Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Generátor měst pro Blender

2016

Jan Hamerník

Vedoucí práce: RNDr. Eduard Bartl, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Jan Hamerník
Název práce:	Generátor měst pro Blender
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2016
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	RNDr. Eduard Bartl, Ph.D.
Počet stran:	76
Přílohy:	1 CD
Jazyk práce:	český

Bibliograhic info

Author:	Jan Hamerník
Title:	Blender City Generator
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2016
Study field:	Applied Computer Science, full-time form
Supervisor:	RNDr. Eduard Bartl, Ph.D.
Page count:	76
Supplements:	1 CD
Thesis language:	Czech

Anotace

Tato bakalářská práce obsahuje základní informace o 3D grafickém editoru Blender. Dále se text věnuje detailnímu popisu ovládání vytvořeného doplňku, a popisu algoritmů použitých pro generování náhodných 3D modelů budov, částí cest až po konečné vytvoření celého města.

Synopsis

This thesis contains basic information about 3D graphical editor Blender. The text then offers a detailed description of how to use the created addon, and a description of the algorithms used for generating random 3D models of buildings, road parts and finally the entire city.

Klíčová slova: Blender; doplněk; generátor měst; náhodná budova

Keywords: Blender; addon; city generator; random building

Děkuji vedoucímu práce Mgr. Eduardu Bartlovi, Ph.D. za cenné rady a přátelský přístup. Dále děkuji všem, co mě podporovali během studií a umožnili tak dokončení této práce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Schopnost vytvářet 3D modely měst může být uplatněna v mnoha odvětvích, počínaje domácí tvorbou až po herní či filmový průmysl. Jelikož města zpravidla obsahují mnoho budov a „příslušenství“ (například pouliční osvětlení či zaparkovaná auta), vytvářet je „ručně“ je zdlouhavá a neefektivní práce. Odpovědí na tento problém je právě generátor měst, který dokáže takové město vytvořit daleko rychleji a s náhodnými parametry – to je také tématem této práce.

Následující text lze pomyslně rozdělit na uživatelskou a programátorskou část. V kapitole 2 popisuji důvod, proč jsem se rozhodl tento doplněk vytvořit, a také zmiňuji svůj první (neúspěšný) přístup. Kapitola 3 stručně představuje 3D grafický editor Blender, do kterého se tento doplněk instaluje. V kapitole 4 se poté věnuji potřebným základům práce s Blenderem, včetně instalace tohoto doplňku. Následuje kapitola 5, která podrobně popisuje ovládání a veškeré možnosti tohoto doplňku.

Kapitola 6 představuje programátorskou část textu. V ní popisuji nejdůležitější funkce potřebné pro generování měst, jako je například funkce pro generování budov, či náhodných cest.

2 Motivace pro vznik generátoru měst

Už přes jedenáct let se zabývám 2D a 3D grafikou. Začínal jsem tedy zhruba ve dvanácti letech, kdy jsem měl pouze mizivé znalosti o programování. Po pár letech v 3D grafice jsem však začal mít větší ambice a zkoušel větší projekty, včetně vymodelování města. Pokusil jsem se o to dokonce několikrát, pokaždé jsem od toho však po pár dnech upustil, protože dělat něco takového ručně je velice neefektivní. Také se těžko simuluje náhodnost budov a ulic, protože se vždy jedná o vědomou volbu.

Začal jsem se tedy zajímat o to, jak „naučit počítač“, aby takové věci tvořil za mě. Nutno podotknout, že ještě před čtyřmi roky jsem o programování neměl o moc větší znalosti, než jsem měl v dětství. Svůj úplně první kód jsem napsal teprve v prvním ročníku zde, na Univerzitě Palackého v Olomouci (typické „Hello World!“).

Hlavní motivací tedy bylo vyzkoušet si získané znalosti v programování na něčem praktickém, co bych doopravdy použil při své práci s 3D grafikou. Jelikož jsem v několika svých předchozích projektech potřeboval 3D model města, byl generátor měst přirozenou volbou.

Další (neméně důležitou) motivací byla také možnost naučit se programovat v jazyku Python, který Blender používá. Jak jsem již zmínil dříve, s Blenderem pracuji velice často, a právě programování v Pythonu je velice mocným nástrojem.

Díky této práci jsem se tedy naučil mnoho nových věci, které prakticky využiji. **Nejedná se samozřejmě o pouhé zvládnutí syntaxe Pythonu**, ale převážně o použití Pythonu pro přístup k datům Blenderu, jako jsou právě 3D modely – ať už jejich transformace, vytváření nových modelů, nebo importování modelů. Díky tomuto experimentování jsem se naučil také věci, které použiji mimo tuto práci – například práci s „drivery“ (pokročilejší funkce Blenderu).

2.1 První metoda

Abych příliš nepředbíhal, pouze zjednodušeně nastíním, jak tento generátor tvoří náhodné budovy: úplný základ jsou 3D modely menších sekcí budov (například část stěny s okny, či vchodové dveře) a pomocí Pythonu se z těchto částí poskládá budova podle zadaných parametrů (ty jsou při generování města v jistých mezích náhodně generovány).

Původně jsem však zamýšlel vytvořit celé budovy čistě kódem. Python totiž umožňuje vytvářet nové objekty a těm definovat vrcholy, hrany a stěny. Bohužel jsem se tohoto řešení dlouho nechtěl vzdát, takže došlo ke zbytečnému zdržení, nicméně nakonec jsem dospěl k názoru, že takhle to nepůjde. Zjednodušeně proto, že se tímto způsobem prakticky nedají tvořit ani mírně složitější útvary, navíc je jejich zápis značně nepřehledný a nevypovídá nic o jejich vzhledu.

3 Blender obecně

Blender je open source 3D grafický editor (dále jen editor), který umožňuje pokročilé 3D modelování, animaci, rigování, simulace, renderování, compositing, motion tracking (snímání pohybu kamery/objektu), editaci videa a dokonce i vytváření her. Více informací o jeho možnostech je k dispozici na oficiálních webových stránkách: www.blender.org/features/.

Obrázek 1: Blender logo

Blender je vyvíjen neziskovou organizací Blender Foundation, která do letošního roku (2016) produkovala 4 krátké animované filmy:

1. Elephants Dream (2006),
2. Big Buck Bunny (2008),
3. Sintel (2010),
4. Tears of Steel (2012).

Všechny tyto filmy byly vytvořeny téměř výhradně pomocí Blenderu, vřele je tedy doporučuji všem, kteří by chtěli vidět, co je možné pomocí Blenderu vytvořit.

Následuje popis výhod a nevýhod Blenderu. Bude se jednat o mé vlastní názory posbírané za mnoho let používání tohoto editoru. Je tedy možné, že co zde popisuji jako výhodu, může někomu jinému připadat spíše jako nevýhoda a naopak.

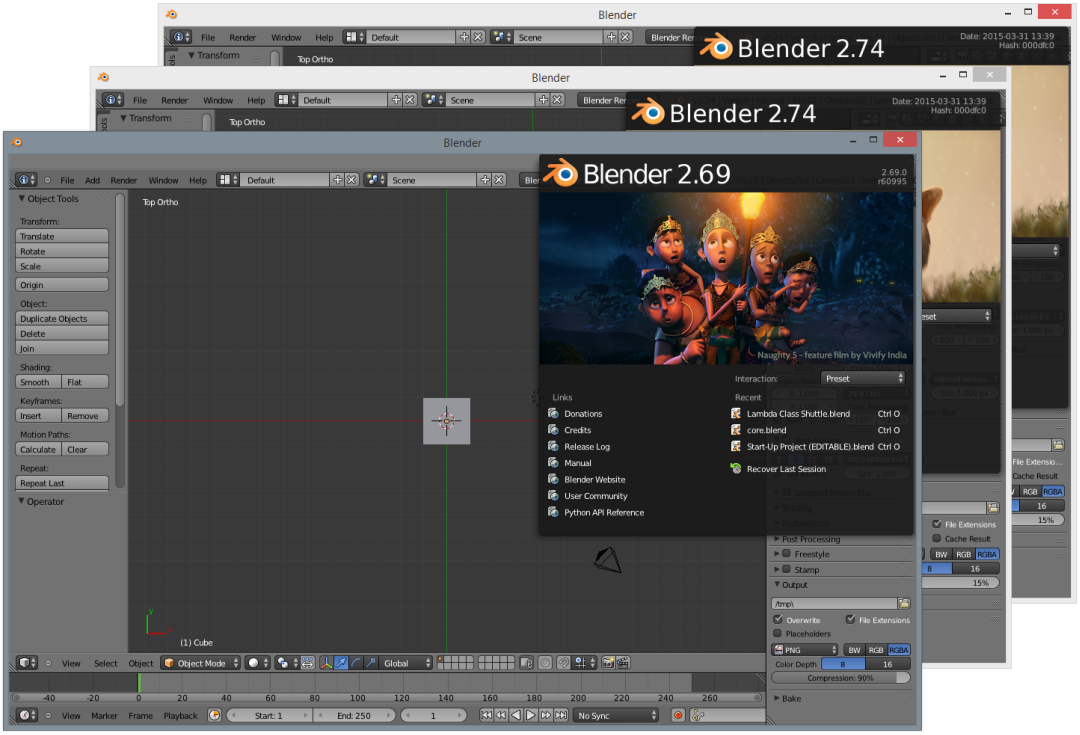
3.1 Výhody Blenderu

Asi největší výhodou Blenderu je fakt, že je k dispozici ke stažení zcela zdarma, a dokonce za jakýmkoli účelem – ať už to jsou jen drobné osobní projekty pro pobavení, nebo studijní či komerční záměry.

S trochou šikovnosti a dostatečnými zkušenostmi lze v Blenderu vytvořit působivá díla, schopná konkurovat placeným programům, jako jsou například 3Ds Max, Maya či Cinema 4D, které jsou populární v Hollywoodské filmové tvorbě.

Blender je neustále aktualizován a každoročně tak vychází minimálně jedna nová verze. Ve chvíli, kdy potřebujete použít starší verzi by u mnoha programů nastal problém, ne však u Blenderu. Na jednom počítači totiž může běžet libovolné množství verzí současně a nezávisle na sobě. Je také možné mít otevřené

libovolné množství instancí Blenderu,[1] můžete tedy současně renderovat v jednom okně, a v druhém okně pokračovat v práci. Ze zkušenosti mohu potvrdit, že se jedná o velice užitečnou funkci, hlavně ve chvíli, kdy starší projekt nefunguje v novější verzi – je tu tedy možnost nadále ho otevírat ve starší verzi, bez nutnosti pře-instalace. Všechny verze Blenderu jsou opět k dispozici na oficiálních stránkách: download.blender.org/release/.



Obrázek 2: Je možné pracovat s libovolným množstvím instancí Blenderu (i s různými verzemi) současně.

Je také vhodné zmínit, že Blender je k dispozici pro mnoho platforem, jako například GNU/Linus, Microsoft Windows a Mac OS X.

