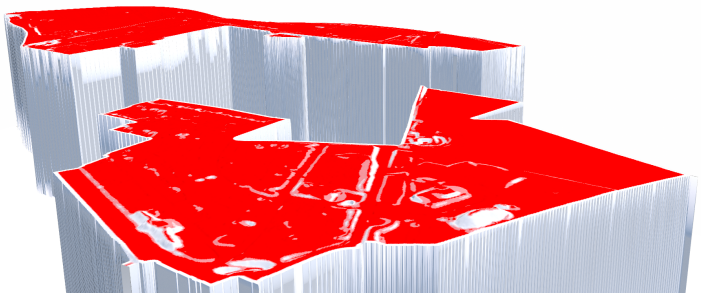
Obrázek 3.1: Oblast Velkého Háje z prostředí QGIS.

Po nahrání dat do Houdini bylo nutné vytvořit geometrie pro jednotlivé oblasti a vytvořit skupiny sloužící těmto oblastem pro vrstvy s ulicemi a výškovými regulacemi.

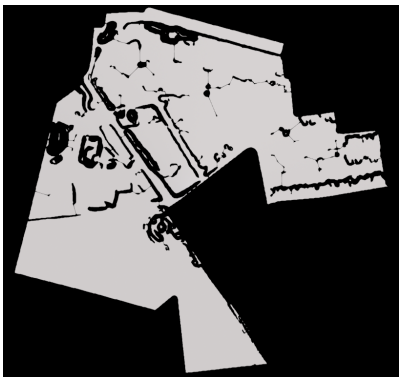
[1] Pro Houdini existují HDA (Houdini Digital Asset) konvertory z Shapefile do formátů vhodných pro Houdini, ale jsou tvořené komunitou a nefungují stoprocentně. Navíc není zaručená jejich nastávající podpora.

3.3 Terén

Fundamentálním vstupem pro algoritmické vytváření je terén. Velikost terénu je nastavena na rozsah řešené oblasti s malým přesahem. Následně je vzorkován mřížkou s délkou hrany 1 m. V Houdini mají výšková pole a terény reprezentaci volumetrické geometrie. Je možné nad nimi pracovat pomocí spousty operátorů včetně simulací eroze, terasování, výřezů a primárně maskování 3.2. Následně je vstupem podsítě řešící jeho polygonizaci.



Obrázek 3.2: Maskování terénu pomocí výšky, zakřivení a sklonu (žádoucí hodnoty červeně).

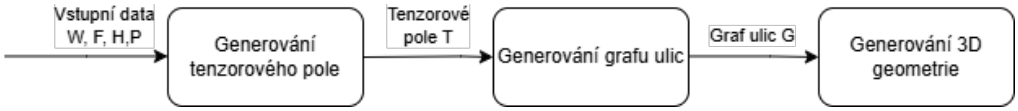


Obrázek 3.3: Vyčištěný polygon terénu podle masky na obrázku 3.2 s dolní limitou masky $l = 0.8$.

Z terénu je následně maska (atribut ležící v intervalu $[0, 1]$) vyříznuta číslem definujícím spodní hranici ležícího v intervalu $[0, 1]$. Maskovat terén není nutné, pokud uživatel nechce omezit řízení následujících kroků, jako je například vhodná plocha zástavby. Tím, že vyříznutí masky nemusí být perfektní, mohou ve vytvořeném polygonu oblasti terénu vzniknout artefakty. Kromě maskování je polygonizace dosažena pomocí operátorů *extract borders, hole* a následně převzorkování na subdivizní křivky. Jsou také vymazány základní degenerativní prvky geometrie (samostatné body aj.).

3.4 Tvorba sítě ulic

Většina metod využívajících generativní struktury jako L-systémy a jiné nedovolují zásah do tvorby nebo úplně nepřipouští uživatelský vstup. Proto byla zvolena metoda využívající tenzorová pole [20]. Pro generátor je totiž důležité, aby vytvářel nejen 3D geometrii silniční sítě, ale také její logickou strukturu, a to na základě čisté parametrizace a uživatelských dat. Nejedná se o čistou kopii zmíněného algoritmu, ale jeho variantu. Algoritmus následuje stejný postup (diagram 3.4). Pro původní algoritmus jsou vstupními parametry binární mapa vodních ploch W, binární mapa lesních a parkových prostředí F, výšková mapa H a mapa populační hustoty P. Pro prozkoumání algoritmu v našem případě budeme zvažovat pouze mapu H ve formě terénu a výškové mapy, protože je rozsah prostředí malý a ostatní funkce se dají nahradit zpracováním dat a parametrizací v Houdini.



Obrázek 3.4: Diagram tvorby uliční sítě.

3.4.1 Tvorba tenzorových polí

V rámci této práce definujme *tenzor* jako 2×2 symetrickou matici s nulovou stopou. Tenzor t má vždy formu

$$t = R \begin{bmatrix} \cos(2\Theta) & \sin(2\Theta) \\ \sin(2\Theta) & -\cos(2\Theta) \end{bmatrix}, R \geq 0, \Theta \in [0, 2\pi). \tag{3.1}$$

Hlavními vlastními vektory t jsou množiny

$$\{\lambda \begin{bmatrix} \cos(\Theta) \\ \sin(\Theta) \end{bmatrix} | \lambda \neq 0\}, \tag{3.2}$$

vedlejšími vlastními vektory jsou množiny

$$\{\lambda \begin{bmatrix} \cos(\Theta + \frac{\pi}{2}) \\ \sin(\Theta + \frac{\pi}{2}) \end{bmatrix} | \lambda \neq 0\}, \tag{3.3}$$

což má za důsledek, že jsou na sebe hlavní a vedlejší vektory kolmé.

Tenzorové pole T je obecně spojitá funkce, která každému bodu $p = (x, y) \in R^2$ přiřazuje tenzor $t(p)$. Bod p je *degenerativní*, pokud $t(p) = 0$, jinak je regulární. Toho můžeme docílit s vzorkováním pomocí mřížky a následně spojité hodnoty získáme použitím bilineární interpolace.

Základem pro tenzorové pole je tedy mřížka vytvořená z polygonu terénu s délkou hrany mezi vrcholy na základě vzdálenosti specifikované uživatelem. Jednotlivá tenzorová pole se mezi sebou dají kombinovat podle

$$T(p) = \sum_i e^{-d_i ||p - p_i||^2} T_i(p), \tag{3.4}$$

kde p je řešený bod, p_i je střed geometrie, která řeší danou oblast (designový prvek), d_i je parametrizovatelný faktor útlumu dané oblasti, která je vybrána objektem i. T_i je bázové tenzorové pole připadající tomuto objektu. Prakticky v prostředí Houdini můžeme procházet jednotlivé objekty ovlivňující tenzorová pole postupně a přičítat je na základní tenzorovou mřížku v následujícím pořadí.

1. Mřížkové tenzory,
2. radiální tenzory,
3. tenzory výškového pole,
4. hraniční tenzory,
5. tenzory existujících cest.

3.4.2 Tvorba mřížkových vzorů

Fundamentálním vzorem pro městské struktury je mřížkový vzor. Parcely jsou generovány ortogonálními strukturami rovnoběžných ulic. Vzor lze definovat například vektorem indikujícím směr pole hlavních vektorů. Hlavní mřížka tenzorového pole se generuje podle uživatelem zadaného úhlu Θ. Pro $\Theta = 0$ hlavní vektor směřuje k ose Z.

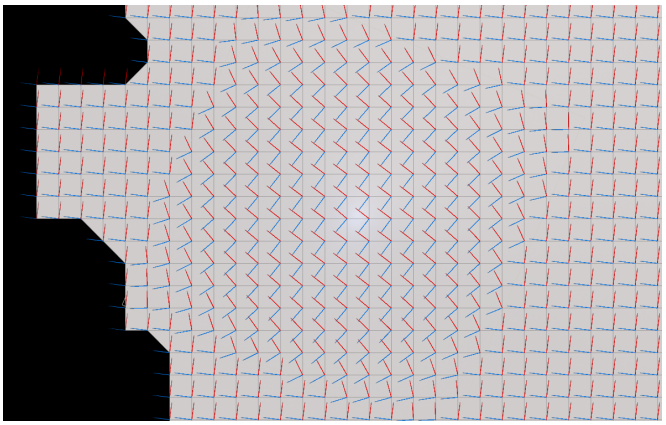
Pro výpočet tenzoru je použit vzorec

$$T(p) = l \begin{bmatrix} \cos(2\Theta) & \sin(2\Theta) \\ \sin(2\Theta) & -\cos(2\Theta) \end{bmatrix}, \tag{3.5}$$

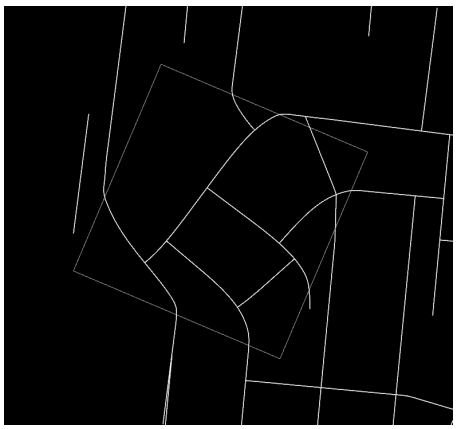
kde l se rovná vzdálenosti geometrického středu designového prvku od vzorkovaného bodu v mřížce.

Protože je zamýšlený úhel známý, stačí už jen definovat parametr útlumu, který určuje aplikační sílu na existující tenzorové pole. Z tohoto důvodu je výpočet založený na pozici středu geometrie, která určuje ovlivňovanou oblast a parametru d, který určuje sílu této oblasti.

Na obrázku 3.5 a 3.6 je možné vidět vliv na tenzorové pole a následně vygenerovanou geometrii pro oblast znázorněnou čtvercem s parametry $\Theta = -57°$ a $d = 11.7$.



Obrázek 3.5: Vizualizace tenzorového pole s mřížkovým prvkem, hlavní vlastní vektory červenými glyfy, vedlejší vlastní vektory modrými glyfy.



Obrázek 3.6: Vygenerovaný graf cest kombinovaného tenzoru mřížkového typu z obrázku 3.5.

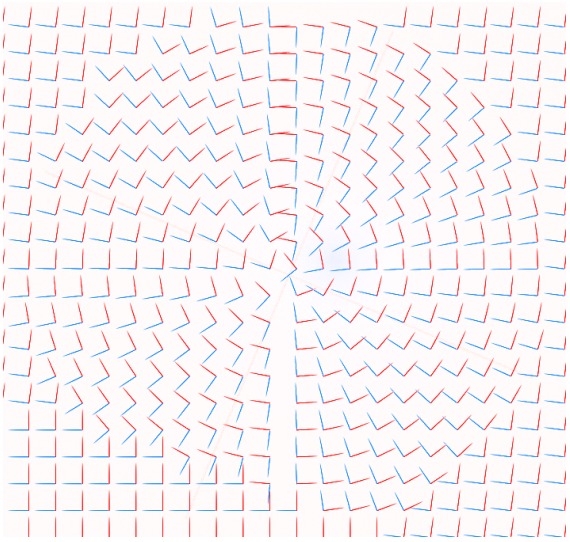
3.4.3 Tvorba radiálních vzorů

Pro tvorbu radiálních vzorů je nutné vytvořit takové objekty, jejichž hlavní proudnice (angl. *streamline*) budou vytvářet pro hlavní vlastní vektory kružnice a pro vedlejší vlastní vektory dostředivé linie. Radiální vzory se vyskytují například v rezidenčních oblastech a nebo v oblastech okolo důležitých monumentů, například Vítězný oblouk v Paříži.

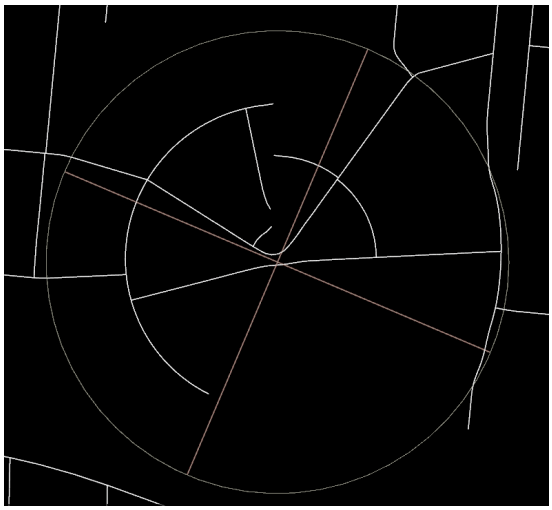
To je dosažitelné analogickým nastavením designového objektu jako u mřížkových vzorů. Liší se ale výpočtem tenzorového pole se středovým prvkem $p_0 = (x_0, y_0)$

$$T(p) = \begin{bmatrix} y^2 - x^2 & -2xy \\ -2xy & -(y^2 - x^2) \end{bmatrix}, \tag{3.6}$$

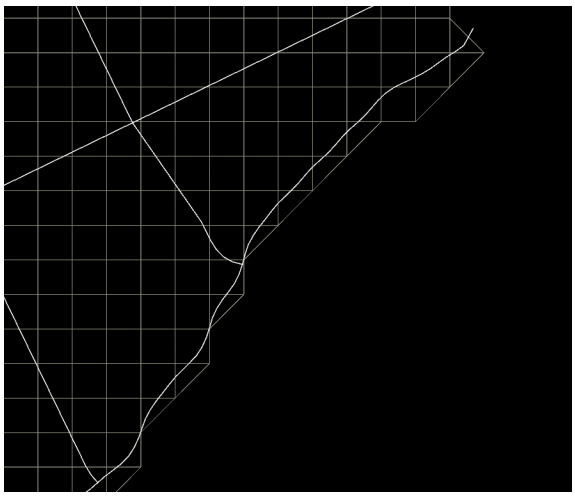
kde $x = x_p - x_0$ a $y = y_p - y_0$ pro bod p.



Obrázek 3.7: Vizualizace kombinovaného tenzorového pole s radiálním prvkem, hlavní vlastní vektory červeně, vedlejší vlastní vektory modře.



Obrázek 3.8: Vygenerovaný graf cest kombinovaného tenzoru radiálního typu z obrázku 3.7, upravená oblast v kružnici.



Obrázek 3.9: Vliv hranice polygonu na generovanou síť.

3.4.4 Vliv hranice na tenzorové pole

Někdy je důležité mít cesty kopírující tvar vedlejší struktury. Proto je vhodné, aby tenzorové pole mělo na okrajích těchto oblastí hodnoty takové, aby hlavní vlastní vektory kopírovaly tvar dané hraniční oblasti. Pro hranici polygonu L tak můžeme vymezit segmenty úseček $u = [A, B] \in L$, které pro bod A ležícím na tenzorovém uzlu mřížky polygonu přímo určují směr hlavního vlastního vektoru tenzoru. Tenzor T_b je následně kombinací všech těchto prvků. Výpočet probíhá analogicky jako u mřížkových tenzorů podle vzorce 3.5 s rozdílem, že

$$\Theta = arctan(\vec{u}_y/\vec{u}_x),$$

$$l = ||\vec{u}||,$$

kde vektor $\vec{u} = B - A$ je z bodů A, B úsečky u.

Tento způsob také simuluje stejné chování pro cesty podél pobřeží (obrázek 3.9).

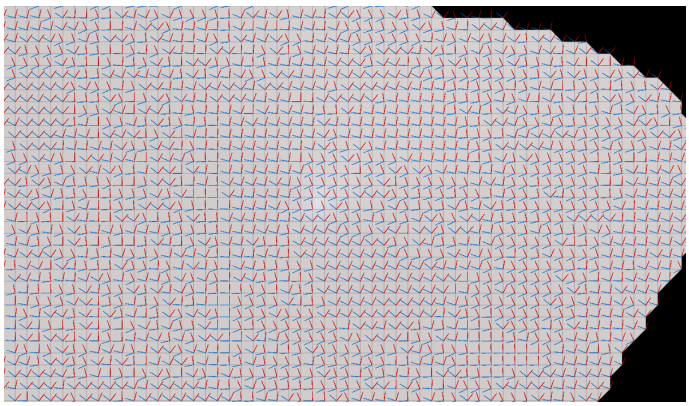
3.4.5 Vliv terénu na tenzorové pole

Převýšení je důležitým omezením pro většinu silničních sítí. Konstruktéři často vycházejí z gradientu výškového pole terénu. Abychom mohli výškové pole H derivovat, je spočítán gradient

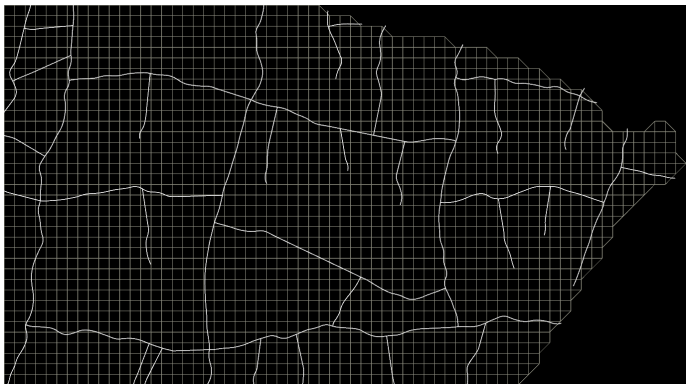
$$\Delta H = (\frac{\partial H}{\partial x}, \frac{\partial H}{\partial y}).$$

Poté stačí definovat tenzorové pole $T(x, y)$ podle rovnice 3.1, kde R nahradíme $R = \sqrt{(\frac{\partial H}{\partial x})^2 + (\frac{\partial H}{\partial y})^2}$ a $\Theta = arctan(\frac{\frac{\partial H}{\partial y}}{\frac{\partial H}{\partial x}}) + \frac{\pi}{2}$. Výpočet je tímto způsobem možný; lze však využít faktu, že pole vedlejších vlastních vektorů tenzoru $T(x, y)$ odpovídá gradientu výškového pole v každém bodě. Můžeme tedy v Houdini využít interní funkce výpočtu gradientu a dostat tak pomocí vzorce 3.3 zpětně parametr Θ. Stále je ale nutné vypočítat R zmíněným způsobem.