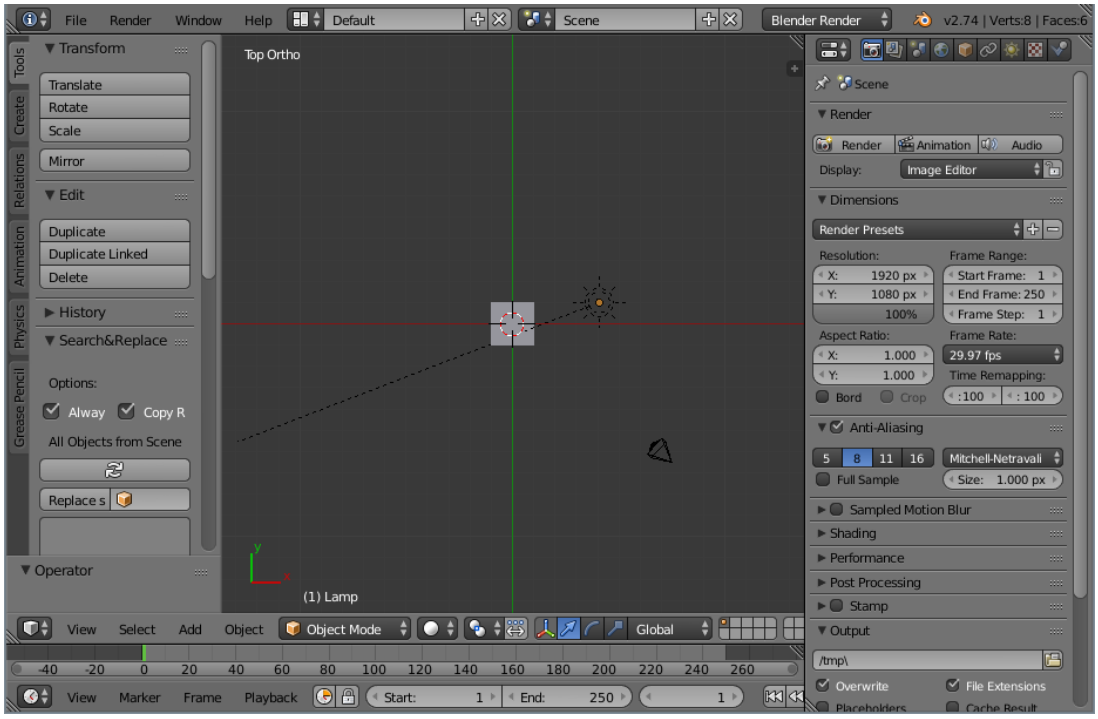
Velikou výhodou (hlavně pro programátory) je také možnost rozšiřovat Blender o specializované funkce pomocí Python skriptů – bez této možnosti by nemohla vzniknout tato práce.

[1] Je to možné na všech operačních systémech mimo Mac OS X. I zde je ovšem možnost obejít nastavení systému a spustit tak více instancí Blenderu.

3.2 Nevýhody Blenderu

Jako každý program, i Blender má samozřejmě své nevýhody, během mého několikaletého používání tohoto editoru jsem ale nenarazil na žádnou zásadní chybu. Většina problémů je spíše kvůli návrhu Blenderu proti běžným zvyklostem.

Asi nejzásadnější problém představuje prohozená funkcionalita tlačítek myši. Zatímco u většiny programů se objekty označují levým tlačítkem myši, u Blenderu je tomu právě naopak – objekty se označují pravým tlačítkem myši. K popisu tohoto problému se později ještě dostaneme.



Obrázek 3: Výchozí rozložení grafického rozhraní Blenderu se může ze začátku jevit jako chaotické.

Grafické prostředí Blenderu (dále GUI) – dlouho jsem zvažoval, zda ho umístit mezi výhody či nevýhody. Rozložení GUI Blenderu je plně přizpůsobitelné, lze rozdělit nebo smazat jeho části, ukládat rozložení a další. Já osobně to považuji za velikou výhodu (GUI lze udržovat čisté a elegantní), jelikož je tento text ale zasvěcen popisu Blenderu, předpokládám, že ho budou číst spíše lidé, kteří Blender neznají. Pro takové lidi se tento aspekt může stát spíše překážkou, protože Blender je navržen na ovládání klávesovými zkratkami. Jelikož začátečníci budou sotva znát všechny tyto zkratky, musí se zpravidla „proklikat“ několika menu. Jelikož ale každá část GUI Blenderu má svoji vlastní lištu s menu, je mnohdy obtížné určit, kde položku vůbec hledat. Mnohdy tak musí přijít na pomoc internet.

Ještě do nedávna Blender neupozorňoval uživatele na neuložený projekt, pokud ho chtěl uživatel zavřít, takže pokud jste omylem klikli místo na ikonu „mi-

nimalizovat“ na ikonu „zavřít“, Blender se bez námitek zavřel a práce byla ztracena.[2] Poslední verze už tuto funkci mají, je však vhodné tento aspekt zmínit, protože starší verze se ještě stále používají.

V málo případech se stalo, že starší projekt byl v nové verzi „rozbitý“. Zřejmě tedy dochází k problémům s kompatibilitou mezi verzemi, nutno ovšem dodat, že se to opět týká spíše starších verzí a osobně jsem zaznamenal jen málo případů (navíc, jak se zmiňuji ve výhodách Blenderu, je možné projekt dále otevírat ve starší verzi Blenderu).

[2]Blender nabízí možnost načtení posledního dokumentu, osobně však nevím, jak přesně funguje, protože mnohdy tak dostanete jiný projekt, než na kterém jste pracovali. Nemohu však mluvit o posledních verzích Blenderu, u kterých jsem tuto funkci ještě nepoužil.

4 Základy práce v Blenderu

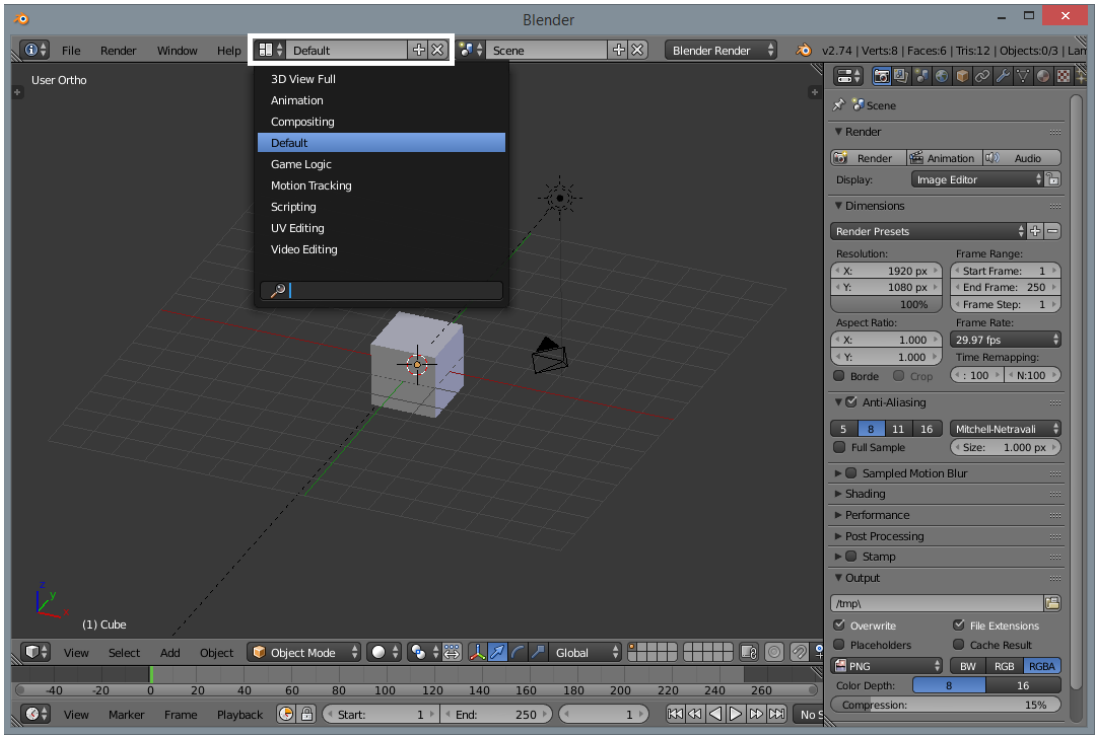
Jelikož nemohu předpokládat, že každý, kdo bude chtít vyzkoušet generátor měst (téma této práce), umí pracovat v Blenderu, rozhodl jsem se věnovat této problematice kratší kapitolu.

4.1 Grafické rozhraní Blenderu

Jak jsem již zmínil v předchozí kapitole, rozložení GUI Blenderu je plně přizpůsobitelné, ale pro práci s generátorem měst si vystačíme pouze s výchozím rozložením.

Pro pokročilejší práci je však dobré mít alespoň povrchové znalosti o GUI Blenderu, minimálně jak přepnout výchozí rozložení do rozložení pro například animaci. Přepínání je velice jednoduché, na horní liště[3] je seznam rozložení, kde je možné si po kliknutí na dropdown ikonu vybrat jiné rozložení (viz obrázek 4).

Animovat se dá samozřejmě i z výchozího rozložení, nemáme však k dispozici grafy a další užitečné pomůcky.[4]



Obrázek 4: Grafické rozhraní Blenderu

[3]Ve skutečnosti se nejedná o horní lištu okna Blenderu, ale o spodní lištu části GUI, která je umístěna nahoře a je zmenšená natolik, že jde vidět pouze lišta. Jelikož se opět jedná o část GUI, lze ji zvětšit, smazat, rozdělit atd. Blender nemá žádnou globální lištu

[4]Pokročilí uživatelé Blenderu samozřejmě vědí, že je možné si je zobrazit přidáním další části do rozložení GUI.

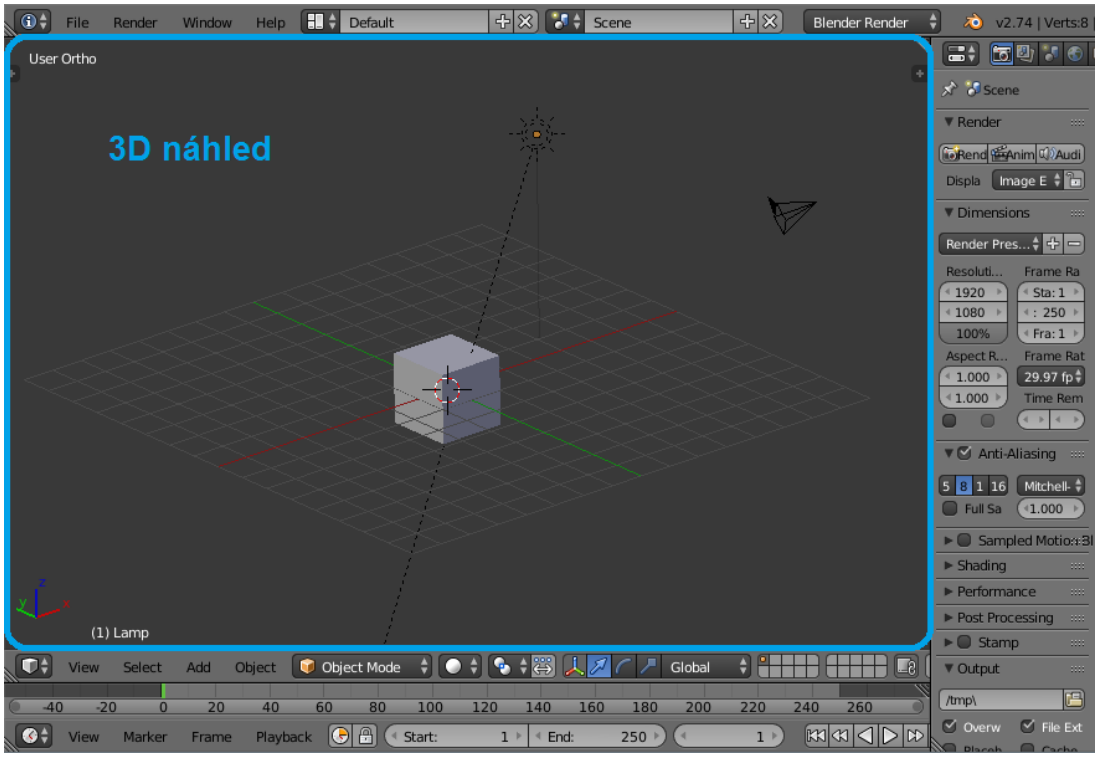
Na pravé straně výchozího rozložení je nastavení pro scénu, renderování, informace o označeném objektu, kameře, možnosti materiálů, textur atd., na spodní části je pak časová osa. Obě tyto části zmiňuji spíše pro úplnost, pro práci s generátorem měst je nepotřebujeme.

4.2 Myš a klávesové zkratky

Jak bylo zmíněno v předchozí kapitole, vývojáři Blenderu se odchýlili od typického označování objektů levým tlačítkem myši, a objekty v Blenderu se tak musí označovat pomocí pravého tlačítka myši.[5]

Co tedy dělá levé tlačítko myši? Mění pozici 3D kurzoru.[6] Ten, mimo jiné, určuje, kde se vytvoří nový objekt a dá se také využít pro transformace (například rotace okolo něj).