Při kombinaci tenzorových polí je v případě terénu započten pouze faktor útlumu d_H pro uživatelskou kontrolu. Pro $d_H \in (0, 0.3)$ jsou zároveň výsledkem cesty vyhovující parkovým či lesním cestám (obr. 3.11).



Obrázek 3.10: Vizualizace tenzorového pole závislého na výškovém poli, hlavní vlastní vektory červeně, vedlejší vlastní vektory modře.

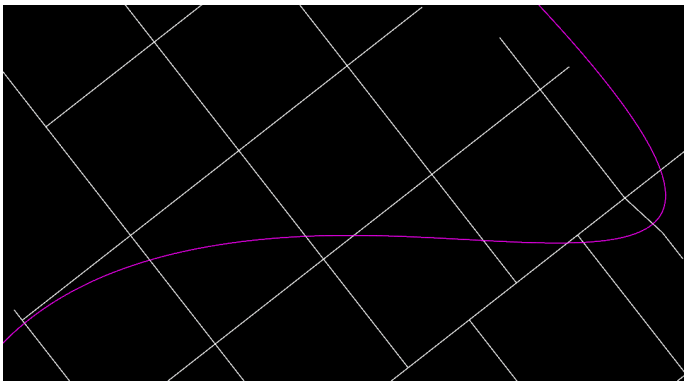


Obrázek 3.11: Vygenerovaný graf cest pomocí tenzoru z obrázku 3.10.

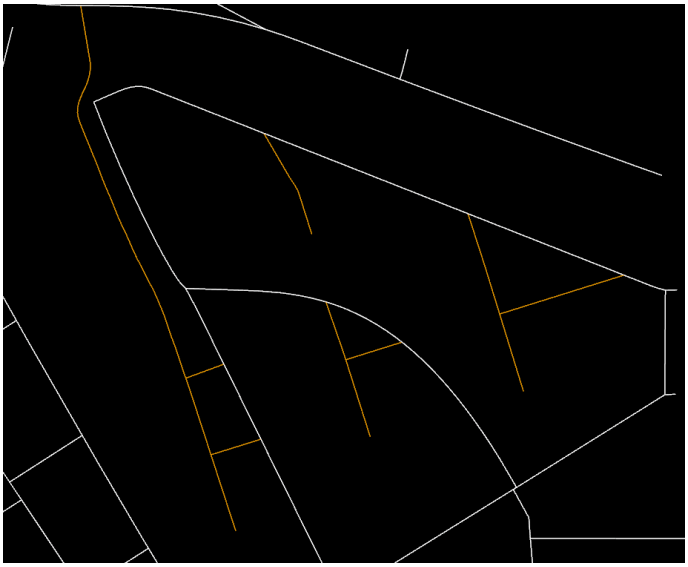
3.4.6 Vliv existujících ulic

Existující trasy dokreslené uživatelem můžeme rozdělit na dva typy - trasy ovlivňující tenzorová pole a trasy překryvné. Trasy překryvné se pouze začlení do sítě a nevytváří nové tenzory ani neomezují existující (obr. 3.12).

Pro typ ovlivňující tenzorová pole je podobně jako u hranice vybrán bod, následně jeho sousední bod a postupně podél křivky extrapolují (v parametrizovatelné vzdálenosti od křivky) hodnoty směru po křivce do existujících vlastních vektorů tenzorů. To je opět provedeno pomocí faktoru útlumu d. Vliv existujících křivek na trasování je možné vidět na obrázku 3.13.



Obrázek 3.12: Generovaný graf ulic (bíle) a překryvná ulice definovaná uživatelem (růžově).



Obrázek 3.13: Existující ulice (bíle) upravující napojování a tvar ulic generovaných (žlutě).

3.4.7 Trasování tenzorovým polem

Pro tvorbu uliční sítě je nutné v tenzorové struktuře trasovat počáteční body. Počáteční body jsou zvolené náhodným rovnoměrným rozmístěním po polygonu na základě hustoty, pevného počtu či manuálně. Následně je jim udána priorita p na základě nejbližší vzdálenosti od hranice polygonu d_b a v průběhu algoritmu priorita na základě vzdálenosti od nejbližšího degenerativního bodu d_d. Výsledná priorita p je tedy vypočítána ve tvaru $p = e^{-d_b} + e^{-d_d}$. Následně jsou počáteční body seřazeny podle priority p tak, aby algoritmus trasování zahájil body blíže k hranici. To zároveň splňuje i požadavek algoritmu, aby uliční sítě rostly primárně z bodů zvolených v

úzkých částech polygonu a nebo přímo na jeho okraji. Výsledkem je síť proudnic typu *polyspline* (polygonní aproximace křivky).

Trasovací algoritmus na základě momentálního koncového bodu na pozici P_i získá směr, ve kterém proudnice poroste. Získá tak v tenzoru hodnotu hlavního nebo vedlejšího vlastního vektoru na základě stádia vývoje. Pro popis algoritmu uvažujme, že trasování začíná hodnotou hlavního vektoru.

Definujme operátor tenzoru T pro hledání vlastního vektoru tenzoru jako $T(P_i, m)$ podle typu vlastního vektoru (hlavní pro $m = 1$, vedlejší pro $m = -1$). Následně se podle zvolené numerické metody posune ve směru $\vec{v}$ určeném příslušným tenzorem a zaznamená si předchozí směr $\vec{v_{pre}}$ (pro první iteraci nulový vektor). Vzhledem k definici 3.2 a 3.3 mají vlastní vektory hodnoty pouze v I. a II. kvadrantu, takže je nutné dodržet pravidlo, že $\vec{v} \cdot \vec{v_{pre}} \leq 0$. Toho je docíleno za pomocí obrácení směru $\vec{v}$ v případě, že podmínka nastane. Výsledná pozice je pak ovlivněna délkou kroku h.

Je důležité stanovit omezující podmínky. Na základě chování jsou chronologicky vybrána následující stopovací kritéria.

1. Trasovaná proudnice narazí do degenerativního bodu,

2. proudnice narazí na hranici řešeného polygonu,

3. trasovaná proudnice protne již existující síť proudnic,

4. proudnice překročí maximální délku.

Pokaždé, co proudnice při růstu dosáhne separační vzdálenosti d_{sep}, algoritmus zasadí jako další počáteční bod proudnice poslední dosažený vrchol p_s a vystřídá směr růstu z hlavních vlastních vektorů na vedlejší vlastní vektory (a obráceně). Zároveň je bod p_s označen jako separační bod.

Navíc algoritmus trasování proudnic prohledává prostor ve směru růstu v maximální dopředné vzdálenosti d_{dop}. V této vzdálenosti prohledává při každé iteraci již vygenerované body a automaticky se přichytí k nejbližšímu bodu. Vždy v malém rozsahu upřednostňuje separační body, aby nevznikla dvě různá zkřížení velice blízko sebe.

Algoritmus je z větší části v Houdini implementován pomocí Wrangle uzlů v jazyce VEX podle pseudokódu popisujícího algoritmus trasování cest. Krátké proudnice jsou na základě uživatelského vstupu opomenuty. Protože algoritmus ze své sémantiky dává smysl primárně v rekurzivním zanoření, musel být pro Houdini přepsán do iterativní formy za použití *feedback-foreach-merge* cyklu (feedback uzel přebírá vygenerovanou geometrii v cyklu, foreach = extract uzel izoluje počáteční body).

Trasovací algoritmus je zároveň vnořeným cyklem pro cyklus prioritní fronty. Do té jsou načteny počáteční body na začátku běhu algoritmu i během jeho průběhu. Výslednou implementací je tedy cyklus přes předgenerované počáteční body sítě pomocí cyklových bloků v SOP síti, ve kterém jsou dvě úrovně vnořených cyklů ve Wrangle uzlu. Může se zdát, že vytvářet větší množství počátečních bodů není nutné, ale tímto se dá vyvarovat brzkého zaniknutí algoritmu.

Algorithm 1 Algoritmus trasování cest

$iter \leftarrow$ maximální počet iterací
$h \leftarrow$ délka kroku
$T \leftarrow$ vstupní tenzorové pole
$majmin \leftarrow majmin$ separačního bodu, pro počáteční bod $majmin = 1$
Inicializace vektorů: $\vec{v} \leftarrow (0,0,0)$, $\vec{v_{pre}} \leftarrow (0,0,0)$
Inicializace čítačů: $d_{uražená} \leftarrow 0$ Vytvoř novou polyline: `prim`
Nastav iniciální pozici P do počátečního bodu P_0
for $i = 1$ **to** $iter$ **do**
    $eigentype$ = "maj_eigen", pokud $majmin == 1$, jinak "min_eigen"
    Vypočti nový směr $\vec{v}$ pomocí numerické metody a vstupního tenzorového pole na základě $eigentype$
    Zkontroluj degenerativní bod
    if $\vec{v} \cdot \vec{v_{pre}} \leq 0$ **then**
        $\vec{v} = -\vec{v}$
    end
    $P_{i+1} = P_i + \vec{v} \cdot h$
    if $\vec{P_{i+1}}$ *mimo hranice* **then**
        Ošetři průnik s hranicí a přidej bod do geometrie
        return
    end
    if $P_i + \vec{v}$ *má průnik s jinou cestou nebo ve vzdálenosti* d_{dop} **then**
        Uprav $\vec{P_{i+1}}$ na nejbližší průsečík
        Přidej nový bod a označ jej jako separační bod
        return
    end
    $d_{uražená} = d_{uražená} + h$
    $\vec{v_{pre}} = \vec{v}$
    if $d_{uražená} \geq d_{sep}$ **then**
        $d_{uražená} = 0$
        Označ bod jako separační bod
        Přidej nový počáteční bod do prioritní fronty s prioritou p a aktualizuj $majmin$ na jeho zápornou hodnotu
    end
end
Result: Geometrie trasované proudnice

Algorithm 2 Zapouzdření algoritmu trasování cest do prioritní fronty

for *každý předgenerovaný bod* P_i **do**
    Definuj počáteční hodnoty pro Algoritmus trasování cest
    Přidej P_i do prioritní fronty pq
    while pq *není prázdná* **do**
        $P \leftarrow$ pq.pop()
        Použij Algoritmus trasování cest na P
    Pro každý nově vytvořený polygon zkontroluj průsečíky cest a ořež body po prvním průsečíku
Result: Síť proudnic

Zvolené numerické metody

Existuje několik algoritmů úzce spjatých s numerickou analýzou. Při vývoji aplikace byla zprvu použita Eulerova metoda, ale při dokončení byla nahrazena Rungeovou-Kuttovou metodou 4. řádu.

Eulerova metoda je pro náš případ definována vztahem 3.7,

$$P_{i+1} = P_i + h \cdot T(P_i) \tag{3.7}$$

kde P_i je pozice na mřížce (nebo mimo ni), T je tenzor s normalizovanými vlastními vektory, h je délka kroku a i je číslo iterace.

Rungeova-Kuttova metoda 4. řádu je pro náš případ definována vztahy 3.8.

$$\begin{aligned} \vec{k_1} &= T(P_i) \\ \vec{k_2} &= T(P_i) + 0.5h \cdot \vec{k_1} \\ \vec{k_3} &= T(P_i) + 0.5h \cdot \vec{k_2} \\ \vec{k_4} &= T(P_i) + h \cdot \vec{k_3} \\ \vec{k} &= (\vec{k_1} + 2 \cdot \vec{k_2} + 2 \cdot \vec{k_3} + \vec{k_4})/6 \end{aligned} \tag{3.8}$$

Následně je podobně jako u Eulerovy metody vypočítána pozice jako

$$P_{i+1} = P_i + h \cdot \vec{k},$$

kde k_i jsou jednotlivé báze metody, $T(P_i)$ jsou hodnoty vlastních vektorů tenzoru na pozici P_i, h je délka kroku a i je iterace.

Rungeova-Kuttova metoda je očividně složitější a náročnější na výpočet, ale vynáší mnohem realističtější výsledky bez zbytečných chyb na křivkách a jiných artefaktech.

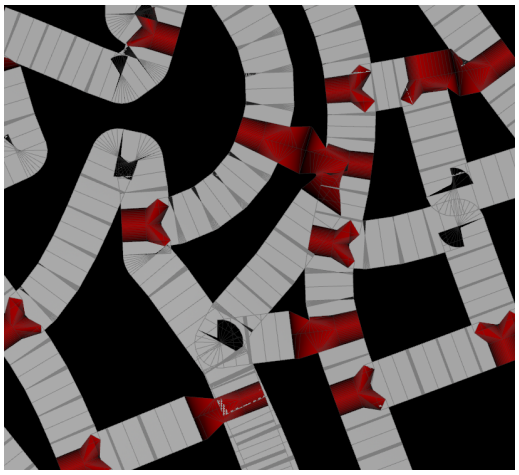
3.4.8 Graf ulic

Pro generování grafu ulic $G(V, E)$ je nutné rozdělit jednotlivé kroky a parametry podle typu řešené sítě ulic. Definujme dva typy grafů - graf hlavních ulic $G_H(V, E)$ a graf vedlejších ulic $G_V(V, E)$. Zároveň platí, že $(G_H(V, E) \cup G_V(V, E)) \subseteq G(V, E)$. Je důležité rozlišit, že grafy hlavních ulic a grafy vedlejších ulic nesouvisí přímo s hlavními a vedlejšími vektory, ale tyto grafy mohou a nemusí mít rozdílná příslušná tenzorová pole. Po vytvoření grafu hlavních ulic poté vnitřky jejich oblastí reprezentují omezující hranice pro graf vedlejších cest. Tento efekt lze zpětně vidět na obrázku 3.13, kde bílé linie značí hlavní cesty, žluté linie cesty vedlejší. Zpravidla je lepší pro vedlejší cesty vytvořit hustší graf, čehož docílíme snížením separační vzdálenosti d_{sep}.

3.5 Tvorba geometrie ulic

Houdini dokáže díky sadě nástrojů LABS, konkrétně nástrojem *Road generator*, generovat geometrii cest. Při základních parametrech nástroje se jeví jako postačující, nicméně při zvětšení šířky geometrie už začínají vyvstávat první artefakty (viditelné na obr. 3.14). Proto byl vyvinut vlastní nástroj, který byl původním nástrojem inspirován, ale pracuje s elementárními prvky geometrie. Mimo jiné byl inspirován videem uživatele Genisease [21].

Protože graf importovaných silnic $G(V, E)$ nemá ověřené body křižovatek (separačních bodů), musí se jeho geometrie prohledat a zvolit zpětně i ty body, které mají valenci (stupeň) $val(v_i) \geq 3$. Následně je bodům přiřazen identifikátor příslušného primitiva, kterému náleží v případě, že jeden z bodů na křivce je separačním bodem. Pokud tomu tak není, jsou tyto cesty označeny jako cesty vnější. Zbývající části grafu musíme opět rozdělit na konkrétní křižovatky a cesty k nim vedoucí, které jsou zároveň cestami vnitřními. Toho je docíleno za pomoci parametrů určujících délku křižovatky a spodní limity délky silnic mezi nimi. Je nutné zdůraznit, že délka ramen křižovatky nemusí být všude stejná právě kvůli spodní limitě délky vnitřních segmentů silnic (respektive relativní pozici křižovatek).



Obrázek 3.14: Artefakty LABS Road Generator ve formě překrývající se geometrie, nesmyslných geometrií křižovatek (červeně) a špatně natočených ploch.

Zároveň je důležité všechna primitiva uličního grafu rozdělit na poloviny a jejich pořadí bodů v druhé polovině otočit opačným směrem pro souřadnici u v rámci UV souřadnic, aby křivky rostly vždy směrem od křižovatky; tyto úpravy umožní algoritmu v budoucnu pracovat správně.

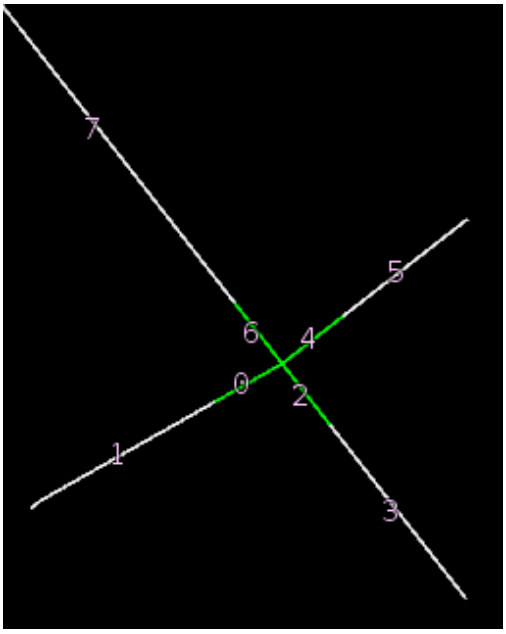
Následně je na body všech rozdělených částí přiřazen parametr *pscale* určující škálování bodu, což ovlivní šířku výsledné geometrie a zároveň tak můžeme rozlišit šířku jednotlivých sítí silnic podle jejich významnosti (kapitola 2.2.4). Stejným způsobem je nutné provést úpravy pro vnější primitiva grafu silnic.

3.5.1 Geometrie křižovatky

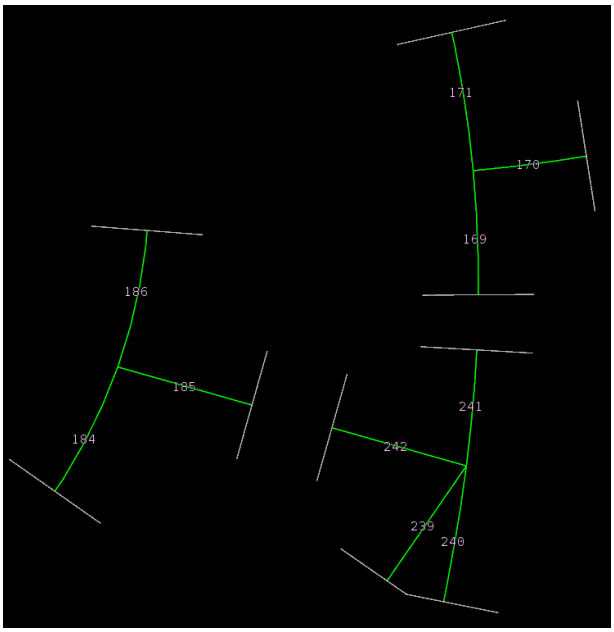
Důležitým krokem pro přípravu geometrie je seřazení cest uvnitř konkrétní křižovatky k příslušné koncovce. Zároveň se tím vytvoří podmínky pro definici okolních hran křižovatky. Toho je možné dosáhnout pomocí radiálního seřazení primitiv podle jejich průsečíku (středu křižovatky). Je zvolen střed křižovatky S a následně normalizovaný vektor $\vec{v_i} = norm(C_i - S)$, kde C_i je těžiště primitiva i. Poté už stačí jen získat úhel $\sigma = atan(\frac{\vec{v_x}}{\vec{v_z}})$ (křižovatka je v rovině XZ). Podle úhlu seřadíme primitiva vzestupně a dostaneme se k výsledku na obrázku 3.15.

Dalším krokem algoritmu je promítnutí kolmých průmětů konců silniční sítě na křižovatky. Příklad s délkou $10m$ a spodní limitou vnitřních částí $2m$ je zobrazen na obrázku 3.16, kde vnitřní vodící linie křižovatek jsou znázorněny zeleně a kolmé průměty bíle. Je důležité si povšimnout, že kolmé průměty silnic u primitiv s číslem 239 a 240 ve spodní části obrázku jsou spojené v jejich průsečíku. To je kvůli situacím, kdy úhel v křižovatce mezi dvěma napojujícími ulicemi je malý relativně k šířce ulice. Požadovaného efektu je docíleno zjištěním průniku dvou kolmých průmětů, které jsou následně ořezány podle délky a jejich směru do křižovatky. Kdyby se průměty nespojily a pronikaly dále, vznikl by artefakt s překryvem plošek a jejich otočením (obr. 3.17).

Dalším krokem je rohy průmětů po dvojicích spojit s příslušnými středovými



Obrázek 3.15: Seřazená primitiva geometrie křižovatky, vnitřní část definovaná délkou zeleně, okolní vnitřní cesty bíle.



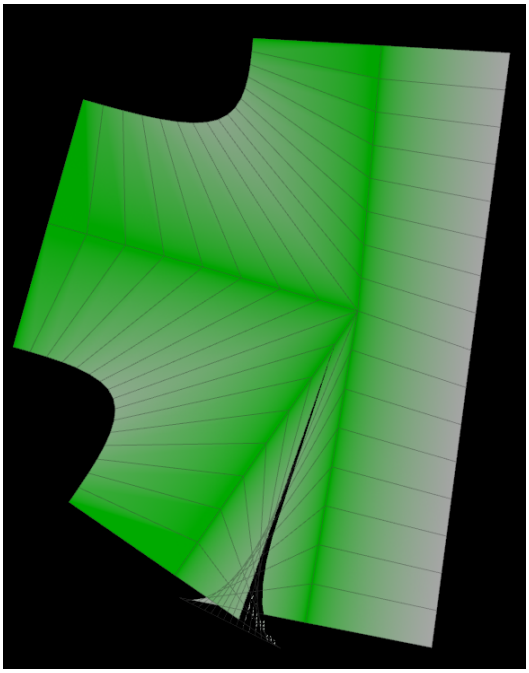
Obrázek 3.16: Vodící geometrie křižovatek, vnitřní část křižovatky definovaná délkou zeleně, kolmé průměty okolních vnitřních cest bíle.

liniemi křižovatky a vyplnit je plochami. U posledního seřazeného primitiva je důležité otočit směr a vrátit se k prvnímu primitivu v pořadí. Princip zní jednoduše, ale vzhledem k nutnosti úpravy vstupů geometrie na nízkou úroveň je potřeba vykonat sérii operací zahrnující vnořené cykly, výřezy geometrie, čištění geometrie, převzorkování (to je závislé na délce jednotlivých ramen křižovatky) a průměty pomocí nejkratších vzdáleností bodů. Spojení primitiv je dosaženo pomocí SOP *skin* a převzorkováním s postupným přiřazováním plošek k příslušným bodům. Je nutné, aby převzorkování odpovídalo vždy sudému číslu, čímž bude dodržena čtyřhrannost polygonů. Jedná se pravděpodobně o nejsložitější cyklus v celém projektu. Správný výsledek řešící křižovatku 4 ulic ve hraničním případě je zobrazen na obrázku 3.18.

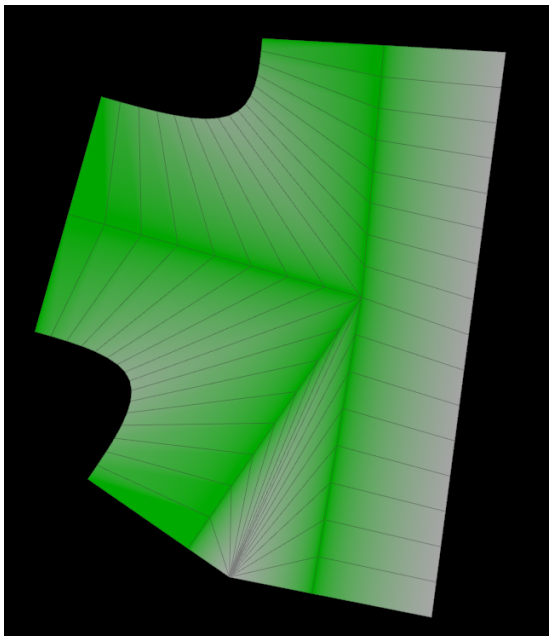
Algoritmus umí vytvářet geometrie křižovatek s více napojeními, než jsou 3 nebo 4 (obr. 3.19), ale problém spočívá v nutném ladění hodnot délky křižovatek a limit vnitřních částí mezi křižovatkami při těsnější koincidenci křižovatek jako takových. Aby se artefakty vzniklé tímto způsobem daly odstranit, musel by se algoritmus vyřešit o logiku spojování křižovatek samotných, nebo lépe problém vyřešit v rámci grafu sítě ulic.

3.5.2 Geometrie silnic

Tvorba geometrie silnic je oproti generování křižovatek jednodušším procesem. Je nutné spojit geometrii rozdělených segmentů vnějších silnic se segmenty silnic mezi křižovatkami (primitiva zůstávají separátní, ale chovají se jako jedna geometrie). Na segmentovanou síť silnic je pak aplikován SOP *sweep* s módem vyplňování geometrií stuhy. Díky předem nastavenému atributu *pscale* tak získáváme geometrii v žádané šíři silnic. Nesmíme zapomenout na otočení UV souřadnice u zpět na

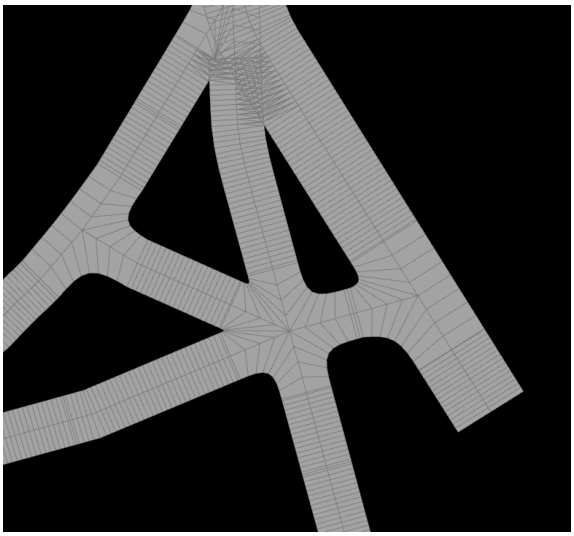


Obrázek 3.17: Artefakt způsobený blízkým napojením dvou silnic na křižovatku.



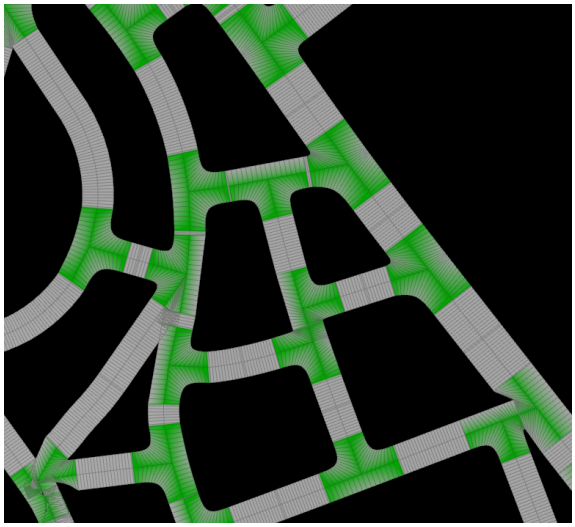
Obrázek 3.18: Opravený artefakt.

správnou orientaci. To je provedeno výpočtem nenulového ovinutí ploch[2] [22]. Pro primitiva se záporným koeficientem ovinutí w je UV souřadnice u otočena opačným směrem.



Obrázek 3.19: Vygenerovaná geometrie křižovatky s 5 vstupními ulicemi, nahoře konsolidace artefaktu pomocí booleanovských operací.

[2]Ve dvourozměrné počítačové grafice je pravidlo nenulového ovinutí způsob, jak určit, zda daný bod leží uvnitř uzavřené křivky.



Obrázek 3.20: Geometrie silnic (šedivě) a křižovatek (zeleně). Na pravé straně širší pás geometrie silnic a křižovatek. V levém dolním rohu a pravém dolním rohu lze vidět artefakt způsobený koincidencí křižovatek.

3.6 Tvorba bloků zastavitelných ploch

Po tvorbě geometrie silniční sítě z grafu ulic musíme vymezit oblasti, které jsou z původního polygonu zastavitelné. Toho lze docílit booleovskou operací odečtení geometrie silnic od polygonu (SOP *boolean subtract*). Následně byly do jednotlivých částí rozděleného polygonu vsazeny plošky definované hranicí s odstupem d_p, které jsou následujícím krokem odečteny od polygonu zastavitelných ploch oblasti. Zároveň tyto plochy definují oblasti chodníků.

Po provedení výše uvedených operací už jsou exaktně definovány plochy zastavitelných bloků. V průběhu už můžeme použít pravidlo definované v sekci 2.2.3 a zbavit se tak ploch nevhodných k zástavbě.

Zároveň tak dosahujeme obecnosti algoritmů a možnosti pro uživatele, kdyby generátor geometrie silnic nechtěl použít a importoval vlastní 3D data splňující stejné konvence.

3.6.1 Rozdělení stavebních bloků

Pro jednotlivé pozemky v rámci stavebních bloků byly bloky rozděleny subdivizním algoritmem. K tomu posloužil nástroj Houdini *LABS Lot Subdivision*, který rozděluje polygon na panely s parametrizovatelnou minimální rozlohou a seskupováním panelů, díky kterému panely získávají zajímavější tvary. Nástroj ale není sám o sobě dostatečně robustní a jeho ideálním vstupem jsou polygony konvexního tvaru (hraniční případ pro nekonvexní polygon je rozdělení pouze v jedné z částí celého polygonu). To ale neplatí pro náš případ, kdy oblasti ohraničené silnicemi mohou mít různé typy výčnělků (respektive dutin uvnitř polygonu).

Téměř nutným vylepšením pro orientaci pozemků v rámci bloku je vyrovnání k ose X nebo Z podle nejdelší hrany zjednodušeného tvaru řešeného polygonu během subdivize. Během tohoto procesu je vhodné pro vyrovnání izolovat body, které jsou uvnitř generované silniční sítě a ne na okraji celkového polygonu řešené oblasti. Dalším kritériem je zmiňovaná konvexita polygonů, pro kterou je instinktivním a pravděpodobně robustním procedurálním řešením vytvořit konvexní obálku pro izo-

lovaný polygon, tuto obálku rozdělit a následně rozdělení promítnout na stavební blok (obr. 3.22). Mohou tak vznikat artefakty kvůli promítání geometrie konvexní obálky na původní. Bylo ale zjištěno, že přiblížením se konvexitě díky parametrizovatelné subdivizi polygonu mřížkou stavebního bloku v kombinaci s nástrojem *LABS Lot Subdivision* je postačující a vynáší mnohem lepší výsledky (obr. 3.21), a to z hlediska orientace i dělení bloků.

Obrázek 3.21: Rozdělené bloky pomocí nástroje *LABS Lot Subdivision* s osovým vyrovnáním a předrozdělením bloků pomocí mřížky.

Obrázek 3.22: Rozdělené bloky pomocí nástroje *LABS Lot Subdivision* s osovým vyrovnáním a konvexní obálkou.

3.6.2 Příprava parcel pro geometrie budov

Pro každý stavební blok je nutné připravit parcely tak, aby byly vhodné pro zástavbu nejen z hlediska pravidel územního plánu, ale i pro geometrické účely. Uživatel tedy může odstranit parcely, které nejsou sousedící s žádnou ulicí nebo krajem celého polygonu. Tímto vznikne v rámci bloků vnitroblok příslušných parcel. Zároveň jsou znovu odňaty parcely podle pravidla v sekci 2.2.3.

Následně je pro každou parcelu vytvořeno parametrizovatelné vnitřní vsazení plošek pomocí SOP *PolyExpand2D*. Vsazením definujeme vnitřní hrany geometrie, od kterých budou extrudovány geometrie samostatných budov. Pro těsnou zástavbu se tento parametr blíží nule. Během vsazení se může stát, že se potenciální geometrie budov rozdělí v úzké části parcely a vzniknou tak dva samostatné objekty. Ty jsou opět ošetřené pro splňování podmínek zástavby. Parcela i její geometrie přesto stále náleží pozůstalému objektu. Příklad tohoto případu je možné vidět na obrázku 3.24.

3.7 Tvorba budov

Pro jednoduchost jsou definovány dva typy budov, a to budovy typu O s maximální podlažností do 4 podlaží a budovy typu A s minimální podlažností rovnou 5 podlaží a omezením RPB[3] nad 500. Zároveň musí obdélníková obálka pro typ A základů splňovat poměr stran alespoň 1:2. Pokud tato kritéria nejsou splněná, budova je opět označena typem O bez restrikce podlažnosti.

Dále bylo vystaveno pravidlo, že pokud orientovaná obdélníková obálka budovy nesplňuje poměr stran zadaných uživatelem, budova nebude vytvořena. Díky tomuto pravidlu jsou tedy odstraněny dlouhé budovy. Výchozí hodnotou je poměr stran 1:4.

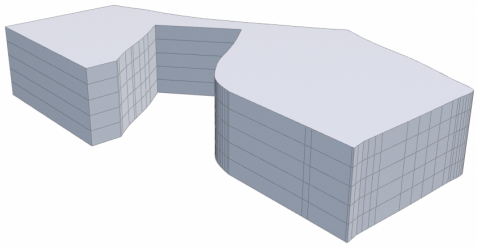
Počet podlaží je definován vstupní geometrií určující podlažnost (na obr. 3.1 kolmý průmět modře). Tato geometrie obklopuje tvar lokality a podléhá extruzi v souřadnici Y celočíselnou výškou určující počet podlaží. Vzorek tohoto parametru pro parcelu se počítá z těžiště parcely. Pokud geometrie neexistuje nebo je výška rovná nule, je maximální počet podlaží pro budovy typu A roven 12, pro budovy typu O je roven 4. Některé parametry budovy jsou ovlivněny pseudonáhodným číslem s.