Užitečné je také kolečko myši, scrollováním se typicky přibližuje/oddaluje. Pokud se kolečko stiskne a pohybuje se myší, mění se úhel 3D náhledu. Jestliže se ke stisknutému kolečku navíc podrží klávesa Shift, dá se pohybováním myši posouvat 3D náhled.



Obrázek 5: 3D náhled v Blenderu

[5]Zajímavé je, že se jedná o jeden z nejvíce kritizovaných a diskutovaných aspektů Blenderu. Tato zdánlivá drobnost již dokázala odradit spoustu lidí. Ze zkušenosti vím, že neustálé klikání na špatné tlačítko je začátku značně nepříjemné. Je sice možné funkce tlačítek změnit, kvůli nekonzistenci v různých částech GUI se to však nedoporučuje.

[6]3D kurzor je bílo-červený kroužek v 3D prostoru. Výchozí pozice je (0,0,0).

Klávesových zkratek je mnoho, my si však vystačíme s naprostými základy. Následující výčet zkratek (tabulka 1) je určen pro 3D náhled (viz obrázek 5), mějte proto na paměti, že při použití těchto zkratek musí být kurzor myši právě nad 3D náhledem!

Tabulka 1: Klávesové zkratky v 3D náhledu Blenderu

zkratka	funkce
0	přepnutí 3D náhledu do pohledu kamery
1	přepnutí 3D náhledu do pohledu zepředu
3	přepnutí 3D náhledu do pohledu z boku
7	přepnutí 3D náhledu do pohledu shora
2	pootočení 3D náhledu dolů
8	pootočení 3D náhledu nahoru
4	pootočení 3D náhledu doleva
6	pootočení 3D náhledu doprava
5	přepínání mezi perspektivním a orthografickým zobrazením
T	zobrazí panel s nástroji na levé straně 3D náhledu (zde po instalaci naleznete generátor měst)
N	zobrazí panel s nastavením na pravé straně 3D náhledu
Tab	přepnutí mezi objektovým módem (posun a další transformace) a editačním módem (práce s vrcholy, tvar objektu) označeného objektu
G	posun označeného objektu
R	rotace označeného objektu
S	změna velikosti označeného objektu
Del	odstranění označeného objektu (vyžaduje potvrzení)

Pro vytvoření města si teoreticky vystačíme pouze s klávesovou zkratkou „T“, abychom se dostali do nastavení generátoru měst. Jestliže si následně chcete vytvořený 3D model prohlédnout, na řadu přichází číselné klávesové zkratky (tabulka 1) pro navigaci v 3D prostoru (nebo stisknuté kolečko myši).

4.3 Instalace doplňku

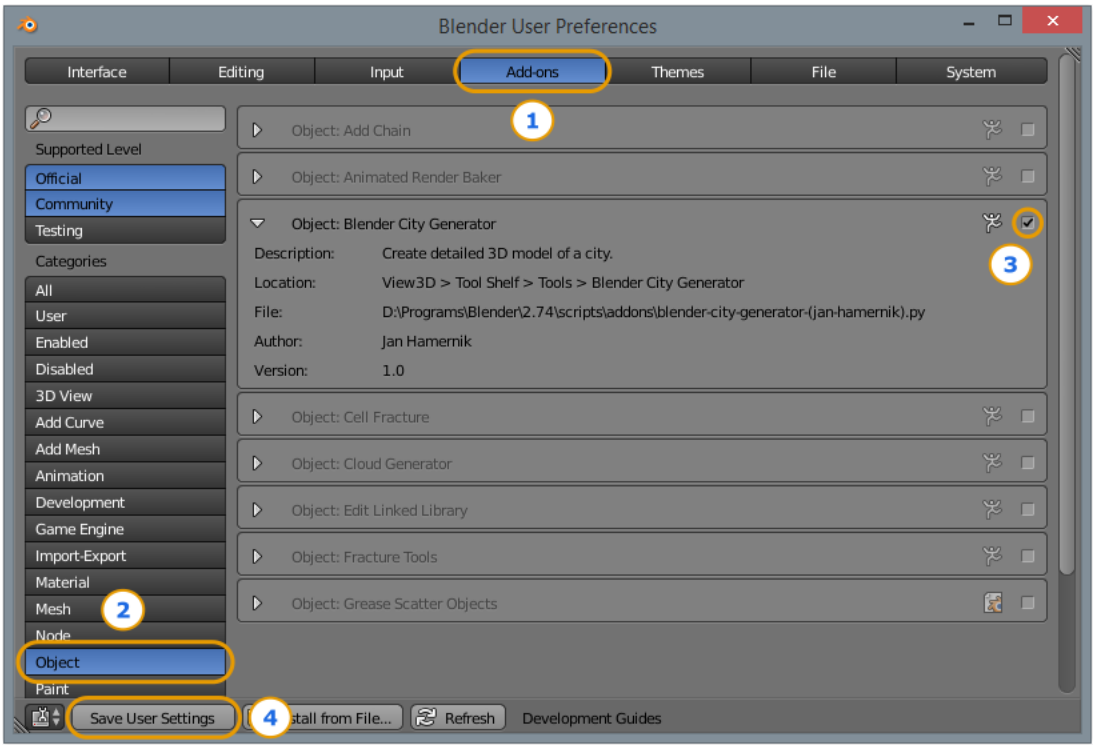
Instalace doplňku do Blenderu je poměrně jednoduchá. Nejprve otevřeme přiložené CD, a pomocí klávesové zkratky „Ctrl+C“ zkopírujeme všechny soubory z adresáře „src/“. Měly by to být tyto:

- složka „blender_city_generator“,
- soubor „blender-city-generator-(jan-hamernik).py“.

Poté ve složce nainstalovaného Blenderu vyhledáme složku určenou pro doplňky („addons“). Cesta k této složce bude pravděpodobně vypadat obdobně jako tato:

- C:\Program Files\Blender Foundation\Blender\2.74\scripts\addons.

Do této složky vložíme zkopírované soubory (klávesová zkratka „Ctrl+V“). Poté můžeme složku zavřít a otevřeme Blender. Na horní liště v Blenderu klikneme na „File“ a vybereme položku „User Preferences...“ (nebo použijeme klávesovou zkratku „Ctrl+Alt+U“), což vyvolá nové okno s nastavením Blenderu.

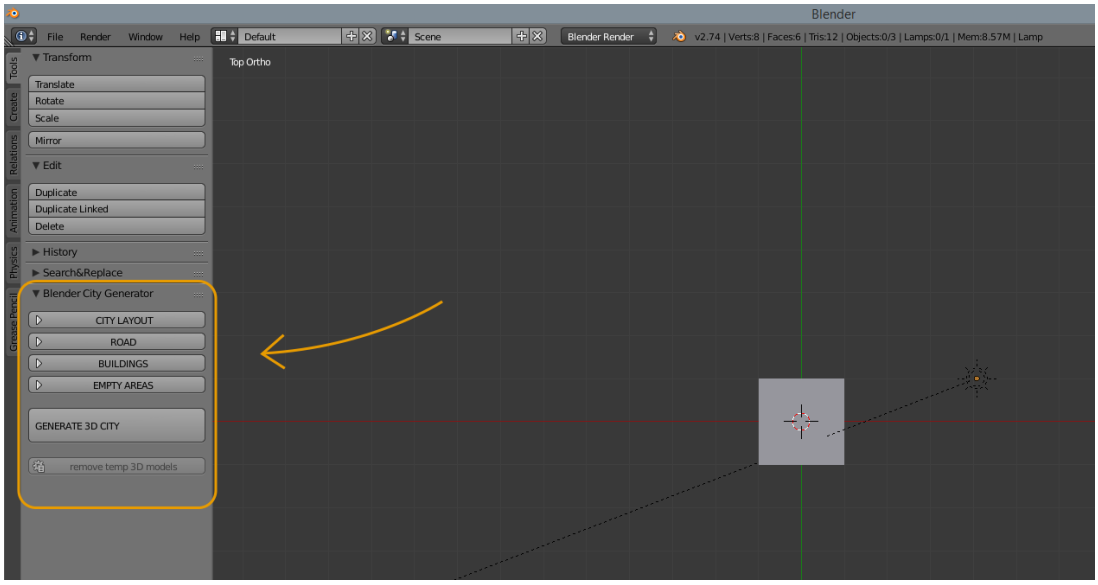


Obrázek 6: Okno „Blender User Preferences“ – nastavení Blenderu

Okno s nastavením je k vidění na obrázku 6, kde je také očíslovaný postup. Nejprve tedy musíme kliknout na záložku „Addons“, a poté zvolit kategorii „Ob-

ject“.[7] V hlavní části okna vyhledáme požadovaný doplněk („Object: Blender City Generator“), a pomocí tlačítka označeného číslem 3 doplněk aktivujeme. Další krok (4) je dobrovolný. Jedná se o tlačítko „Save User Settings“, které, jak název napovídá, uloží aktuální nastavení. V praxi to znamená, že generátor měst se vždy při otevření Blenderu automaticky načte (není tedy nutné jej pokaždé ručně aktivovat). Nakonec okno s nastavením můžeme zavřít.

Instalace je tímto prakticky skončena. Když teď s kurzorem nad 3D modelem stiskneme klávesu „T“, zobrazí se tvz. „Tool Shelf“. Vlevo má Tool Shelf mnoho dalších záložek, nám však stačí pouze první záložka „Tools“. Právě této záložce bychom měli najít ovládací panel pro generátor měst (ciz obrázek 7).



Obrázek 7: Okno „Blender User Preferences“ – nastavení Blenderu

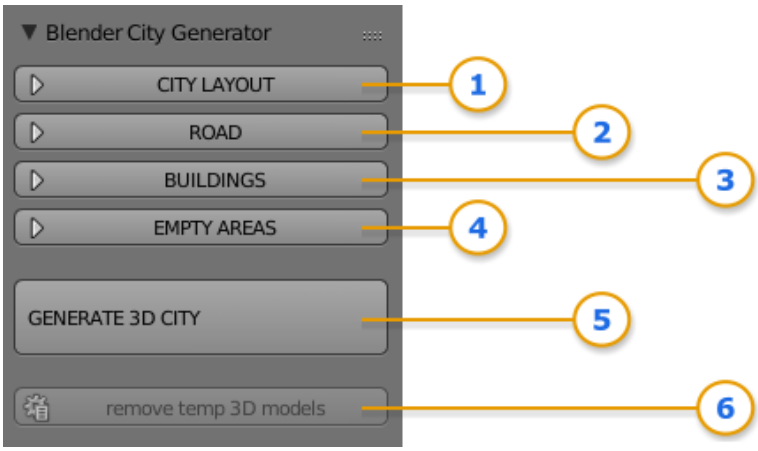
4.3.1 Potřebná verze Blenderu

Generátor měst jsem vyvíjel v Blenderu 2.74, doporučuji tedy pro práci s ním používat také verzi 2.74. Na předchozích verzích by kód sice měl fungovat také, není to však otestované. Ve složce „install/“ jsou k dispozici instalační soubory pro Blender 2.74 na různé operační systémy.

[7] Tento krok můžeme přeskočit, a na místo vyhledávání v kategoriích můžeme doplněk vyhledat pomocí vyhledávacího políčka v levém horním rohu. Název pro vyhledávání je „Blender City Generator“.

5 Ovládání generátoru měst

Ovládání je rozděleno do několika sekcí a tlačítek. Sekce (1-4 na obrázku 8) vypadají jako tlačítka,[8] ale mají na levé straně ikonu šipky. Po kliknutí na tlačítko sekce se zobrazí další možnosti ovládání. Jejich detailní rozbor je v dalších podkapitolách.