Geometrie budovy je následně vytvořena opakovanou extruzí v souřadnici Y pomocí SOP *PolyExtrude*. Počet iterací je založen na minimální a maximální podlažnosti a je vybírán náhodně. Při iteracích jsou body extrudovány náhodně o $2.7m$, $3m$, nebo $3.3m$ a transformovány na příslušné pozice. Zároveň je pro každé podlaží vytvořena vnitřní geometrie kopírující kolmý průmět geometrie. Zmíněné parametry lze upravovat podle požadavků posuzovaného územního plánu.

3.7.1 Tvorba střech

Z důvodů neurčitosti geometrií kolmých průmětů byly pro prozkoumání algoritmů vybrány 3 typy střech - střecha rovná, střecha valbová a střecha valbová s mansardami. Budovy typu O mají přístupné všechny typy střech, zatímco budovy typu A jsou omezené na střechy rovné a střechy valbové s mansardami. Pro střechu rovnou je geometrie velice jednoduchá, a tak se pouze zkopíruje kolmý průmět a je přidán na vrch budovy. Pokud algoritmus zjistí, že vytvořil geometrii nevhodnou pro danou budovu, je nahrazena střechou rovnou.

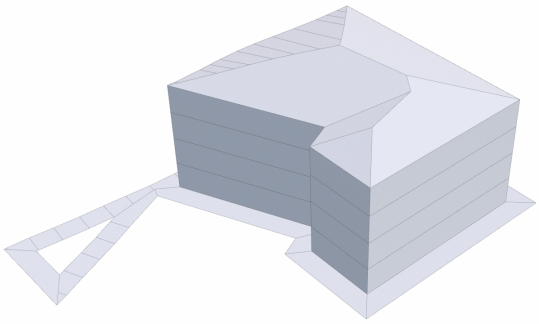
[3]Regulovaná plocha budovy



Obrázek 3.23: Pětipodlažní budova typu *A* omezená výškovou regulací s rovnou střechou.

Střecha valbová

Střecha valbová je generována pomocí kolmého průmětu nejsvrchnějšího patra. Je opět generována pomocí SOP *PolyExpand2D*. Při tomto kroku je uchováván atribut vzdálenosti d_e, který odpovídá vzdálenosti vytvořených vsazených bodů od původní hrany nebo průsečíku hran (respektive rohu). Následně jsou vybrány vytvořené vnitřní body a posunuty do vzdálenosti tak, aby odpovídala intervalu $10° - 35°$. Plocha omezující střechu zespod se nezapočítává mezi podlaží.



Obrázek 3.24: Parcela s extrudovanou čtyřpodlažní budovou typu *O* s valbovou střechou a odebranou nevyhovující částí způsobenou vsazením.

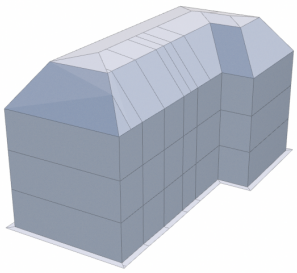
Střecha mansardová

Tento typ střechy je generován analogicky jako střecha valbová s tím rozdílem, že je rozdělená na dvě části, a to část mansardovou a svrchní část střešní (valbová). Mezi těmito částmi se už nepřidává podlaží, část omezující střechu zespod za podlaží naopak budeme považovat kvůli budoucímu průzkumu obytných ploch. Část mansardová má mnohem větší úhel sklonu než část střešní ($60° - 75°$). Zároveň pro valbu platí, že může mít sklon v rozmezí $0° - 35°$.

3.8 Projekce geometrie na terén

Posledním krokem je už pouze upravit terén a provést na něj projekci geometrie. Na terén jsou chronologicky projektovány geometrie silnic s chodníky a následně budovy. Důvodem je vyrovnání terénu pod silnicemi a chodníky po jejich šíři.

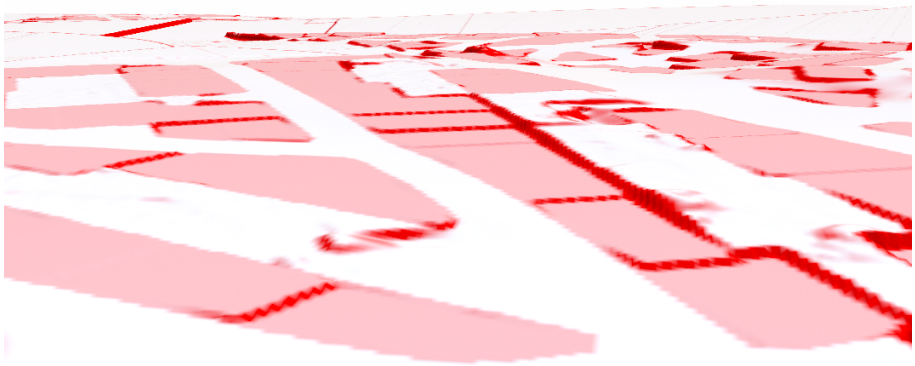
Toho je docíleno za pomocí maskování geometrie na terén a SOP $HeightfieldFlatten$ pod určitým počtem iterací (čím více iterací, tím rovnější povrch). Na vyrovnaný povrch je následně geometrie projekcí pomocí výšky přesunuta pomocí vzorkování souřadnic X a Z každého bodu a výškového atributu terénu.



Obrázek 3.25: Třípodlažní budova typu O s mansardovou střechou a obyvatelným podkrovím.

3.8.1 Projekce budov na terén

Projekce budov je provedena obráceným způsobem, než u ulic a silnic. Se vstupním parametrem upraveného terénu pro silnice a chodníky totiž budovy projektujeme nejdříve pomocí průměrné výšky h_{avg} bodů parcel přilehlých k chodníkům nebo ulicím. Body příslušné vnitroblokům nejsou brány v úvahu. Následně jsou všem bodům budovy přičteny výšky h_{avg}. Zároveň jsou kolmé průměty budov maskovány na terén s menším přesahem a všechny výšky terénu jsou ve zvolené oblasti nahrazeny hodnotou h_{avg} pro každou parcelu (obr. 3.26).



Obrázek 3.26: Vizualizace zarovnání ploch terénu, parcely růžově, velké upravené sklony červeně (přízemní pohled).

Kapitola 4

Výsledky a zhodnocení

V následující kapitole jsou prezentovány výsledky diplomové práce. Výsledky jsou zhodnoceny, porovnány s existujícími vizualizacemi a zasazeny do kontextu. Kapitola popisuje nad nimi vytvořenou statistiku s doprovodnými vizualizacemi. Na závěr jsou posouzeny možná vylepšení generátoru a vliv umělé inteligence na řešenou problematiku.

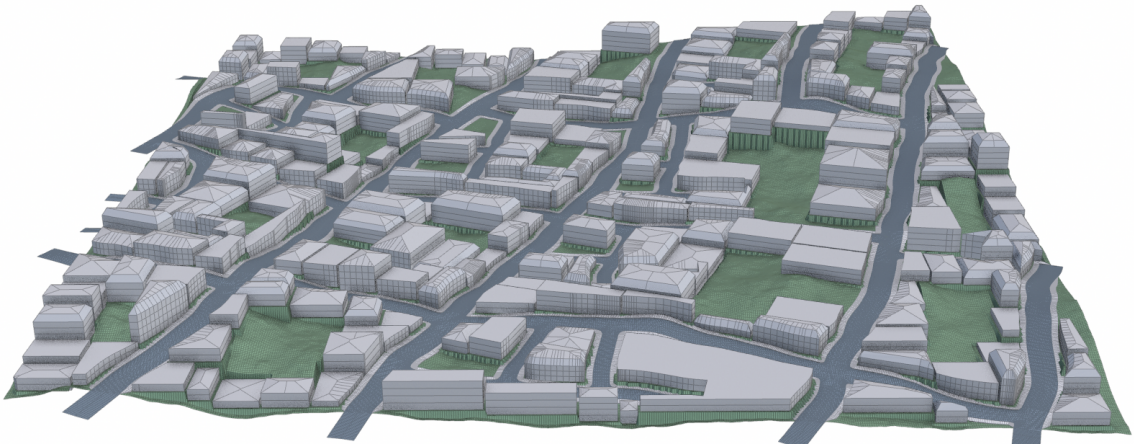
4.1 Výsledky

Cílem práce bylo vytvořit procedurální generátor urbanistických návrhů, který reaguje na vstupní parametry a data uživatele. Parametrizovatelnost v rámci algoritmů je obsáhlá a v prostředí Houdini dovoluje okamžitý zásah do generované geometrie v libovolné části algoritmů. Pro parametrizaci byly vytvořeny ovládací panely v uzlech mimo konkrétní SOP pro následující prvky:

- Chování tenzorů
  - Ovládání rozlišení tenzrového pole
  - Ovládání tenzorového pole podle výškového pole a objektů a jejich vliv
  - Import objektů existujících cest a jejich vliv na tenzorové pole, jejich zpracování
  - Ladící prvky geometrie
- Generátor cest - analogicky pro vedlejší cesty
  - Základní směr grafu ulic
  - Ovládání chování křivek (počet iterací, délka kroku, separační vzdálenosti, počáteční body, dopředná vzdálenost)
  - Uplatnění pravidel minimální délky
  - Šířka geometrie silnic, délky křižovatek a jejich rozlišení
  - Možnost volby mezi trasovacími algoritmy
- Procedurální zástavba
  - Maskování polygonu lokality na základě vlastností
  - Ovládání subdivize bloků a příslušných parametrů

- Omezení minimální plochou polygonů, bloků a kolmých průmětů budov
- Vymezení zastavitelných bloků na základě výběru uživatele
- Ovládání odsazení budov od hranic pozemku, tvorby vnitrobloků, poměr stran budov
- Ladící prvky geometrie

Díky těmto parametrům jsou výsledky dostatečně různorodé. Pro parametry jsou nastavené výchozí počáteční hodnoty, které vyhovovaly některým z předchozích parametrizací a samostatně vynášejí obstojné výsledky i pro obecný vstup (obr. 4.1). Protože proces nepodlehl uživatelskému testování, nelze posoudit, zda je ovládání dostatečně jednoduché pro urbanistu znalého 3D editorů.



Obrázek 4.1: Jednoduchý výstup algoritmu na rozmanitém vygenerovaném terénu nespecifikovaném jinými vstupními parametry s poměrem zástavby budov 53.21% a poměrem zástavby silnic 20.91% vůči ploše terénu.

Kredibilita generátoru závisí na správné parametrizaci. Při správném nastavení parametrů je poměrně jednoduché generovat procedurální varianty zástavby. Varianta zasazená do 3D modelu okolí je zobrazena na obrázku 4.3, původní zástavba z 3D modelu na obrázku 4.2. Další výsledky generátoru jsou na obrázcích 4.15, 4.16, 4.17 a 4.18.

Jednotlivé části generátoru jsou vzájemně nezávislé. Je možné konstatovat, že po provedení jednoduchých úprav jsou části řešící tvorbu grafu ulic, geometrie ulic a následně procedurální zástavby naprosto separátní a dají se nahradit jinými HDA[1]. Uživatel má možnost importovat vlastní data a získat výsledky pomocí externích nástrojů. K tomu je nicméně potřeba více než základní znalost Houdini a struktury používaných dat. Díky této obecnosti se jedná o silný nástroj.

[1]Houdini Digital Asset

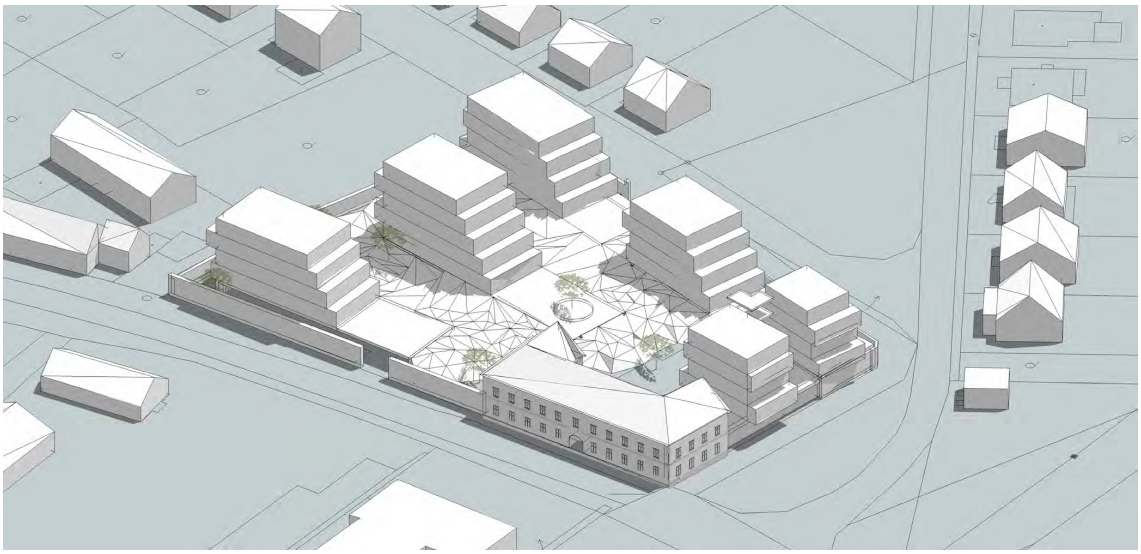
Obrázek 4.2: Původní zástavba lokality Na Dědině.

Obrázek 4.3: Realizovatelný návrh zástavby lokality Na Dědině zasazený do okolního prostředí s plochou 22.123ha. Poměr ploch kolmých průmětů budov je 53.26% a poměr ploch komunikací je 20.53% vůči ploše terénu.

4.1.1 Vizuální výsledky

Výstupem generátoru jsou poměrně jednoduché, avšak různorodé geometrie se specifikovatelným rozlišením podle uživatele. Geometrie nemá textury, jsou jí pouze přiřazeny konstantní barvy. K vizualizacím je upřednostňováno konstantní stínování plošek s drátěným modelem pro naznačení podlažnosti. Při vizualizaci má uživatel možnost použít zastínění okolím (*ambient occlusion*) pro lepší rozlišení jednotlivých objektů, které je vhodnější než používání stínů. Uživatel není omezen čistě na výstupy finální geometrie, ale může dělat výstupy s různými vizualizacemi i v průběhu generování (například obr. 4.7 a 4.8) v nejrůznějších formátech, které Houdini povoluje.

Může se zdát, že vizuální výsledky nejsou postačující kvůli nedostatečnému detailu. Opak je pravdou, protože pro průzkumy a prvotní návrhy vizualizací se dosahuje právě takové úrovně detailu jako na obrázku 4.4. Podle specialistů z IPR "realistické"a barevně rozmanité vizualizace, častokrát s kýčovitým západem slunce a podobnými prvky, naopak odvádějí pozornost od návrhu jako takového a schylují se k nepodstatným detailům (obr. 4.5); tyto vizualizace jsou vhodné spíše pro komerční využití na realitním trhu a prezentaci návrhů pro veřejnost, ne pro pracovní účely stavebních specialistů, architektů a urbanistů. Dále je možné porovnat výsledky s vizualizací na obrázku 4.6.



Obrázek 4.4: Vizualizace z iniciační studie Dukelské kasárny v Jindřichově Hradci, městský architekt Lukáš Soukup, 2023.

Obrázek 4.5: Urbanisticko-architektonická studie areálu bývalých kasáren v Novém Městě nad Metují, Ateliér Zídka, architektonická kancelář spol. s r.o., 2020.

Obrázek 4.6: Vizualizace Smíchov City - Sever, Kuba & Pilař architekti [23].

4.1.2 Dodržení pravidel Metropolitního plánu

Pro ověření dodržování pravidel Metropolitního plánu byla na základě sekce 2.2.2 vytvořena vizualizace vyznačující realizovatelnost zástavby jednotlivých bloků. Ostatní jsou zahrnuty v parametrech geometrie při její tvorbě jako takové. Obrázek 4.7 ukazuje lokality Nové Ruzyně, na které jsou bloky splňující kritéria zástavby znázorněny zeleně, bloky nesplňující podmínky přechodem žluté až červené barvy podle jejich závažnosti. Bílými liniemi jsou vyznačeny kolmé průměty budov. Jsou zde zahrnuty i ty bloky, které nepodlehly zástavbě. Kritériím nepodléhají dva velké bloky zleva, ostatní chybné bloky jsou střední velikosti (opět definováno podle sekce 2.2.2).

Obrázek 4.7: Vizualizace splnění kritérií zástavby bloků vygenerované na lokalitách Na Dědině (vlevo) a Nová Ruzyně (vpravo).

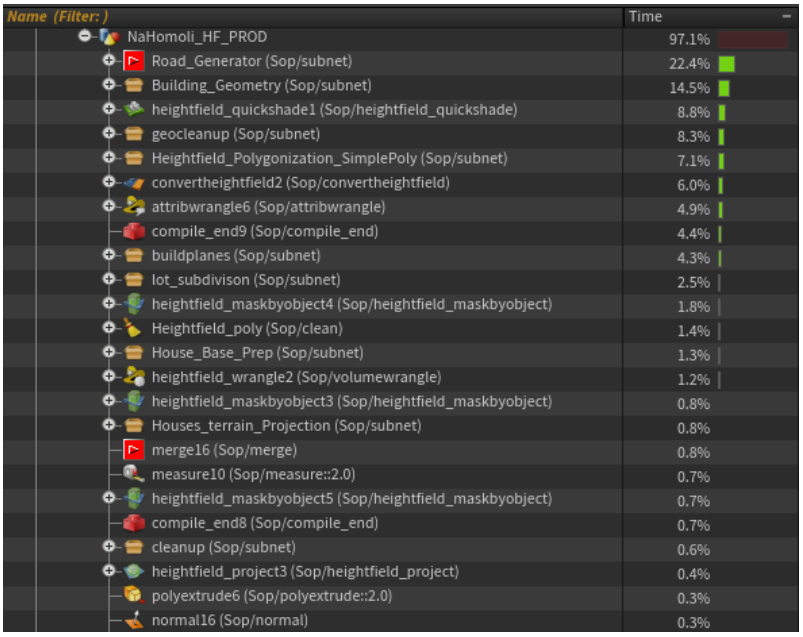
4.1.3 Statistiky

Kromě dodržení pravidel zástavby bloků v sekci 4.1.2 byly na konkrétních případech vypočítány následující statistiky. Pro každou oblast je to poměr celkové plochy zástavby kolmých průmětů budov vůči terénu a celková plocha všech obytných podlaží (vizualizace na obr. 4.8). Druhotně důležitá je délka komunikací a poměr celkové plochy komunikací vůči terénu kvůli aproximaci ceny za vývoj infrastruktury.

Důležitou statistikou je i efektivita algoritmů, která je vypočítána poměrem délky trvání jednotlivých částí vůči celkovému chodu programu 4.9. Některé z částí podlehly optimalizaci ve smyslu rozřazení procesů na více vláken a kompilaci kódu pomocí SOP *CompiledBlock*.

Bylo provedeno 20 testů. Testovacím zařízením byl osobní počítač s procesorem Intel Core i5-6400, 16 GB DDR4 RAM a grafickou kartou NVIDIA GTX 1060 s 3 GB VRAM. Je proto těžké posoudit, zdali jsou kvůli nedostatečnému výkonu stroje výsledky testování relevantní. Zároveň ukazují, že i starší osobní počítač je při používání generátoru postačující. Celkové generování zabírá v průměru kolem 40 sekund. Na obrázku 4.9 je možno vidět jednotlivé časové poměry částí.

Obrázek 4.8: Vizualizace podlažnosti budov z obrázku 4.15. Na obrázku je 724 podlaží o celkové ploše 27.3854*ha*. Úroveň podlažnosti je vyznačena gradientem z černé do modré barvy.



Obrázek 4.9: Poměr časů z 20 testů jednotlivých částí generátoru testovaných na oblasti Na Homoli se stejnou parametrizací.

Z uvedeného testování můžeme vyvodit, že nejvíc času zabírá zpracování terénu. Protože jednotlivé části vyžadují různé typy polygonizace těchto prostorových dat, je nutné jich udělat více (součástí tvorby geometrie komunikací je také jedna polygonizace terénu). Následně je podsíť zahrnující tvorbu tenzorových polí, generovaný graf ulic a geometrie komunikací. Druhým nejnáročnějším procesem je tvorba samotných budov. Dále je to soubor zpracování polygonu terénu a jeho projekce na rovinu XZ. Překvapivě subdivize na jednotlivé bloky i s možností shlukování parcel trvá pouze 2.5% celkového času.

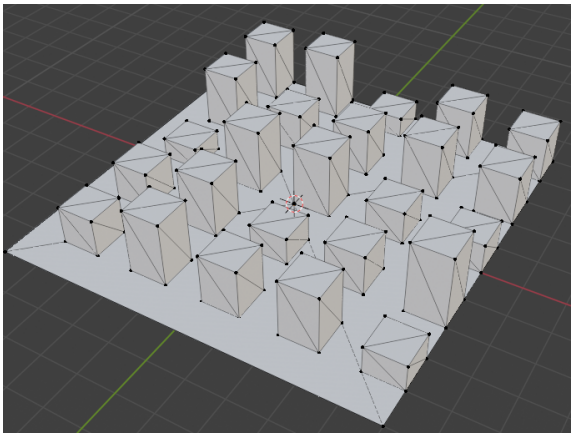
4.1.4 Možná zlepšení

Původně byl vývoj aplikace zaměřen na rozmístění budov. Bylo odhaleno, že urbanistický návrh lokalit se vyvíjí primárně podle uliční infrastruktury. Z tohoto poznatku vyplývá, že chování generátoru z největší části závisí právě na tom, jakým

způsobem se generuje graf uliční sítě a jaká je jeho kvalita. Zároveň graf ulic nepodléhá žádnému souboru pravidel pro jeho tvorbu, až na jeho definici tenzorovým polem a parametry trasovacích algoritmů. Při vývoji generátoru byly však nalezeny hraniční případy, pomocí kterých byly vzniklé chyby identifikovány a algoritmy byly upraveny tak, aby alespoň částečně opravily chyby bez zásahu uživatele.

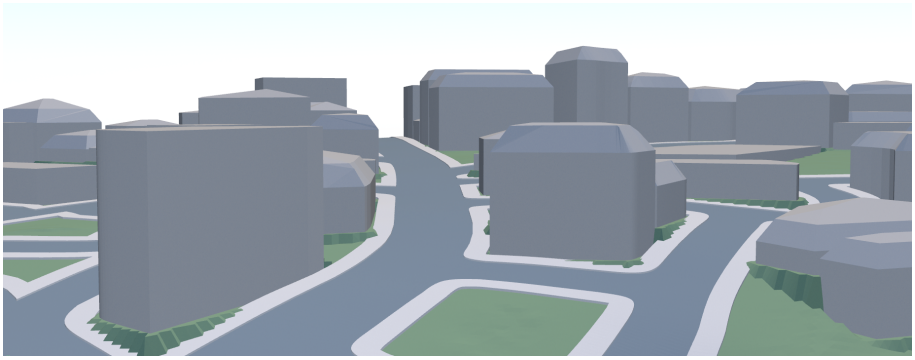
Vliv umělé inteligence

Generativní algoritmy poháněné umělou inteligencí by mohly mít na generátor pozitivní vliv. Protože Houdini umí importovat nejrůznější data, byly prozkoumány způsoby, jak donutit generativní algoritmy s umělou inteligencí k dosažení podobných výsledků. Pro ChatGPT 4o bylo obtížné takový soubor dat vytvořit i s dotazem k vytvoření části podobné Madridu s definovanou šíří silnic a podobných parametrů. Výstupem byla geometrie na obrázku 4.10.



Obrázek 4.10: Geometrie urbanistického návrhu vytvořená pomocí ChatGPT.

Nástroje umělé inteligence spojené se stylizací předčí výstupy čistě generativních algoritmů. V momentě, kdy by stylizace byla dostatečně realistická a opravdu dodržovala veškerá omezení dotazu, byly by tyto nástroje skvělým rozšířením pro prezentování návrhu veřejnosti a komerčním účelům. Na obr. 4.12 a 4.13 například chybí silnice a posuny budov jsou značné. Dotazem pro stylizaci byl obrázek 4.11.

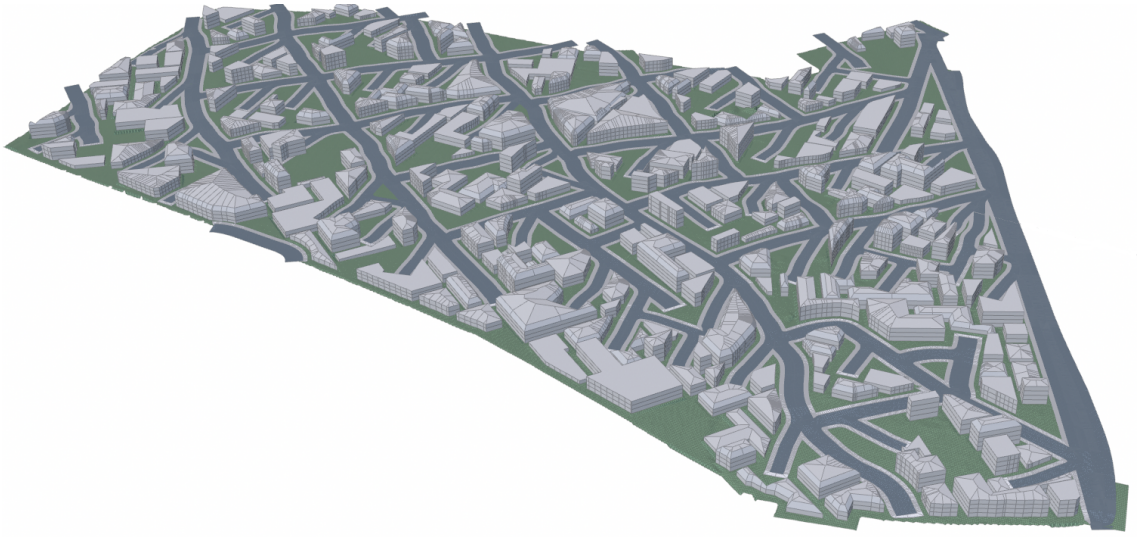


Obrázek 4.11: Vygenerovaný detail použitý jako dotaz pro algoritmus s umělou inteligencí.

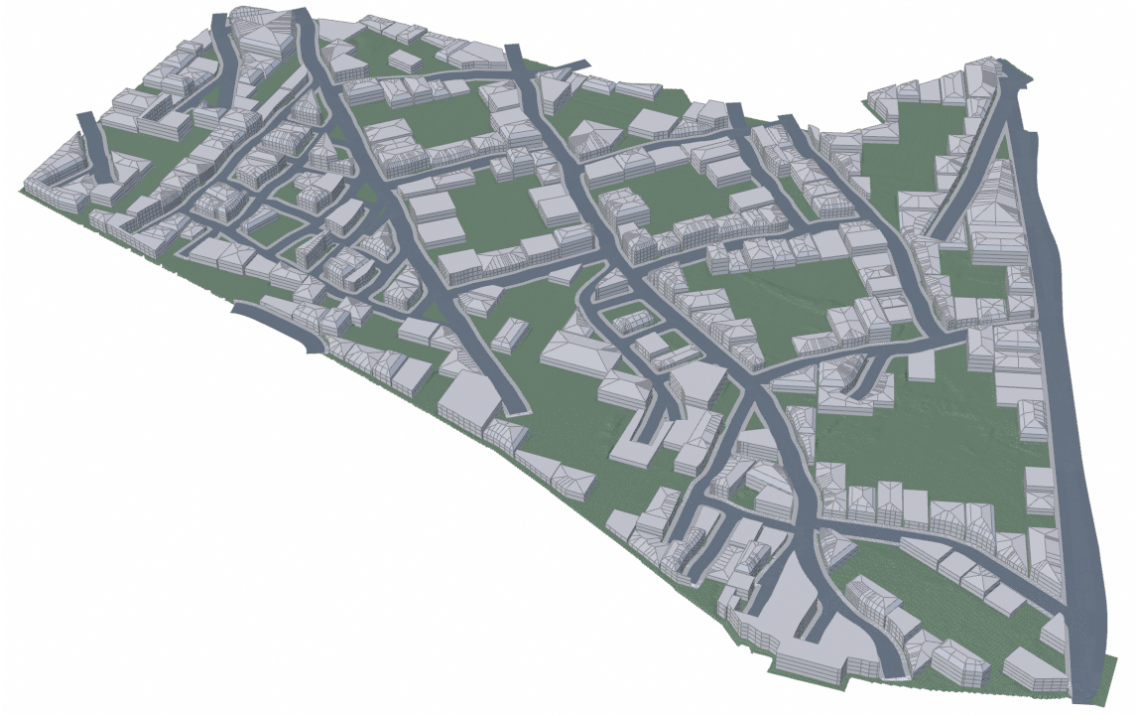
Obrázek 4.12: Stylizace obrázku 4.11, realisticky, zdroj *Dzine.ai*.

Obrázek 4.13: Jiná stylizace obrázku 4.11, realisticky, zdroj *Dzine.ai*.

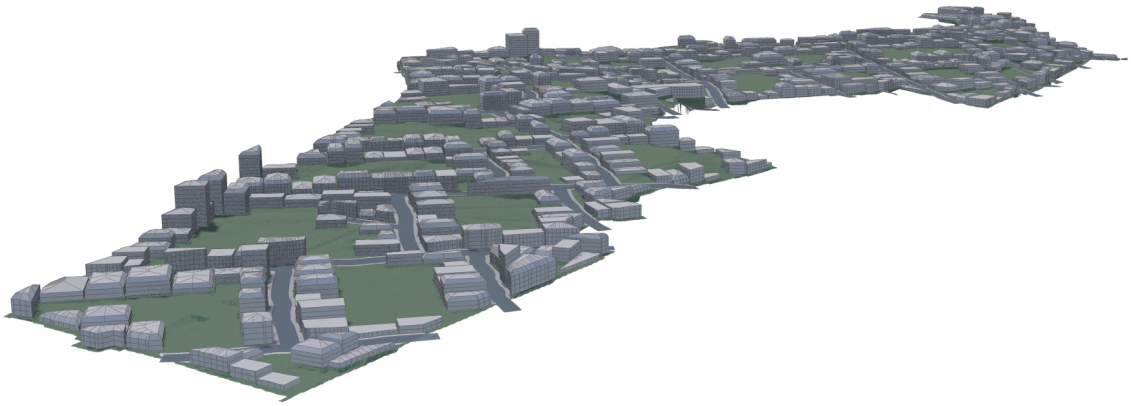
Obrázek 4.14: Stylizovaný obrázek 4.11, umělecky, zdroj *Dzine.ai*.



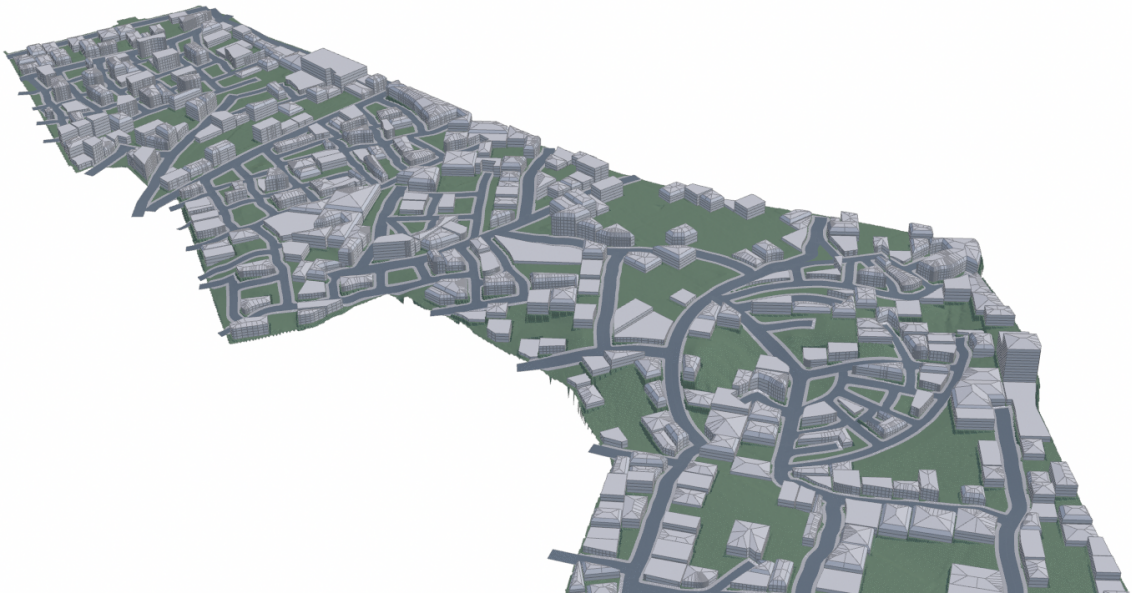
Obrázek 4.15: Realizovatelný návrh zástavby Dolních Počernic - jih s existující cestou na pravé straně. Poměrem zástavby budov odpovídá 44.25% a poměrem zástavby silnic 19.17% vůči ploše terénu. Celková délka komunikací je 10.8km.



Obrázek 4.16: Realizovatelný návrh zástavby Dolních Počernic - jih s těsnou zástavbou a delšími separačními vzdálenostmi hlavních ulic a řidší sítí vedlejších komunikací.



Obrázek 4.17: Realizovatelný návrh zástavby lokality Velkého Háje bez importovaných komunikací, s těsnou zástavbou, terén lokality je rozmanitější a propisuje se do vývoje komunikací.



Obrázek 4.18: Část realizovatelného návrhu zástavby lokality Velkého Háje s importem komunikací, s těsnou zástavbou a menšími bloky vedlejších ulic, terén lokality je rozmanitější a propisuje se do vývoje komunikací.

Závěr

Diplomová práce seznámila čtenáře s principy procedurálního generování, které budou v moderním světě hrát čím dál důležitější roli v pravděpodobné kombinaci s umělou inteligencí, protože se jeho aplikace stále rozšiřují. Byly prozkoumány některé konkrétní techniky, které slouží k procedurálnímu generování dat a bylo uvedeno i jejich použití. Ve druhé kapitole se čtenář seznámil s poskytnutými daty, geoinformačními systémy, pravidly výstavby podle Metropolitního plánu města Prahy, se softwarem SideFX Houdini a souvisejícími tématy této práce.