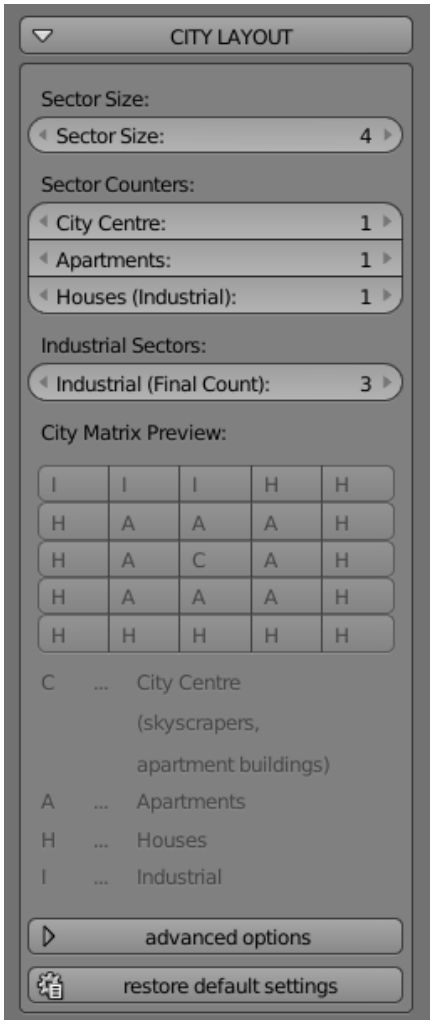


Obrázek 8: Ovládací panel

[8]Nástroje pro vytvoření grafického panelu v GUI Blenderu neumožňují vytvořit rozšiřitelné sekce, proto jsem tento problém musel obejít vytvořením tlačítek, která pouze schovají/zobrazí vybrané části panelu.

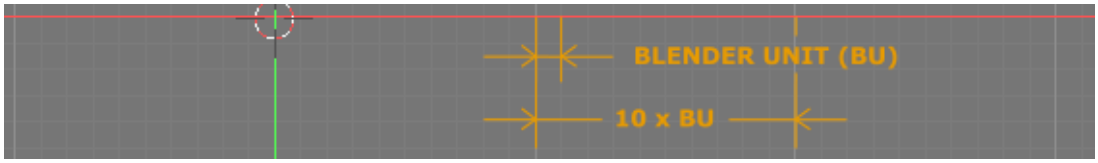
5.1 CITY LAYOUT – Rozložení města

Zde jsou hlavní nástroje pro nastavení vzhledu a rozměrů města. Město je ve své podstatě čtverec složený z několika sektorů (menších čtverců).



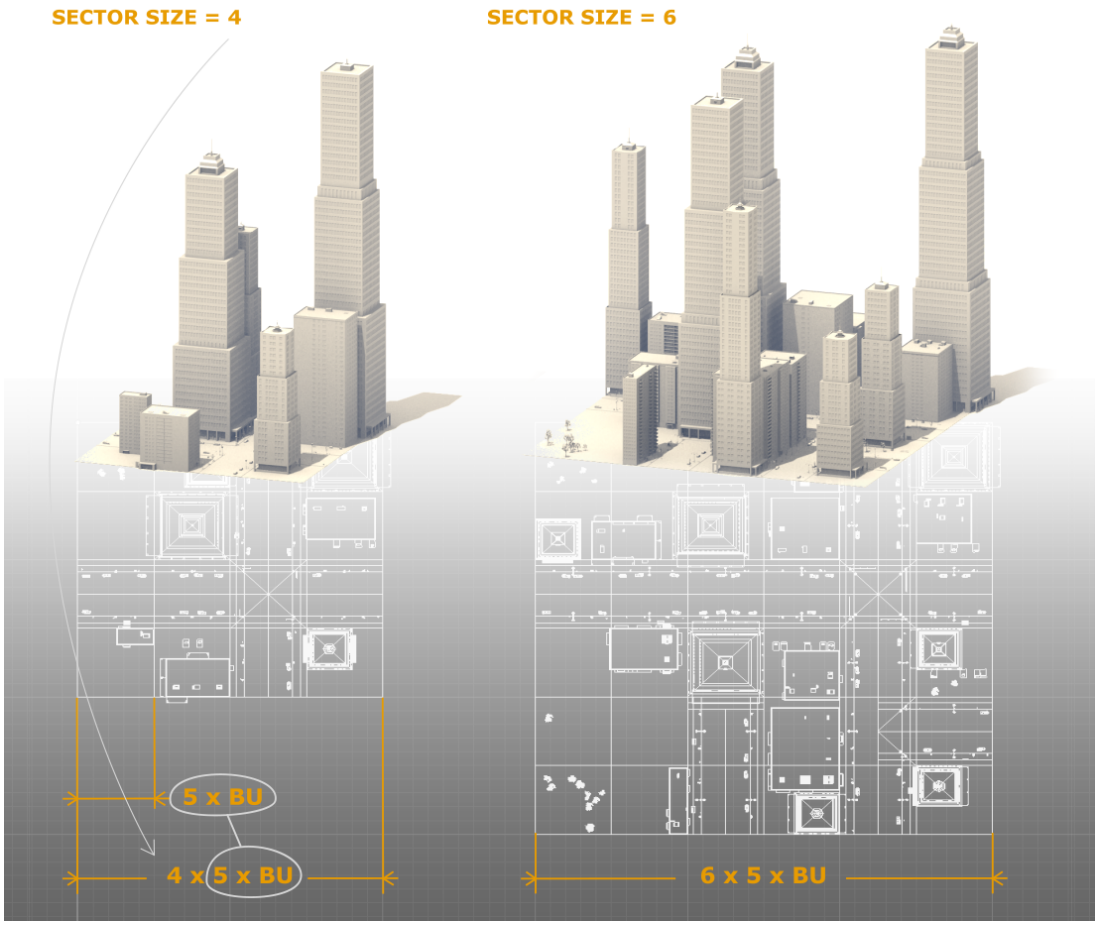
Obrázek 9: Ovládací panel – CITY LAYOUT

Velikost těchto sektorů se dá nastavit pomocí parametru „Sector Size“. Všechny parametry, které nastavují velikost, používají násobky tzv. Blender jednotky (anglicky „Blender units“, dále budu používat zkratku BU), které jsou zobrazeny na mřížce v 3D pohledu (viz obrázek 10).



Obrázek 10: Blender jednotka (Blender unit)

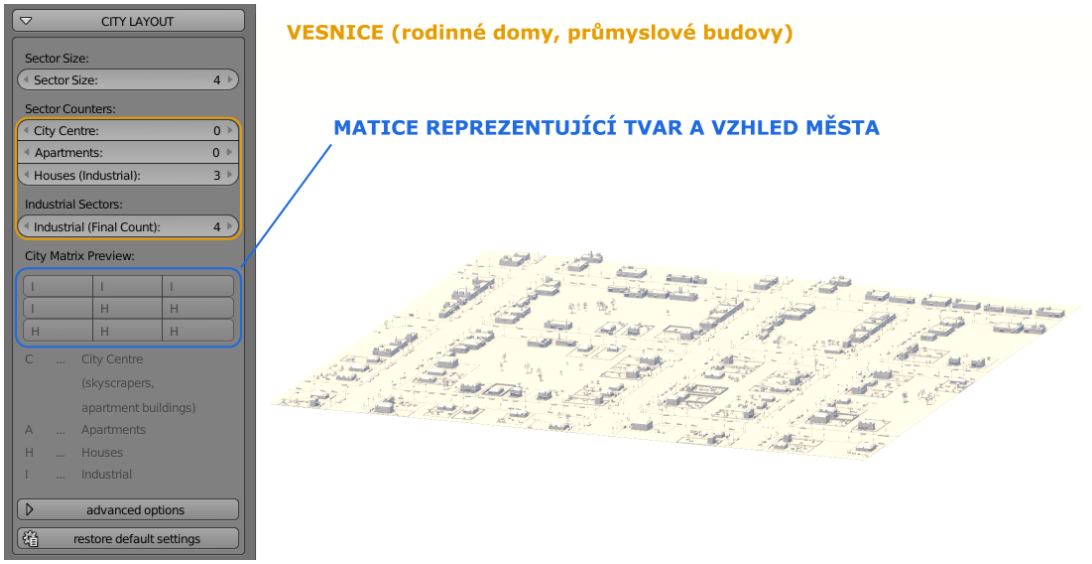
Parametr „Sector Size“ je v jednotkách $5BU$ a z intervalu $\langle 4, 20\rangle$. Každý sektor je opět poskládán z menších částí (čtverců), jejichž velikost je právě $5BU$, proto má „Sector Size“ právě takové jednotky (zjednodušeně řečeno je to počet těchto čtverců). Pro ilustraci uvádím dva příklady (viz obrázek 11) jednoho sektoru, který byl vygenerovaný s rozdílným nastavením „Sector Size“ (4 a 6).



Obrázek 11: Sector Size – velikost sektoru

Všechny parametry mají k dispozici krátký popis, který se zobrazí při delším podržení kurzoru myši nad nimi.

Pomocí „Sector Counters“, neboli „sektorových počítadel“, je možné nastavit rozměry a vzhled města. Rozměr města je dán jako součet všech sektorů.



Obrázek 12: Sector Counters – nastavení pro vesnice

Obrázek 13: Sector Counters – nastavení pro centrum města

Sektor „City Centre“ obsahuje obytné domy a mrakodrapy, sektor „Apartments“ pak pouze obytné domy.

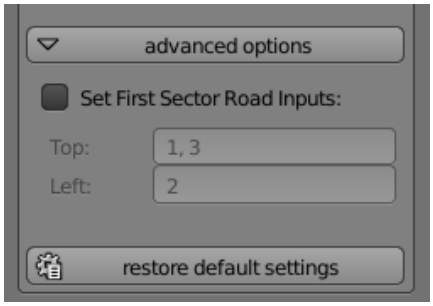
Počítadlo „Houses (Industrial)“ je společné pro rodinné domy (houses) a průmyslové budovy (industrial). Výchozí sektor pro tohle počítadlo je „Houses“ (tedy sektor obsahující rodinné domy), přičemž, po nastavení počtu těchto sektorů, je možné pomocí počítadla „Industrial (Final Count)“ změnit některé z nich na sektory „Industrial“ (sektory s průmyslovými budovami).

Počítadlo „Industrial (Final Count)“ udává na rozdíl od předchozích počítadel skutečný počet sektorů.

Jelikož je těžké si pod tímto popisem přesně představit konečnou podobu města, pod počítadly je k dispozici matice (viz obrázek 12), která reprezentuje konečné rozložení sektorů (rozměr a vzhled města). Matice také názorně ukazuje, jak funguje počítadlo „Houses (Industrial)“ a „Industrial (Final Count)“.

5.1.1 Pokročilé možnosti

Tlačítko „advanced options“ otevírá další (pokročilé) nastavení pro tvorbu města. Zde je možné prvnímu vygenerovanému sektoru nastavit vstupní cesty, pokud je zaškrtnuté políčko „Set First Sector Road Inputs“ (ve výchozím nastavení je vypnuté, viz obrázek 14).



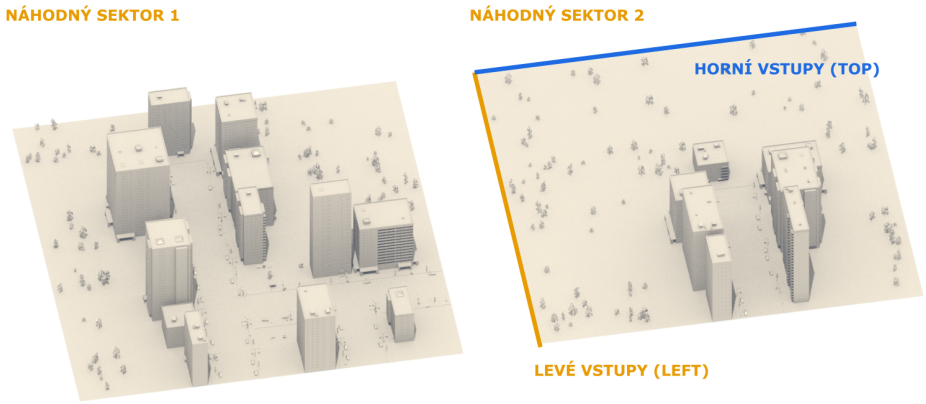
Obrázek 14: Pokročilé možnosti a tlačítko výchozího nastavení CITY LAYOUT

Tato zdánlivě nepotřebná věc otevírá nové možnosti pro „postupné generování měst“. Tímto pojmem myslím proces, kdy si uživatel vždy vygeneruje pouze jeden sektor, který následně ručně přesune na požadované místo. Cesty se v každém sektoru tvoří automaticky a náhodně, když si tedy uživatel vygeneruje další sektor, s velikou pravděpodobností na sebe nebudou cesty mezi sektory navazovat.