V realizační části jsou vysvětleny postupy, kterými generátor pracuje s dostupnými daty, s tvorbou uličních grafů a geometrie komunikací, s tvorbou povrchů zastavitelných ploch a generováním geometrie budov v Houdini. Všechny tyto postupy podléhají obšírné parametrizaci a uživatel má kvalitní a rozsáhlou kontrolu nad výsledky. Těmi jsou urbanistické scény, které dodržují nejen základní principy efektivního 3D modelování, ale i soubor pravidel v kontextu urbanistického plánování. Zároveň existují v rámci generátoru nástroje tvorby základních statistik nad konkrétními vizualizacemi.

Výsledky generátoru během vývoje byly konzultovány se specialisty z Institutu plánování a rozvoje hlavního města Prahy, kteří reagovali na chyby generátoru a prezentovali možná referenční řešení. Na základě jejich podnětů vyžadujících automatizaci s uživatelskou kontrolou byla navržena část tvorby silnic pomocí tenzorových polí. Dále byla na základě podnětů upravena vizualizace a snížen detail budov. Důležitým faktorem pro urbanistické architekty je fakt, že za použití generátoru a při správné parametrizaci důkladně prostudované oblasti jsou schopni vytvořit spoustu různých variant splňujících stejná kritéria v poměrně krátkém čase relativně k době, kterou by nutně museli strávit při zhotovování návrhů zastavitelnosti. Při použití generátoru se jedná řádově o desítky uspořených hodin, finančních prostředků a lidských zdrojů. Zároveň je vyžadováno, aby specialisté připravili kvalitní vstupy, hodnoty parametrů a ostatní související podklady pro zadání do generátoru.

Při analýze i realizaci vyšlo najevo, že uliční sítě mají zásadní vliv na infrastrukturu města i strukturu jednotlivých útvarů. Bohužel v této části se vyskytuje pár nedostatků při hraničních případech, na jejichž základě se chyby promítají do ostatních algoritmů generátoru. Nemělo by se ale stát, aby generátor při smysluplné parametrizaci nebyl schopný data zpracovat.

Kvalita výsledků je dobrá a porovnatelná s jinými vizualizacemi, ale je možné, že generátoru chybí rozmanitost tvarů, jako například střech, struktur v komunikacích a jiných důležitých faktorech urbanistického plánování.

Lze posoudit, že práce byla úspěšná v seznámení se s tématem, a to především s technikami vhodnými pro tvorbu geometrie vhodné pro urbanistické průzkumy. Seznámení s prostředím Houdini je dostatečné během analýzy, ale v realizaci jsou spíše vysvětlené principy oproti exaktní implementaci. To je zdůvodněno obtížným

promítnutím vizuálního programování do textu. Práce neseznamuje čtenáře s hlubšími technikami souvisejících témat, jako jsou procedurální texturování, mapování a převod do jiných formátů použitelných například v geoinformatice.

Bibliografie

1. PARBERRY, Ian. Designer worlds: Procedural generation of infinite terrain from real-world elevation data. *Journal of Computer Graphics Techniques*. 2014, roč. 3, č. 1.
2. EBERT, David S; MUSGRAVE, F Kenton; PEACHEY, Darwyn; PERLIN, Ken; HART, John C; WORLEY, Steven. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.
3. ČMOLÍK. *Procedural modeling of 2D and 3D objects*. 2020.
4. SHORT, Tanya; ADAMS, Tarn. *Procedural generation in game design*. CRC Press, 2017.
5. DEMIR, Ilke; ALIAGA, Daniel G; BENES, Bedrich. Proceduralization for editing 3d architectural models. In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, s. 194–202.
6. MANDELBROT, Benoit B; MANDELBROT, Benoit B. *The fractal geometry of nature*. Sv. 1. WH freeman New York, 1982.
7. KELLY, George; MCCABE, Hugh. A survey of procedural techniques for city generation. *The ITB Journal*. 2006, roč. 7, č. 2, s. 5.
8. WIKIPEDIA CONTRIBUTORS. *Shape grammar — Wikipedia, The Free Encyclopedia*. 2022. Dostupné také z: `https://en.wikipedia.org/w/index.php?title=Shape_grammar&oldid=1073156257`. [Online; accessed 20-January-2024].
9. SCHWARZ, Michael; MÜLLER, Pascal. Advanced procedural modeling of architecture. *ACM Transactions on Graphics (TOG)*. 2015, roč. 34, č. 4, s. 1–12.
10. HUANG, Jingyuan; PYTEL, Alex; ZHANG, Cherry; MANN, Stephen; FOURQUET, Elodie; HAHN, Marshall; KINNEAR, Kate; LAM, Michael; COWAN, William. An evaluation of shape/split grammars for architecture. *Waterloo*. 2009.
11. NÁRODNÍ PAMÁTKOVÝ ÚSTAV. *Slovníček pojmů*. 2024. Dostupné také z: `https://pagis.npu.cz/wordpress/slovnicek/`. Přístupné online k 27. prosinci 2024.
12. KEJVALOVÁ, Jana. *Procedural Model Generation from Real Maps*. 2023. Dipl. pr. ČVUT FEL.
13. WIKIPEDIE. *World Geodetic System — Wikipedie: Otevřená encyklopedie*. 2023. Dostupné také z: `https://cs.wikipedia.org/w/index.php?title=World_Geodetic_System&oldid=23085426`. [Online; navštíveno 20. 01. 2024].

14. INSTITUT PLÁNOVÁNÍ A ROZVOJE HL. M. PRAHY. *Metropolitní plán Prahy*. 2025. Dostupné také z: https://plan.praha.eu/. Přístup na web dne 5. ledna 2025.

15. PRAHY, Institut plánování a rozvoje hl. m. *Sídlení struktura a urbanismus*. n.d. Dostupné také z: https://iprpraha.cz/uploads/assets/uap_pdf/2_03_sidelni_struktura_a_urbanismus.pdf. Přístupné online na IPR Praha.

16. INSTITUT PLÁNOVÁNÍ A ROZVOJE HL. M. PRAHY (IPR PRAHA). *Pražské stavební předpisy - úplné znění (příloha 1 a 2)*. 2021. Dostupné také z: https://iprpraha.cz/uploads/assets/dokumenty/otpp/psp/01_psp_uplne-zneni_text_priloha1-a-2.pdf. Accessed: December 30, 2024.

17. FOTHERINGHAM, Avonlea; FABRICIUS, Taisha. *What's New in ArcGIS CityEngine (July 2024)*. 2024. Dostupné také z: https://www.esri.com/arcgis-blog/products/city-engine/3d-gis/whats-new-in-arcgis-cityengine-july-2024/. Accessed: 2024-12-27.

18. *Tutorial 1: Essential Skills*. 2023. Dostupné také z: https://doc.arcgis.com/en/cityengine/2023.0/tutorials/tutorial-1-essential-skills.htm. Accessed: 2025-01-02.

19. SIDEFX. *Houdini Nodes Documentation*. 2024. Dostupné také z: https://www.sidefx.com/docs/houdini/nodes/index.html. Přístupné online k 27. prosinci 2024.

20. CHEN, Guoning; ESCH, Gregory; WONKA, Peter; MÜLLER, Pascal; ZHANG, Eugene. *ACM SIGGRAPH 2008 papers*. Interactive procedural street modeling. 2008.

21. GENISEASE. *Create Cross Roads In Houdini*. 2023. Dostupné také z: https://youtu.be/mRzGmaHEJiM?si=Ie1iuvVAG7h_JjvU. YouTube video, accessed January 3, 2025.

22. WIKIPEDIA CONTRIBUTORS. *Nonzero-rule — Wikipedia, The Free Encyclopedia*. 2025. Dostupné také z: https://en.wikipedia.org/wiki/Nonzero-rule. Accessed January 3, 2025.

23. PILAŘ ARCHITEKTI, Kuba a. *Smíchov City - Sever*. 2023. Dostupné také z: https://iprpraha.cz/assets/files/files/0e592378ecd96b9694986ea43236787b.pdf. Přístupné: 7. ledna 2025.

Přílohy

A Seznam použitých uzlů

Seznam definovaný tabulkami 1 a 2 je výčtem použitých uzlů v rámci jednoho algoritmického zpracování geometrie. Uzly byly spočítány rekurzivním zanořením následujícím python skriptem. Skript počítá i uzly uvnitř operátorů, které autor nevytvořil.

Listing 1: Python skript pro spočtení všech uzlů sítě

```
import collections
import hou

geo_network = hou.node("/obj/Ruzyne_HF")

all_nodes = geo_network.allSubChildren()

node_types = collections.Counter([node.type().name()
                                  for node in all_nodes])

for node_type, count in node_types.items():
    print(f"{node_type}: {count}")
```

Typ uzlu	Četnost	Typ uzlu	Četnost
heightfield	1	volume	2
xform	51	volumevisualization	2
merge	133	switch	295
output	327	heightfield_file	1
cop2net	1	file	1
mono	1	bright	1
fetch	1	primitive	7
name	9	blast	254
volumemix	23	null	394
heightfield_layer	10	attribwrangle	1098
attribwranglecore	669	block_end	134
block_begin	249	heightfield_copylayer	19
box	7	ray	15
heightfield_maskbyobject	5	subnet	80
labs::road_generator::1.2	3	visualize	6
object_merge	31	fuse::2.0	216
groupcreate	252	sphere	5
intersectionanalysis	11	intersectionstitch	7
color	67	labs::extract_borders	11
polyextrude::2.0	14	heightfield_maskbyfeature	4
volumewrangle	8	volumewranglecore	2
volumeblur	12	volumeanalysis	4
heightfield_maskbyocclusion	2	heightfield_blur	7
opencl	2	switchif	337
convertheightfield	4	extrudevolume	4
convert	22	vdbfrompolygons	2
attribfromvolume	2	attribdelete	214
attribute	180	convertvdb	2
vdbresample	2	clean	8
delete	5	facet	59
add	154	fuse	7
reverse	28	polydoctor	23
groupdelete	160	groupexpression	24
polyexpand2d	19	quadremesh	1
connectivity	33	compile_end	54
compile_begin	75	divide	9
ends	7	vertexsplit	4
sort	154	resample	39
labs::merge_splines::1.0	12	grouppromote	73
split	40	polypath	63
rewire	30	groupcopy	62
attribinterpolate	120	grouptransfer	79
attribcopy	33	convertline	80
measure	30	polycut	19
splitpoints	35	attribpromote	144

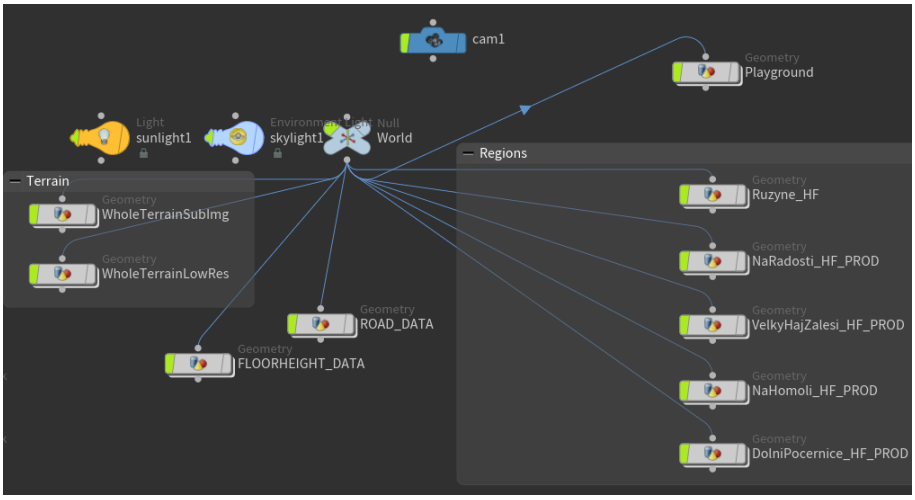
Tabulka 1: Přehled typů uzlů a jejich četností (dva sloupce).

Typ uzlu	Četnost	Typ uzlu	Četnost
tube	21	copytopoints::2.0	50
attribswap	28	circle	7
edgetransport	3	curvesect	1
scatter::2.0	2	distancefromgeometry	1
attribremap	1	unpack	3
windingnumber	12	measure::2.0	33
groupinvert	13	carve	12
attribcreate::2.0	21	sweep::2.0	7
skin	1	Labs::polyscalpel::1.0	2
pack	4	polyframe	4
line	7	normal	64
edgedivide	1	dissolve::2.0	16
boolean::2.0	9	groupexpand	1
findshortestpath	3	polysplit::2.0	2
groupcombine	29	peak	1
edgecusp	2	grouprename	9
explodedview::2.0	2	attribtransfer	1
heightfield_project	3	curve::2.0	2
stash	3	singlecurveoperation	2
edgeequalize	1	edgestraighten	1
smooth::2.0	1	circlefromedges	1
enumerate	4	attribexpression	12
softxform	1	orientalongcurve	1
labs::hf_insert_mask	3	labs::hf_combine_masks	3
heightfield_remap	9	grouprange	1
heightfield_flatten	1	heightfield_layerclear	2
heightfield_quickshade	1	polyfill	9
hole	8	shrinkwrap::2.0	1
labs::lot_subdivision::2.0	3	cluster	1
triangulate2d::3.0	1	labs::remove_inside_faces	2
labs::dissolve_flat_edges::1.0	3	groupfromattribboundary	1
extractcentroid	4	bound	2

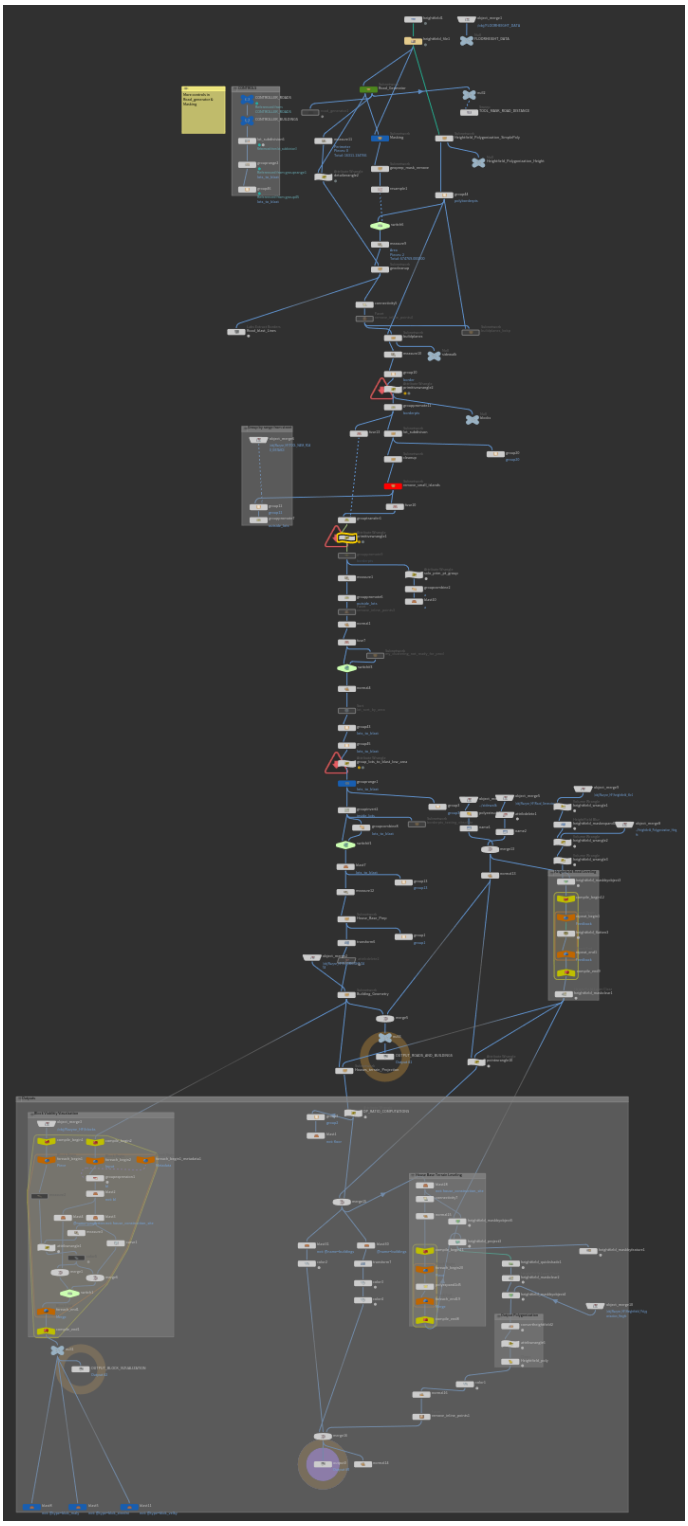
Tabulka 2: Přehled typů uzlů a jejich četností (dva sloupce).