Tento problém řeší právě „advanced options“. Než tedy uživatel vygeneruje další sektor, nastaví mu vstupní cesty podle výstupních cest sousedního sektoru. Na straně 23 jsem uvedl, že sektor je poskládán z menších čtverců. Když tedy zadáváme vstupní cesty sektoru, musíme do políček „Top“ a „Left“ napsat indexy právě těch čtverců, na kterých chceme, aby začínala cesta. Existuje zde však několik striktních pravidel:

1. Vstupní indexy jsou indexovány od nuly. Například pro sektor velikosti „Sector Size = 4“ tak připadají v úvahu indexy 0, 1, 2, 3. Ne všechny tyto indexy lze ovšem použít, jak definují další pravidla.

2. Políčka „Top“ a „Left“ mohou být i prázdná – nezadáme tedy žádné vstupní indexy. Jelikož budovy se tvoří pouze okolo cest, výsledný sektor může být prázdný (podle nastavení zaplněný stromy, viz dále), ale s vysokou pravděpodobností uvnitř něj začnou náhodně nové cesty s budovami (viz obrázek 15).



Obrázek 15: Sektory s nastavenými prázdnými vstupy cest

3. Vstupních indexů může být jeden nebo více. Musí ale být odděleny čárkou, přičemž mezi čárkou a čísly může být libovolně dlouhá mezera a indexy nemusí být seřazené. Následující vstupy jsou tedy ekvivalentní:

   - 2
   - 1,3
   - 1, 3
   - 3, 1
   - 1 , 3
   - 3, 1

   Tyto vstupy jsou naopak nepřijatelné:

   - 1 3
   - 1; 3

4. Vstupní indexy nesmí následovat ihned po sobě, musí mezi sebou mít minimálně jeden volný index. Tyto vstupy jsou tedy přijatelné:

   - 1, 3
   - 1, 4
   - 2, 4

Toto omezení má hned dva důvody. Prvním důvodem je, že v reálném světě není běžné stavět dvě cesty ihned vedle sebe a oddělené chodníkem (vygenerované cesty mají vždy po stranách chodník), což vede k druhému důvodu, že algoritmus pro generování cest nebere tuto možnost v úvahu, a vidí-li vedle sebe dvě cesty, spojí je křižovatkou.

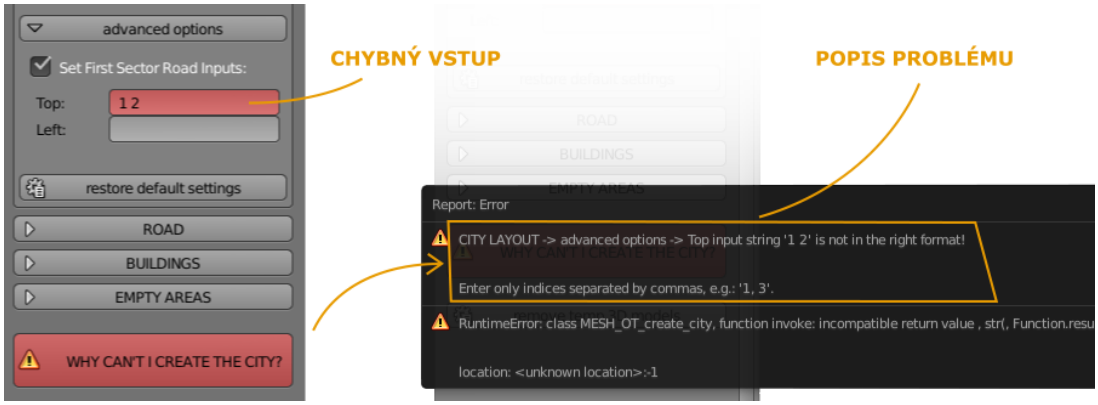
Kdyby tedy vedle sebe vedly dvě cesty, ve výsledku bychom měli mnoho křižovatek jednu vedle druhé, což také není v reálném světě běžné. Následující vstupy jsou tedy naopak nepřijatelné:

- 1, 2
- 3, 4

5. Žádný ze vstupních indexů se nesmí rovnat ani jednomu krajnímu indexu. Dříve jsem uvedl, že pro sektor velikosti „Sector Size = 4“ připadají v úvahu indexy 0, 1, 2, 3. V tomto případě tedy nesmíme použít ani index 0, ani index 3.

   Důvodem je opět zamezení případu, kdy dvě cesty povedou vedle sebe (předchozí pravidlo), protože při tvorbě sektoru nemá algoritmus žádnou informaci o sousedních sektorech a jejich cestách (logicky ji ani mít nemůže, protože minimálně dva sousední sektory při jeho generování ještě neexistují). Může tak nastat případ, že dva sousední sektory budou mít na sousedních hranách cesty, což představuje kromě problému sousedních cest další problém – kde přesně na sebe mají tyto cesty navázat (tedy kde bude křižovatka), aby se tak propojili sektory.

Pokud je do políčka „Top“ nebo „Left“ zadaný špatný vstup, políčko zčervená a hlavní tlačítko pro vytvoření města ze změní z „GENERATE 3D CITY“ na „WHY CAN’T I CREATE THE CITY?“, také zčervená, a znemožní vytvoření města.



Obrázek 16: Chybně zadaný vstup

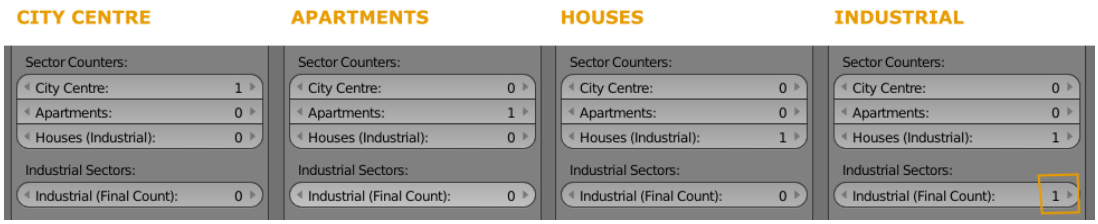
Tlačítko takto bude vypadat, dokud uživatel vstup neopraví . Jak nový popisek tlačítka napovídá, pokud na něj nyní kliknete, zobrazí se chybová hláška

s popisem problému (viz obrázek 16). V horní části popisu je cesta v ovládacím panelu (kdyby uživatel zapomněl kde viděl ono červené políčko) s obecným popisem problému typu „vstup je ve špatném formátu“. Ve spodní části popisu je poté detailnější popis problému, například že vstup musí být oddělený čárkami, zadané indexy jsou mimo povolené meze atd. (viz pravidla na straně 26).

5.1.2 Využití pokročilého nastavení

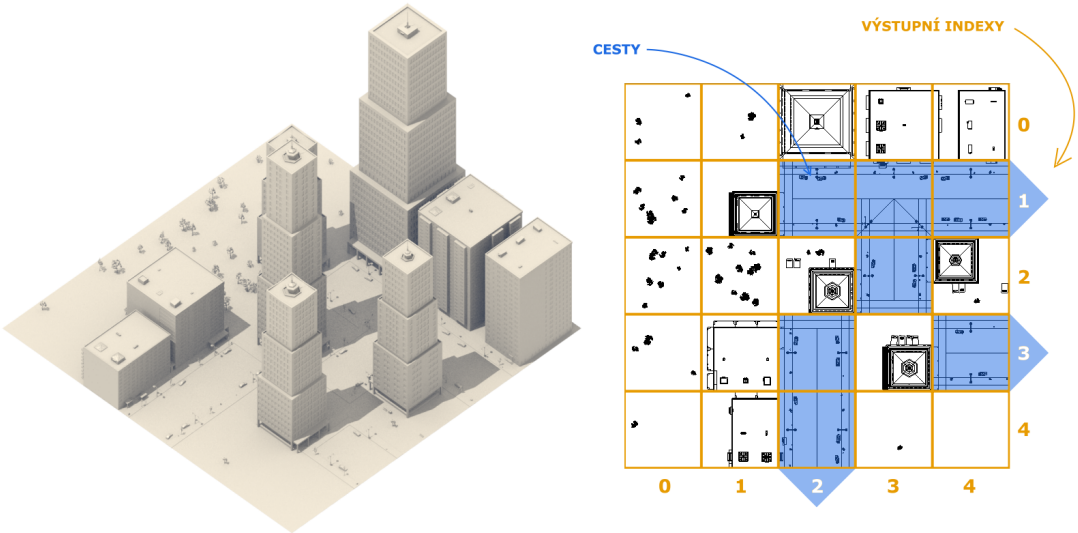
Když už máme definovaná pravidla pro zadávání vstupů, podíváme se, jakým způsobem se toho dá využít při skládání vlastního města ze sektorů.

Nejprve si vybereme, jaký druh sektoru chceme vygenerovat, a nastavíme mu počítadlo na „1“. Pro sektor s průmyslovými budovami (industrial) musíte nastavit „1“ jak na počítadlu „Houses (Industrial)“, tak na počítadlu „Industrial (Final Count)“ (viz obrázek 17).



Obrázek 17: Nastavení města na jeden sektor podle typu.

Jelikož se jedná o první sektor, můžeme stisknout tlačítko „GENERATE 3D CITY“, které nám vygeneruje 3D model sektoru. Vstupy cest budou zadány náhodně, což nám u prvního sektoru nevadí.

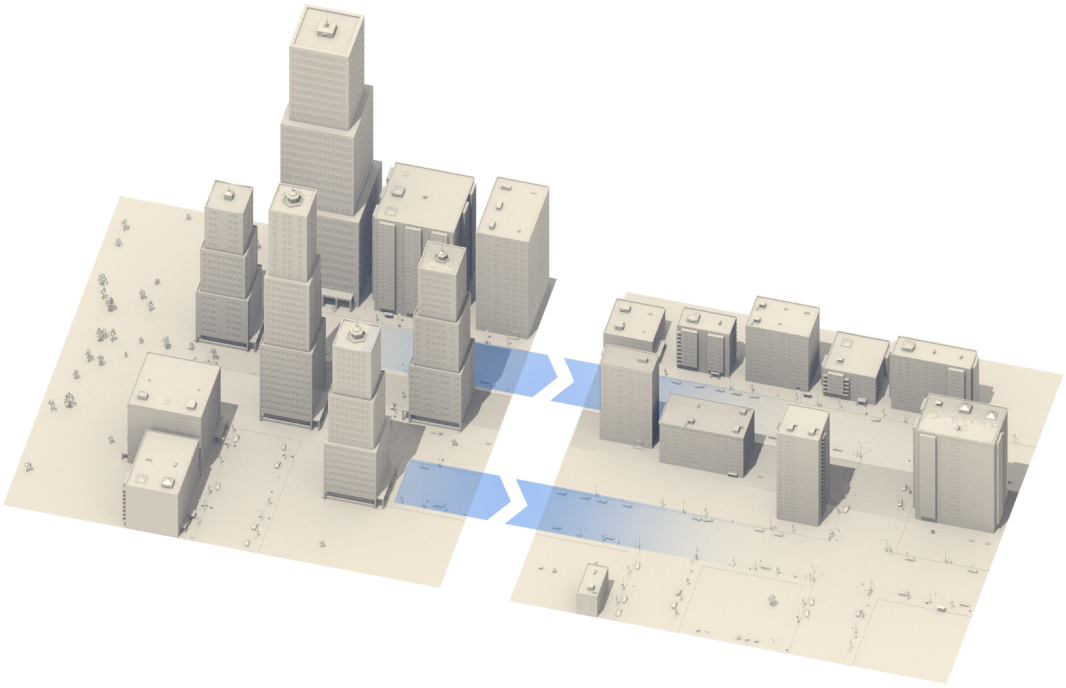


Obrázek 18: První vytvořený sektor a jeho výstupní indexy.

Nyní bychom tedy chtěli vytvořit a navázat druhý sektor, jenomže tentokrát už cesty nemohou začínat náhodně, protože by neseděly s prvním sektorem. Využijeme tedy pokročilé možnosti (advanced options).

Aby vše správně fungovalo, musí být zaškrtnuté políčko „Set First Sector Road Inputs“. Předpokládejme, že chceme další sektor napojit na pravou stranu prvního sektoru.[9] Podívejme se tedy, jaké mají jeho cesty výstupní indexy. Z obrázku 18 je vidět, že výstupní indexy na pravé straně jsou 1 a 3, zadáme je tedy do políčka „Left“. Horní vstupy („Top“) jsou v tuto chvíli volitelné.

Nyní můžeme ještě změnit nastavení sektoru, například zvolit jiný typ. Poté sektor opět vygenerujeme pomocí tlačítka „GENERATE 3D CITY“. Jakmile je vygenerován, můžeme ho přesunout vedle prvního sektoru (klávesová zkratka „G“ a doporučuji držet „Ctrl“ pro posun po Blender jednotkách). Na obrázku 19 je vidět, že sektory na sebe navazují.

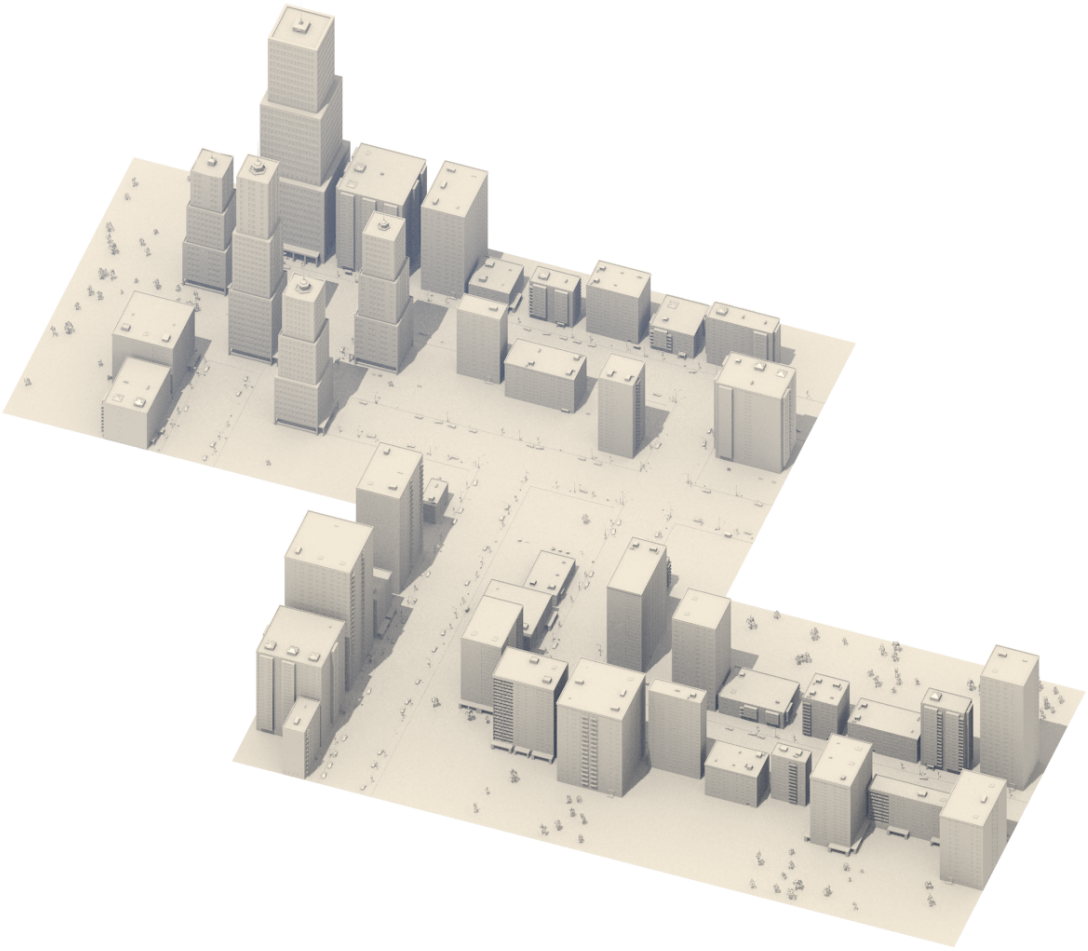


Obrázek 19: Zřejmá návaznost sektorů.

Obdobným způsobem tedy můžeme vygenerovat libovolné množství dalších sektorů, kdy vždy nastavíme požadované vstupy a výsledný sektor napojíme na předchozí. V této ukázce jsem použil pouze vstupy „Left“, vše však analogicky platí také pro vstupy „Top“.

[9]Sektory lze napojovat pouze z pravé nebo spodní strany, protože můžeme volit vstupní cesty pouze z horní a levé strany („Top“ a „Left“). Trasa cesty je pak generována náhodně, výstupy se tedy nedají ovlivnit.

Výhodou tohoto přístupu je, že si můžete poskládat jakýkoli tvar města, libovolně veliký a z libovolných typů sektorů. Další výhodou je, že jestliže se vám vygenerovaný sektor nelíbí, můžete ho smazat, a vygenerovat jiný. Pozor však na mazání sektorů, které už mají své sousedy vpravo nebo dole! Výstupy cest nemůžete ovlivnit, nicméně u malých velikostí sektorů (4-5) je veliká pravděpodobnost že se vygenerují takové, aby opět navazovaly.



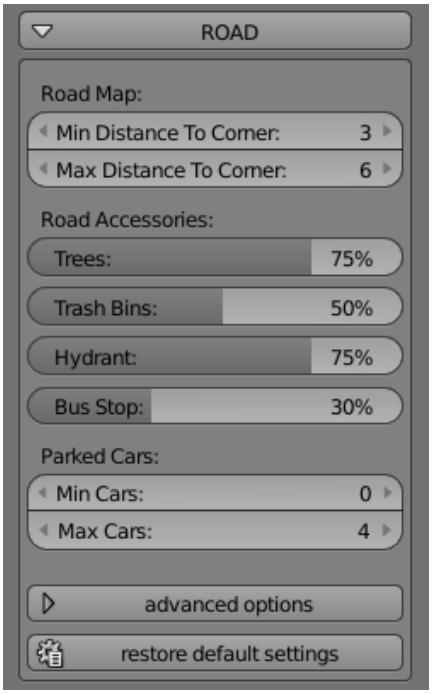
Obrázek 20: Příklad vlastního poskládaného tvaru města.

5.1.3 Obnovení výchozího nastavení

Po kliknutí na tlačítko „restore default settings“ se obnoví výchozí nastavení všech parametrů sekce „CITY LAYOUT“.

5.2 ROAD – Mapa cest, příslušenství

V této sekci je možné měnit nastavení pro generování cest a jejich příslušenství (objekty související s cestami, například auta).



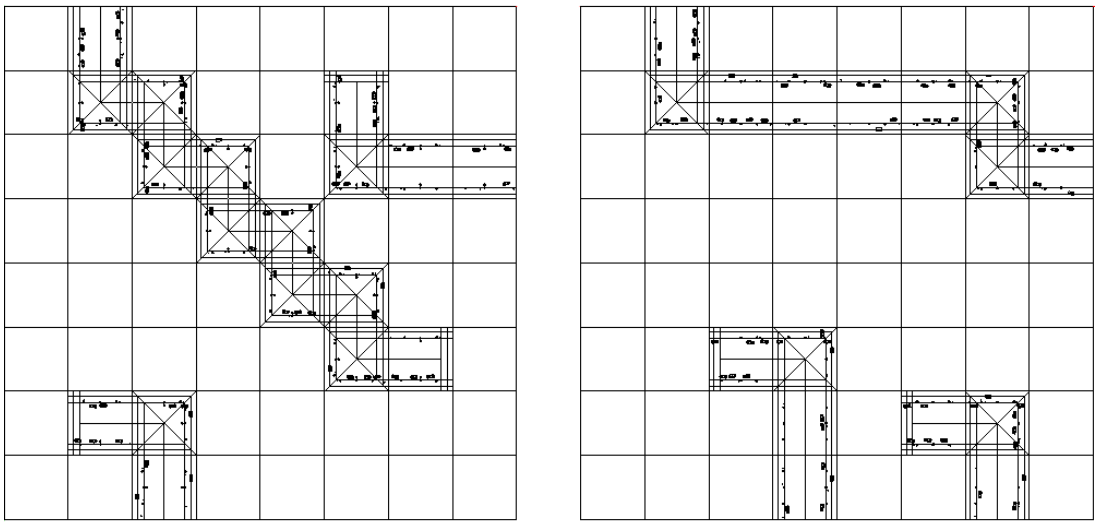
Obrázek 21: Ovládací panel – ROAD

5.2.1 Road Map – Mapa cest

Pomocí parametrů „Min Distance To Corner“ a „Max Distance To Corner“ lze ovlivnit „klikatost“ cest. Parametr „Min Distance To Corner“ určuje minimální vzdálenost (počet čtverců sektoru, tedy $5BU$), kterou musí cesta urazit, než se vyskytne zatáčka. Totéž analogicky platí pro „Max Distance To Corner“, jen se jedná o maximální vzdálenost. Na obrázku 22 lze vidět rozdílné výsledky pro nízké a vyšší hodnoty těchto parametrů.

Je důležité si uvědomit, že tyto parametry uživateli nedávají naprostou moc nad zatáčkami cest. Algoritmus pro vygenerování cesty totiž bere v úvahu další informace, na základě kterých může parametry „Min Distance To Corner“ a „Max Distance To Corner“ ignorovat.

Například má algoritmus definované, jak se má zachovat, pokud narazí na sousední cestu. V takovém případě totiž mohou vzniknout dvě cesty, které vedou těsně vedle sebe. Když je potom taková mapa cest předána dalšímu algoritmu pro vytvoření 3D modelu cest, vyhodnotí sousední cesty jako spoustu křižovatek vedle sebe, což je velice nežádoucí. Aby se takovému případu zabránilo, cesta musí buď zatočit (bez ohledu na parametry „Min Distance To Corner“



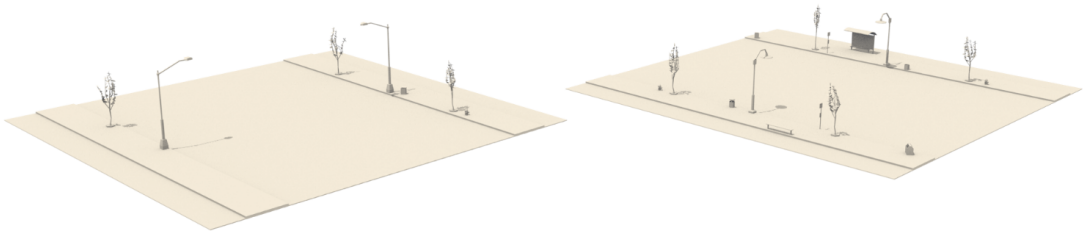
Obrázek 22: Nízké hodnoty „Min Distance To Corner“ a „Max Distance To Corner“ (vlevo => klikatá cesta), Vyšší hodnoty (vpravo)

a „Max Distance To Corner“), nebo je ukončena (v takovém případě většinou vznikne křižovatka se sousední cestou).

Z předchozího odstavce tedy plyne, že čím více cest (a jejich kolizí), tím menší mají parametry „Min Distance To Corner“ a „Max Distance To Corner“ význam.

5.2.2 Road Accessories – Doplňkové objekty cest

Zde je možné nastavit pravděpodobnosti výskytů různých doplňkových objektů cest. Patří sem „Trees“ (stromy po stranách cesty), „Trash Bins“ (odpadkové koše), „Hydrant“ a „Bus Stop“ (autobusová zastávka). Vzhled a pozice doplňkových objektů jsou stále náhodné.