B Obrázky sítí geometrie

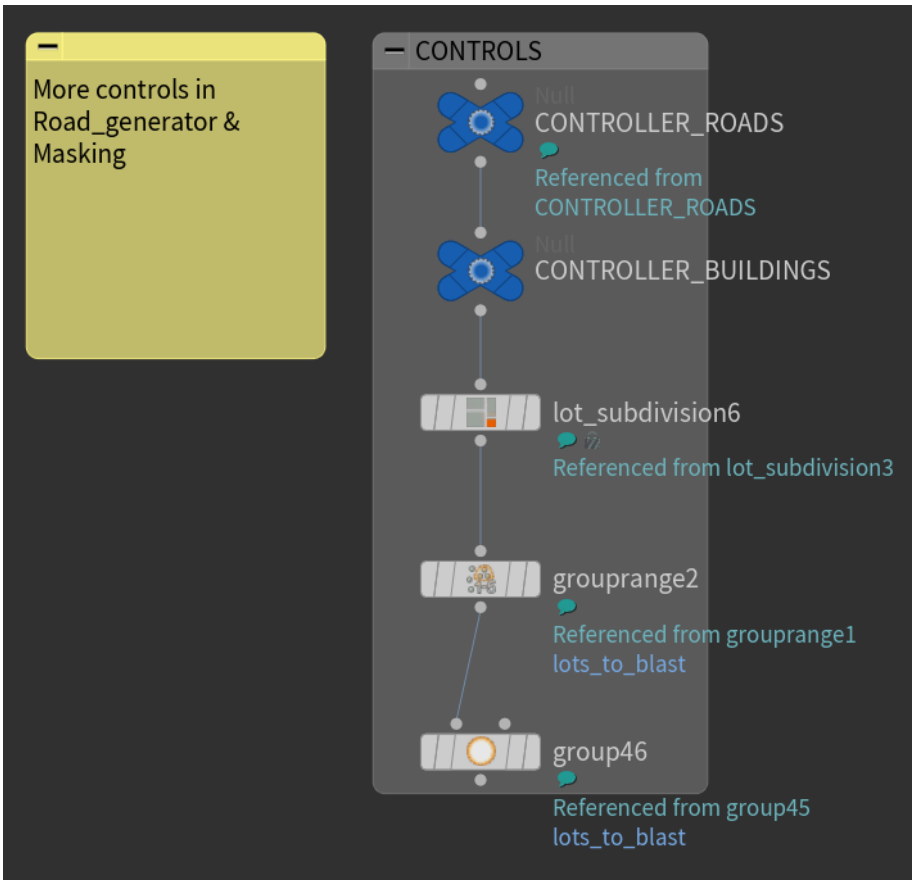
Kvůli praktičnosti popisu algoritmů v rámci diplomové práce bylo opomenuto znázornění postupů v prostředí Houdini. Postupy jsou v této příloze znázorněny pomocí vizuálního programování. Na obrázku 19 je objektová síť, na obrázku 20 je hlavní SOP síť geometrie. Na obrázcích 21 - 27 je znázorněno vytvořené uživatelské prostředí pro povrchní ovládání algoritmů a specifikaci nutných parametrů. Obrázky 28 - 35 ukazují jednotlivé podsítě důležitých algoritmů.



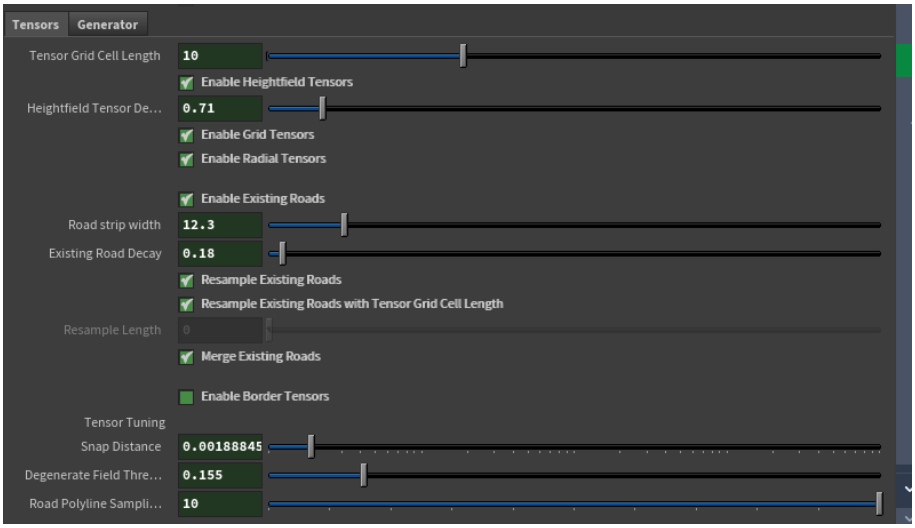
Obrázek 19: Objekty v síti.



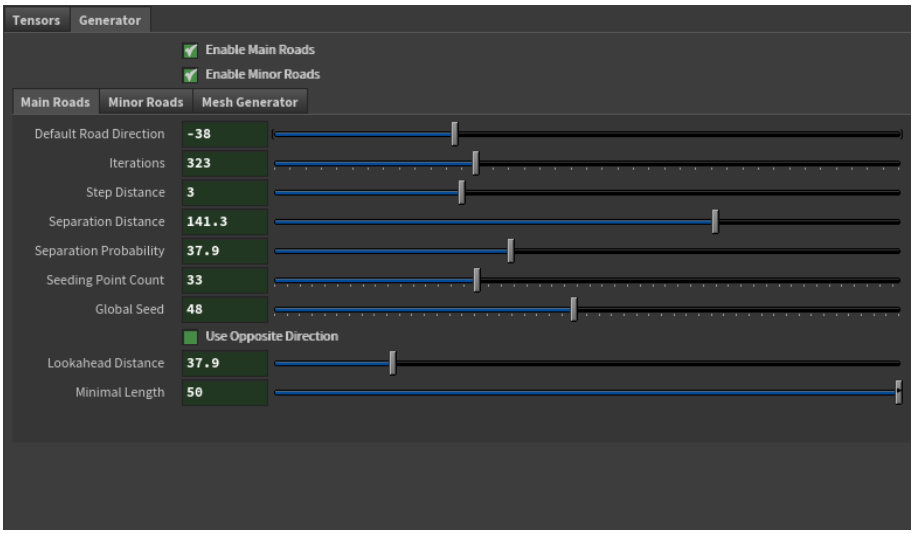
Obrázek 20: Hlavní síť uzlů.



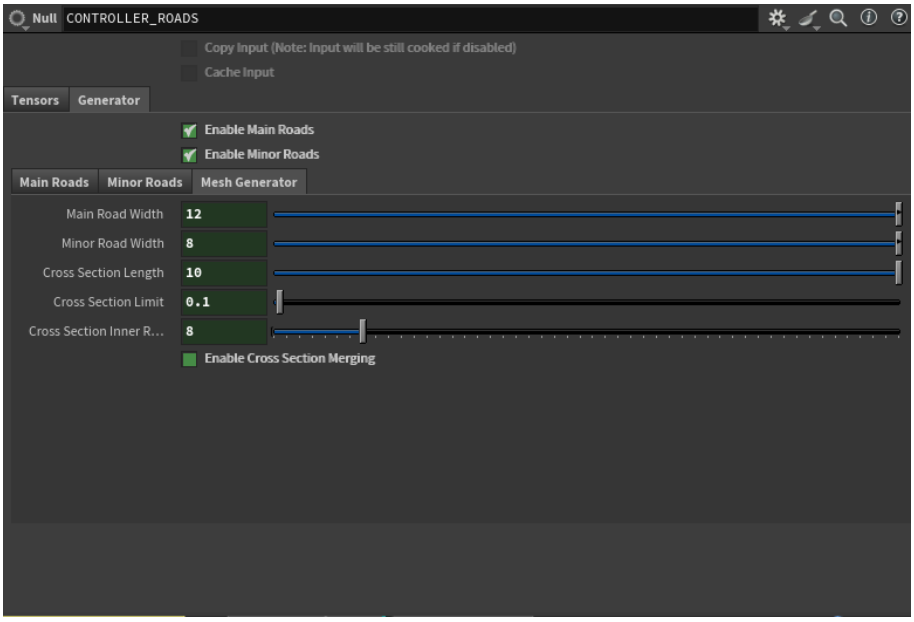
Obrázek 21: Hlavní ovládací panel.



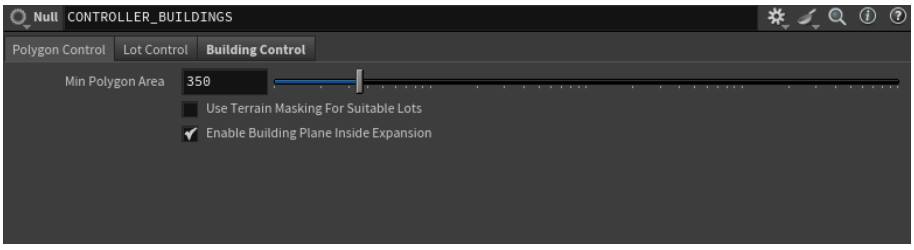
Obrázek 22: Ovládání tvorby tenzorů.



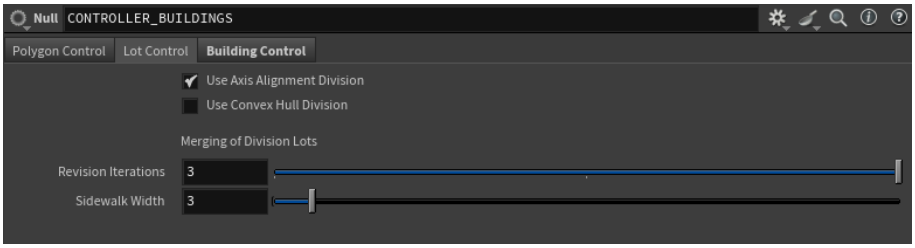
Obrázek 23: Ovládání hlavních silnic.



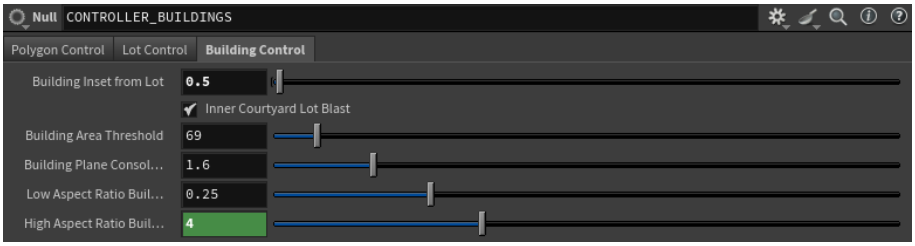
Obrázek 24: Ovládání vedlejších silnic.



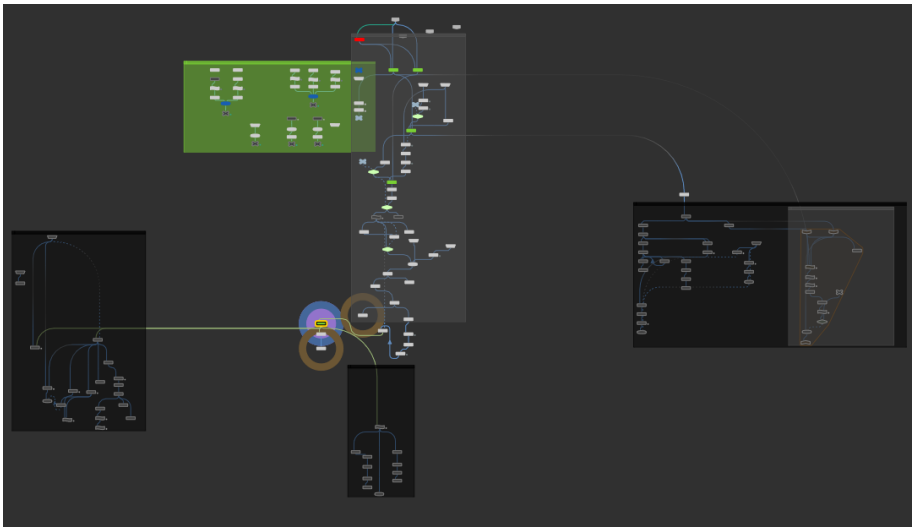
Obrázek 25: Ovladač chování polygonu.



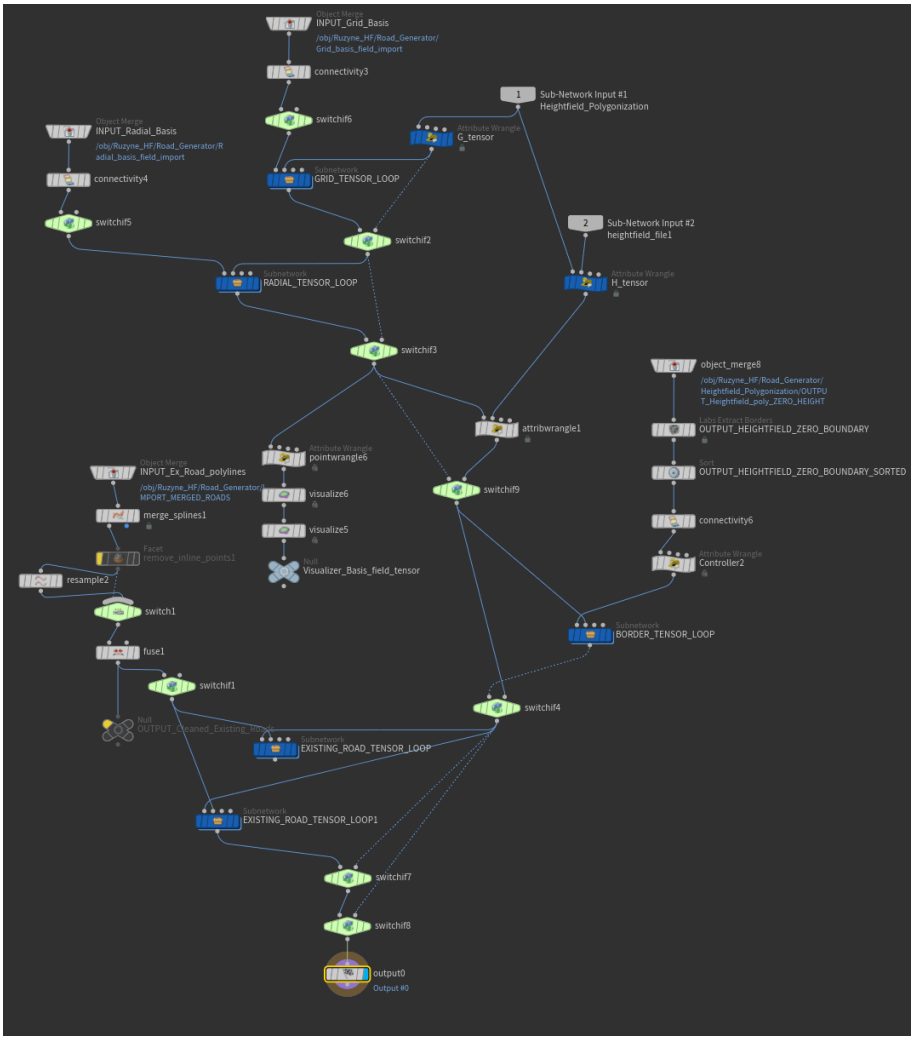
Obrázek 26: Ovladač chování bloků.



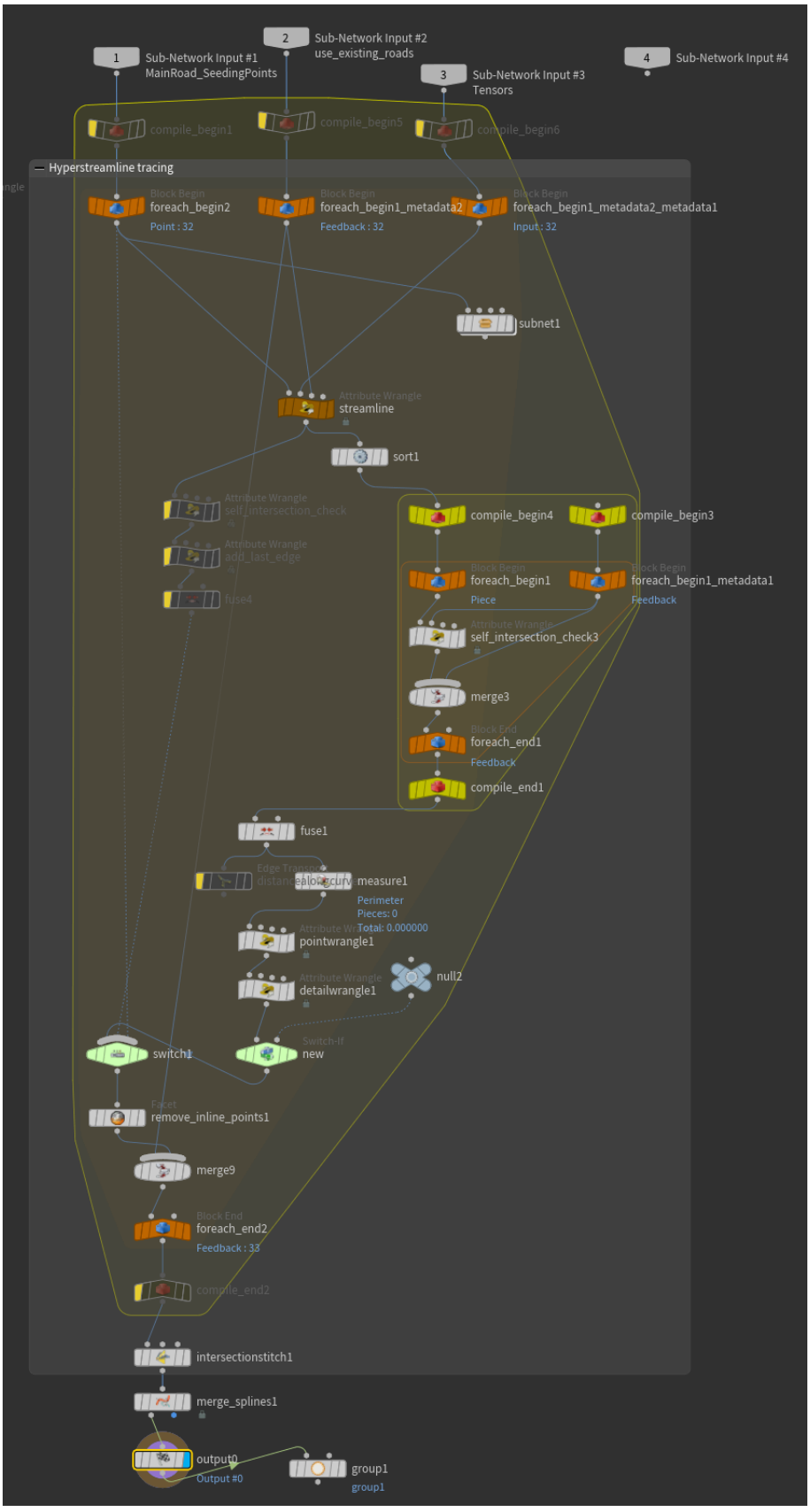
Obrázek 27: Ovladač chování parcel a budov.