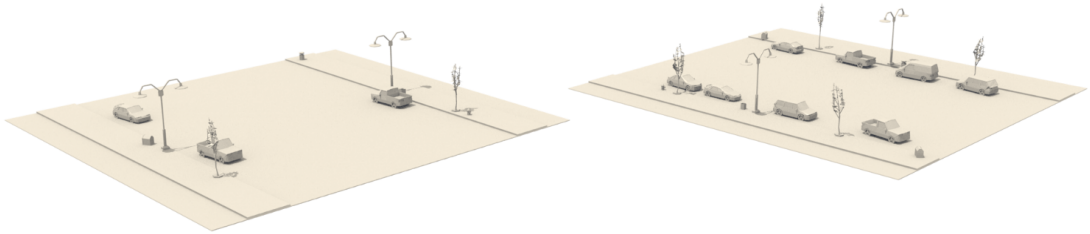


Obrázek 23: Náhodně vytvořené části cest s doplňkovými modely

5.2.3 Parked Cars – Zaparkovaná auta

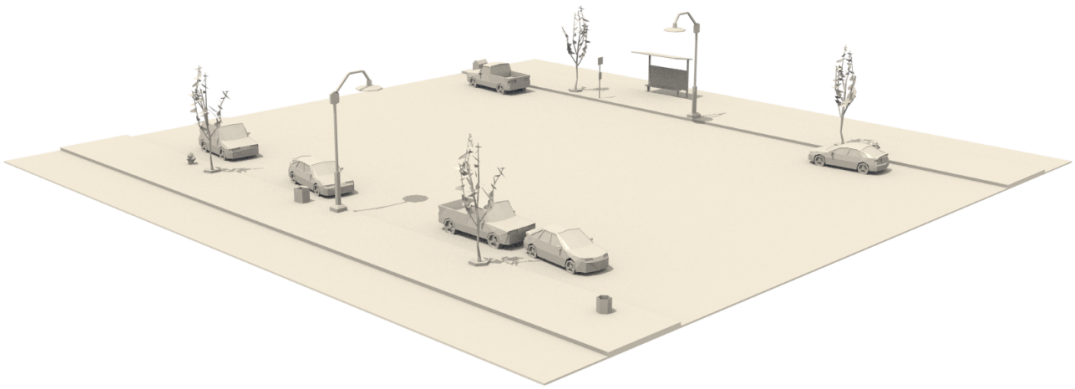
Pomocí parametrů „Min Cars“ a „Max Cars“ lze jednoduše kontrolovat minimální a maximální počet zaparkovaných aut po stranách cesty. Jelikož zde zadáváme počet, a nikoli pravděpodobnost, platí tyto hodnoty pro každou část cesty (čtverec sektoru) zvlášť.

Hodnoty musí být z intervalu $\langle 0, 4 \rangle$, a platí pro jednu stranu cesty (jestliže tedy nastavíme „Min Cars = Max Cars = 1“, výsledná část cesty bude mít dvě auta – jedno na každé straně cesty). Stejně jako u „Road Accessories“, i zde jsou modely stále generovány náhodně.



Obrázek 24: Náhodně vytvořené části cest se zaparkovanými auty

Jestliže se v generované části cesty vyskytuje také autobusová zastávka, platí pro ní pravidlo, že před ní nesmí být zaparkované žádné auto (musí být volné místo pro dvě auta). Toto pravidlo je samozřejmě nadřazené parametrům „Min Cars“ a „Max Cars“. Jestliže tedy nastavíme „Min Cars = Max Cars = 4“, a v části cesty bude autobusová zastávka, auta nebudou čtyři, ale pouze dvě. Tento případ je vidět na pravé straně cesty na obrázku 25.

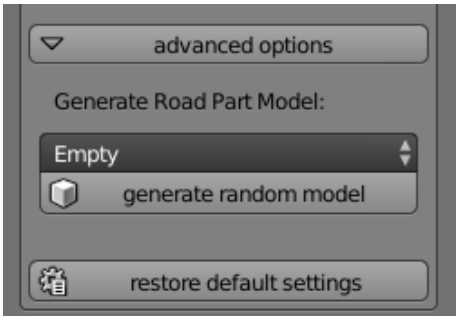


Obrázek 25: Autobusová zastávka

Parametry „Min Cars“ a „Max Cars“ jsou také do jisté míry ignorovány podle aktuální části cesty. Do zatáčky se například čtyři auta nevejdou, to samé do křižovatek, kde nesmí být auta vůbec.

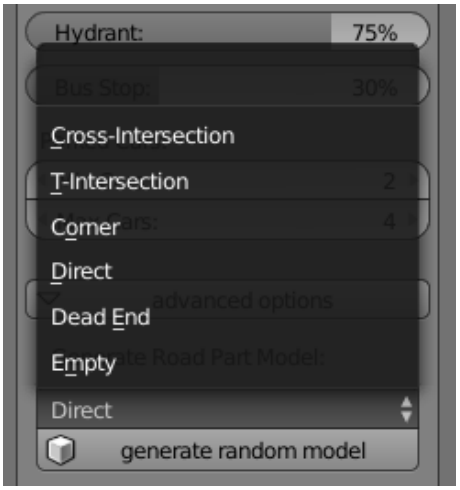
5.2.4 Pokročilé možnosti

V sekci „advanced options“ je možné samostatně vygenerovat 3D modely jednotlivých částí (čtverců) sektorů, podle zadaného typu.



Obrázek 26: Pokročilé možnosti a tlačítko výchozího nastavení ROAD.

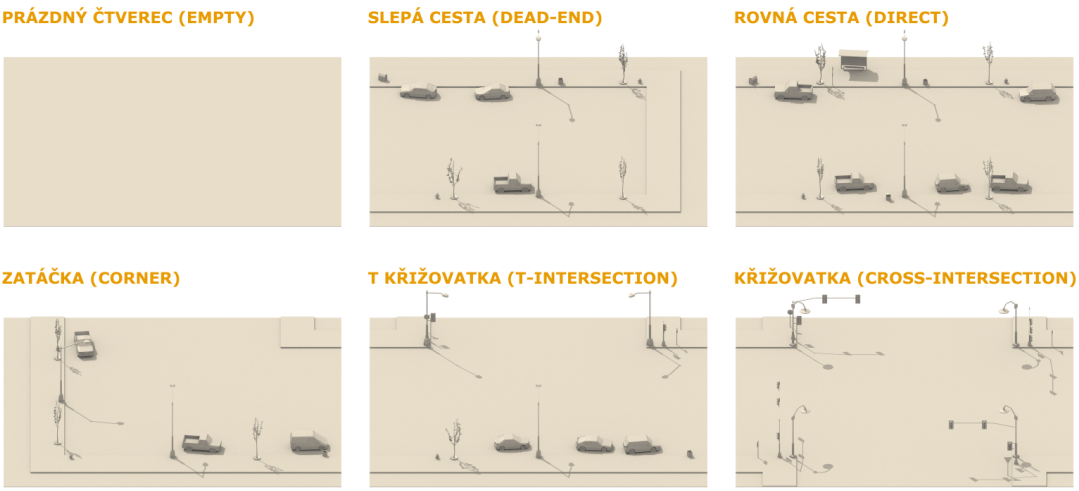
Typ čtverce je možné zvolit z rozbalovací nabídky pod nápisem „Generate Road Part Model“ (viz obrázek 27). Výchozí volba je čtverec „Empty“, tedy prázdný čtverec bez cesty (slouží jako podklad pro budovy, později zjistíte, že i ty se dají generovat po jedné). Ostatní typy čtverců obsahují různé části cest,



Obrázek 27: Výběr typu čtverce z rozbalovací nabídky pokročilých možností.

ze kterých je možné poskládat větší cestu – uživatel tedy opět dostává „volnou ruku“ a může tvořit cesty podle libosti. Na obrázku 28 jsou zobrazeny všechny typy čtverců (cest), které je možné vygenerovat.

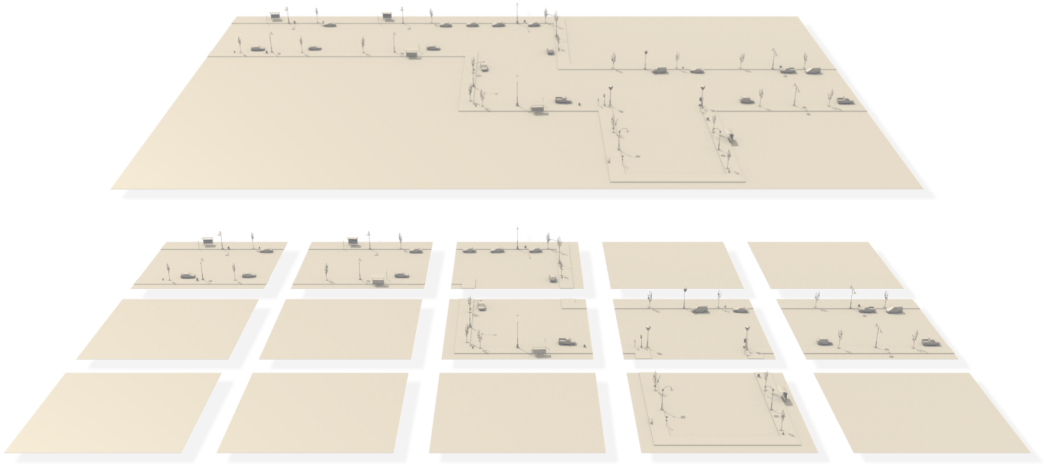
Jakmile má uživatel vygenerovaný 3D model, může s ním opět provádět různé transformace, jako například posun na požadované místo (klávesová zkratka „G“, opět dopočuji držet při posunu klávesu „Ctrl“). Tentokrát bude pravděpodobně potřeba také rotace (klávesová zkratka „R“, opět doporučuji „Ctrl“, tentokrát pro přesnou rotaci po pěti stupních). Jelikož jsme schopni vygenerovat například



Obrázek 28: Typy čtverců (cest).

pouze zatáčku „shora doprava“ (viz obrázek 28 vpravo nahoře), musíme další zatáčky vytvořit otočením této zatáčky.

Tímto způsobem jsme tedy schopni vytvořit všechny potřebné 3D modely částí cest, a poskládat tak vlastní cestu. Příklad takto vytvořené cesty je na obrázku 29.



Obrázek 29: Příklad ručně poskládané cesty s prázdnými místy pro budovy.

5.2.5 Obnovení výchozího nastavení

Po kliknutí na tlačítko „restore default settings“ se obnoví výchozí nastavení všech parametrů sekce „ROAD“.

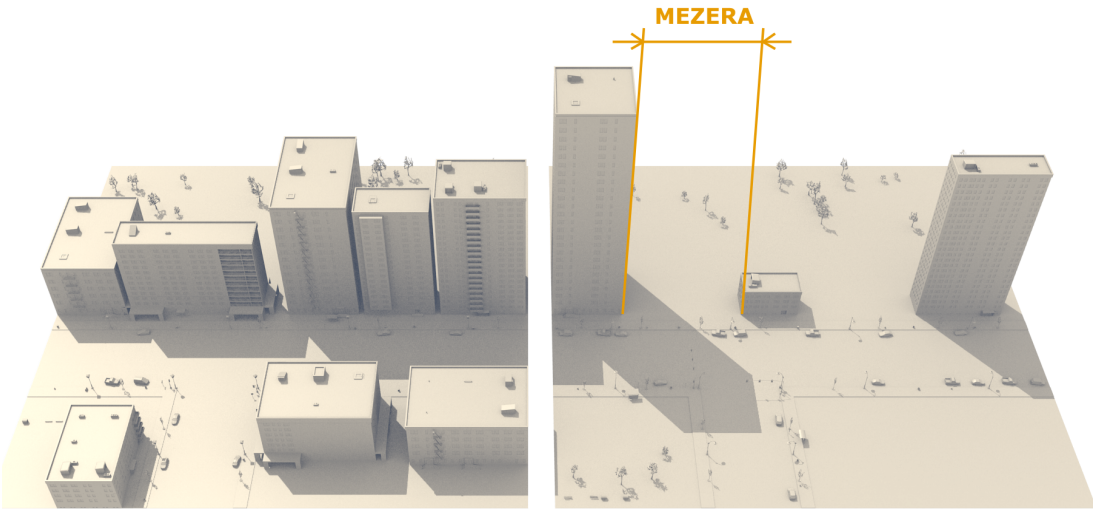
5.3 BUILDINGS – Nastavení budov

V této sekci je možné měnit nastavení parametrů, podle kterých se náhodně tvoří budovy.