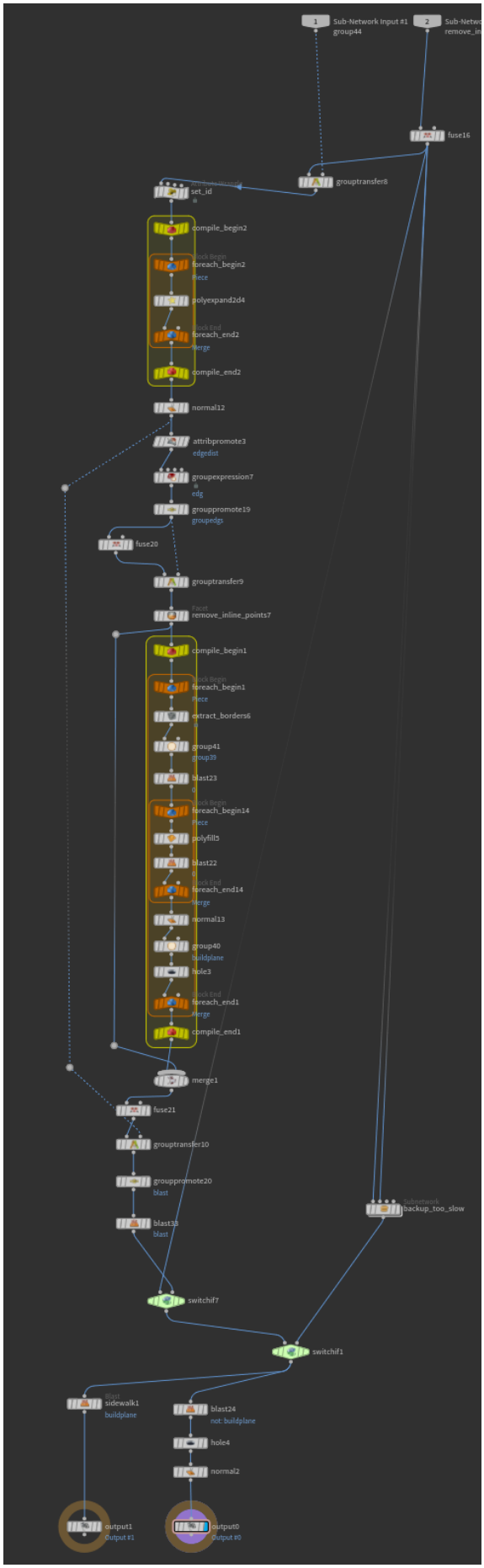
Obrázek 28: Síť generátoru tenzorových polí a grafu ulic.



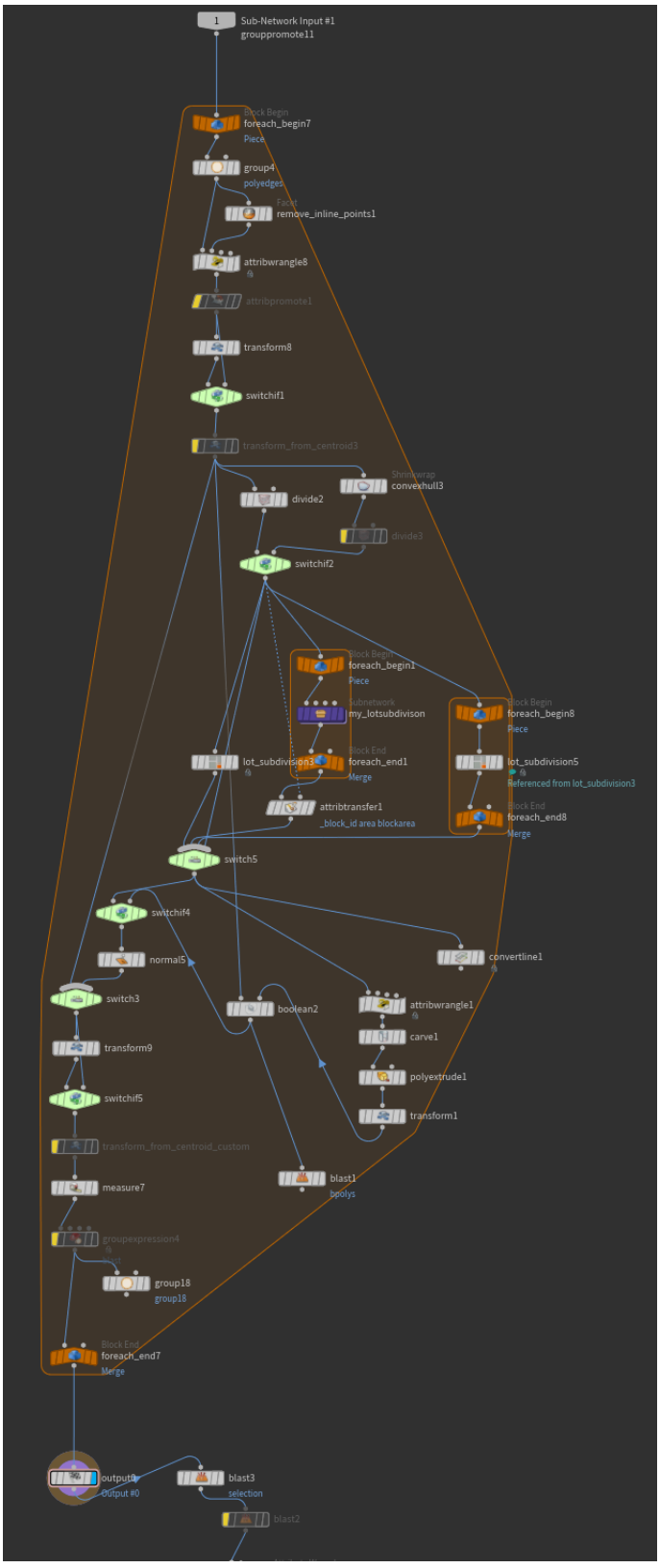
Obrázek 29: Síť s přepínači na ovládání tenzorového pole.



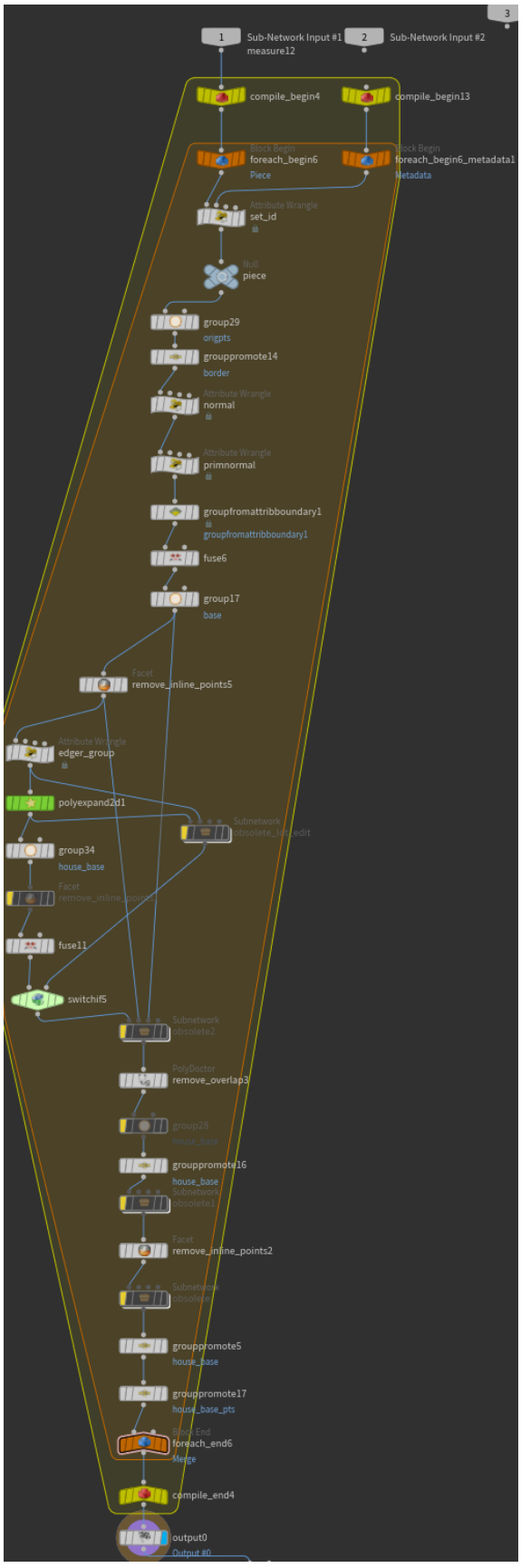
Obrázek 30: Trasovací cyklus.



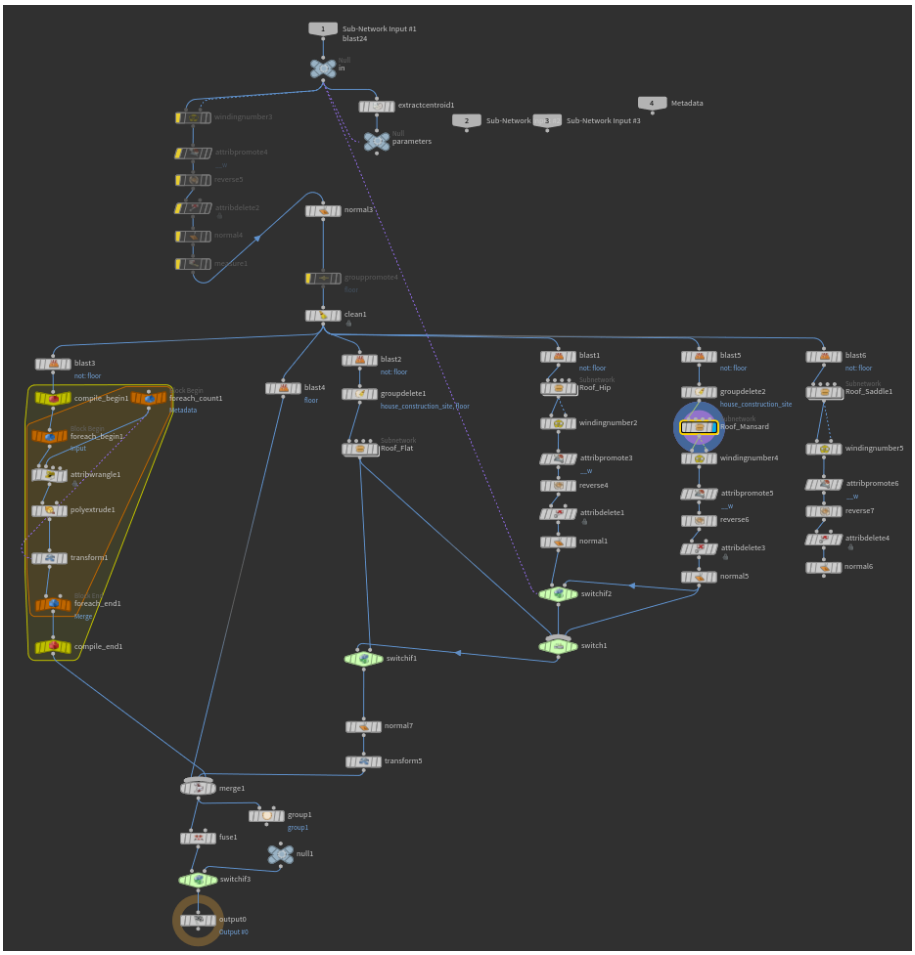
Obrázek 31: Příprava geometrie polygonu s odebraným průmětem geometrie ulic.



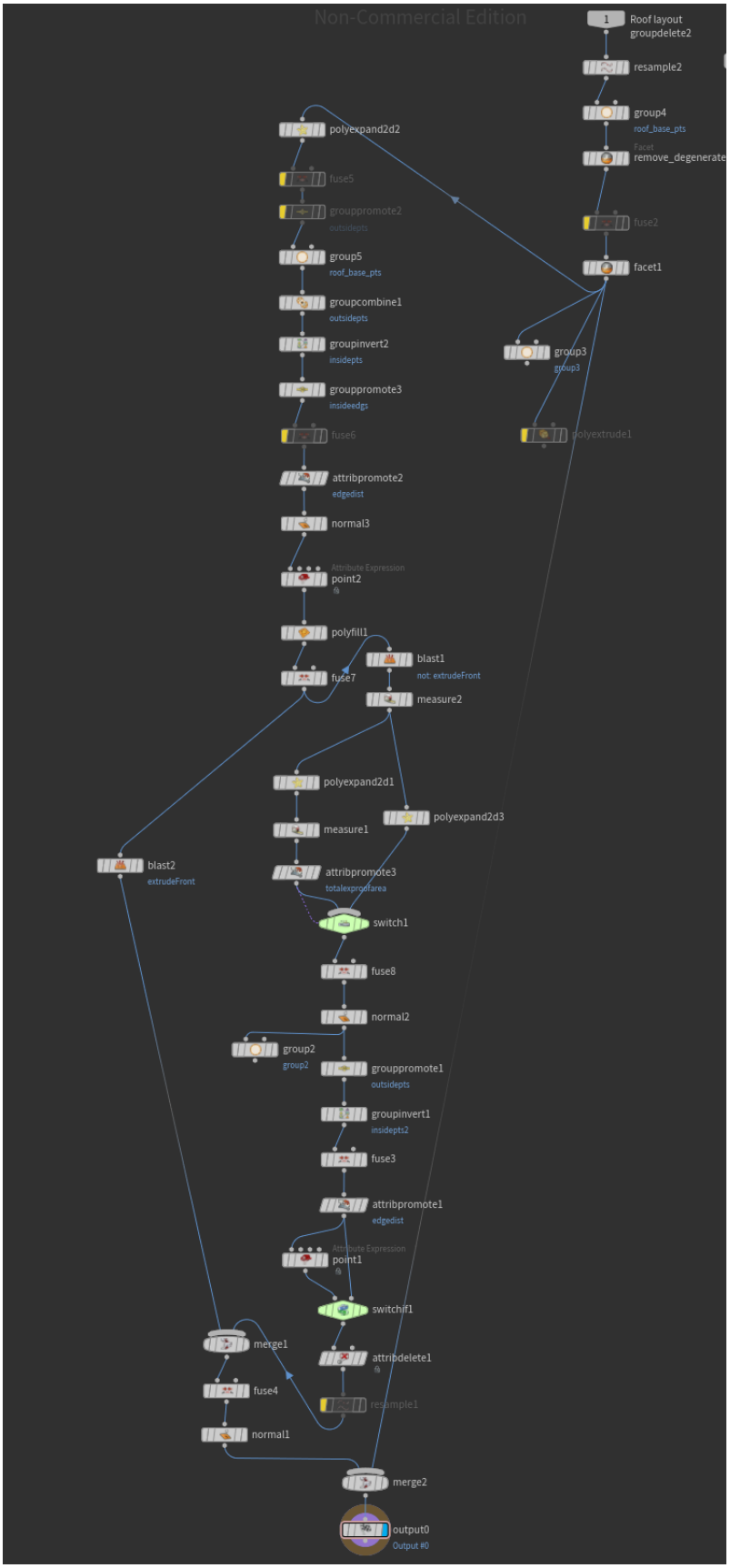
Obrázek 32: Subdivize bloků s vyrovnáním k ose.



Obrázek 33: Příprava základů budovy.



Obrázek 34: Logika geometrie budovy.



Obrázek 35: Logika geometrie střech.