Obrázek 30: Ovládací panel – BUILDINGS.

Jediným společným parametrem pro všechny budovy je „Gap Between Buildings“, což je mezera mezi budovami, které se vygenerují do jednoho bloku/řady podél cesty. K dispozici máme parametry „Min Gap“ a „Max Gap“ (jak lze vyčíst z obrázku 30), pomocí kterých stanovíme minimální a maximální velikost mezery – ta je generována náhodně v těchto mezích.

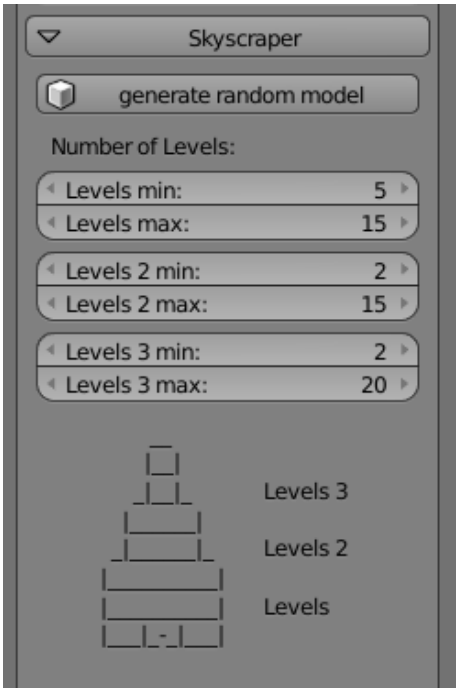
„Min Gap“ a „Max Gap“ musí být z intervalu $\langle 0, 1000 \rangle$. Jednotky jsou $BU/10$, což umožňuje jemnější rozdíly, než pouhé BU.



Obrázek 31: Různé nastavení velikosti mezery mezi budovami.

5.3.1 Skyscraper – nastavení pro mrakodrapy

V této podsekci je možné změnit nastavení pro generování náhodných 3D modelů mrakodrapů.



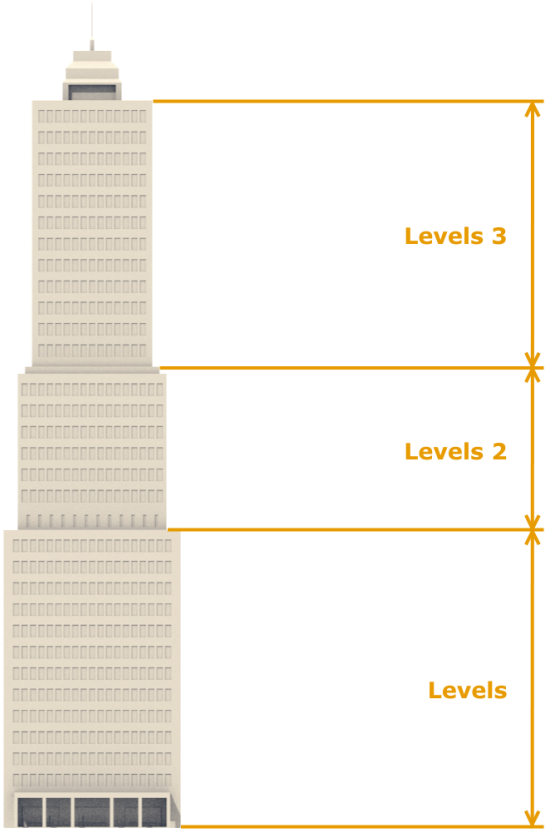
Obrázek 32: Skyscraper – nastavení pro mrakodrapy.

Hned na prvním místě máme k dispozici tlačítko „generate random model“, které, jak název napovídá, po kliknutí vygeneruje náhodný 3D model mrakodrapu podle parametrů uvedených níže.

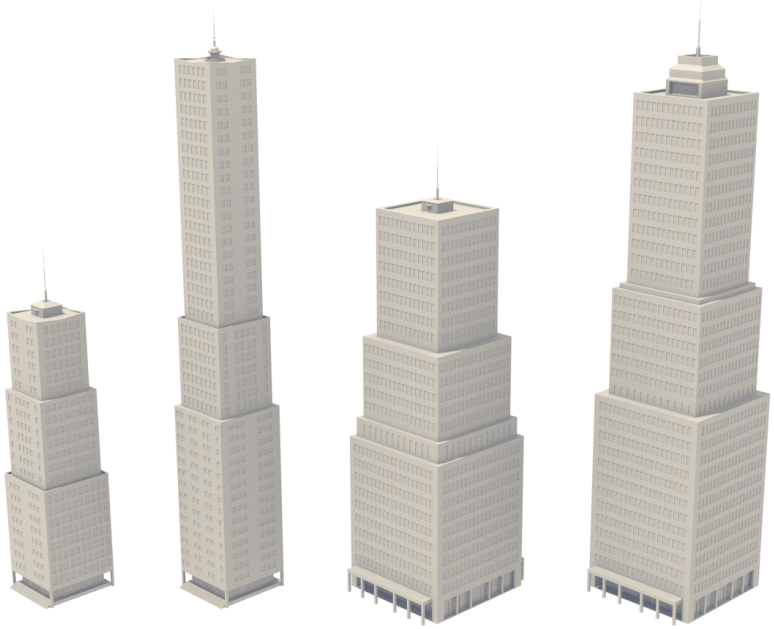
Vzhled částí mrakodrapu je stále generován náhodně (a uživatel ho nemůže ovlivnit), je však možné měnit počet jeho pater. K tomu slouží parametry „Levels min“ a „Leves max“, které určují minimální a maximální počet pater. Výsledný počet pater se poté náhodně zvolí v tomto rozmezí.

Mrakodrapy jsou rozděleny na tři části, které se směrem nahoru zužují. Každé z těchto částí je možné nastavit počet pater. „Levels 2 min“, „Levels 2 max“, „Levels 3 min“ a „Levels 3 max“ fungují podle stejného principu, který je popsán pro „Levels min“ a „Levels max“ v předchozím odstavci. Pod těmito parametry je zobrazeno zjednodušené schéma s názornými popisky. Pro lepší ilustraci je zde obrázek 33.

Na obrázku 34 je k vidění několik příkladů náhodně vygenerovaných mrakodrapů podle stejných parametrů.



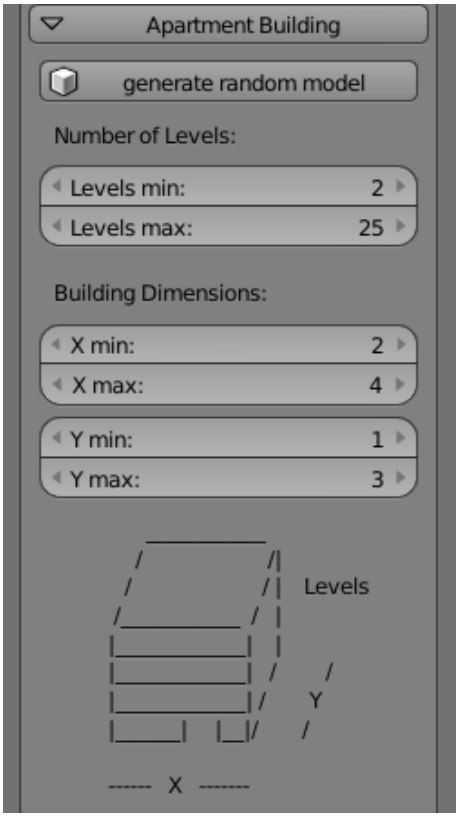
Obrázek 33: Parametry mrakodrapu.



Obrázek 34: Náhodně vygenerované mrakodrapy.

5.3.2 Apartment building – nastavení pro obytné domy

Zde je možné měnit nastavení parametrů pro generování náhodných 3D modelů obytných domů.



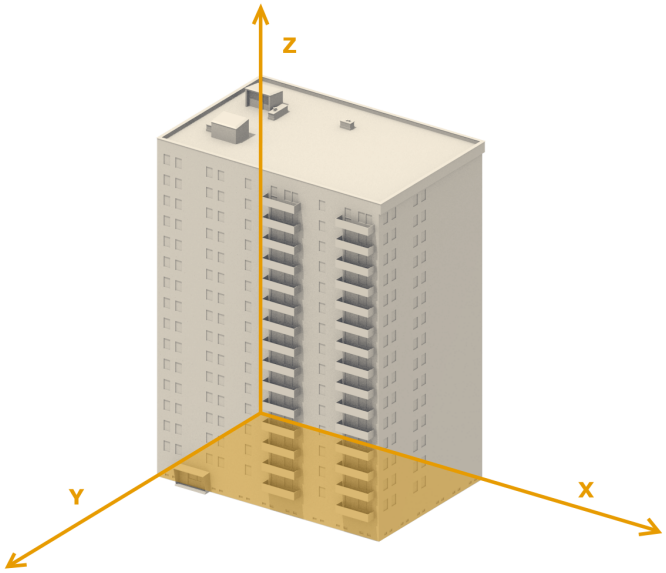
Obrázek 35: Apartment building – nastavení pro obytné domy.

Stejně jako u pod-sekce „skyscraper“ (nastavení pro mrakodrap), tlačítko na prvním místě umožňuje náhodně vytvořit samostatný 3D model budovy. Parametry „Levels min“ a „Levels max“ pro nastavení počtu pater fungují analogicky jako stejnojmenné parametry u mrakodrapu. Rozdíl je v tom, že obytné domy mají pouze jednu část s nastavitelným počtem pater.

Nové jsou zde parametry „X“ a „Y“, které určují minimální a maximální rozměry budovy na osách X a Y. Všechny budovy používají kartézskou soustavu souřadnic v třírozměrném prostoru, osa Z je v našem případě počet pater (výška budovy), jak je zřejmé z obrázku 36. Konečné hodnoty parametrů „X“ a „Y“ jsou opět náhodně vygenerované podle zadaných mezí „min“ a „max“.

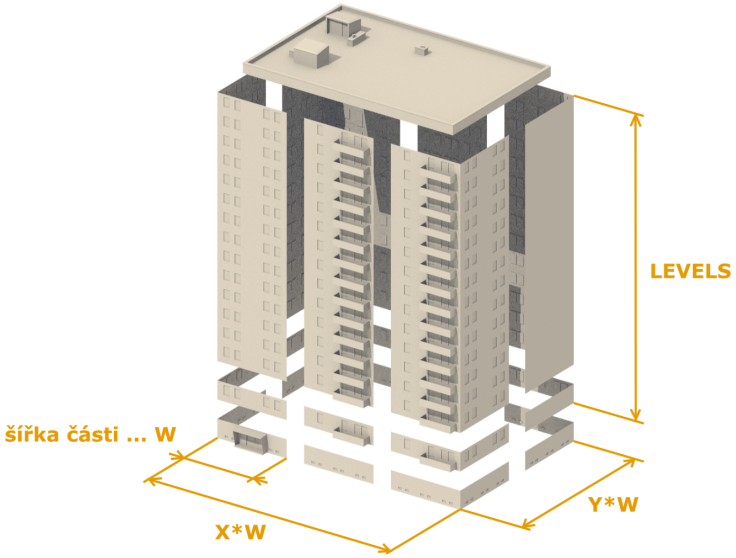
Důležité je také vysvětlit, jaké jednotky parametry „X“ a „Y“ mají. Ve skutečnosti se nejedná o pevně danou hodnotu, záleží totiž na šířce zvoleného 3D modelu části budovy[10] pro vytvoření stěny budovy – ne všechny tyto modely

[10]Všechny budovy jsou poskládány z menších částí, které se náhodně vybírají. Tento výběr nemůže uživatel nijak ovlivnit.



Obrázek 36: Kartézská soustava souřadnic u budov.

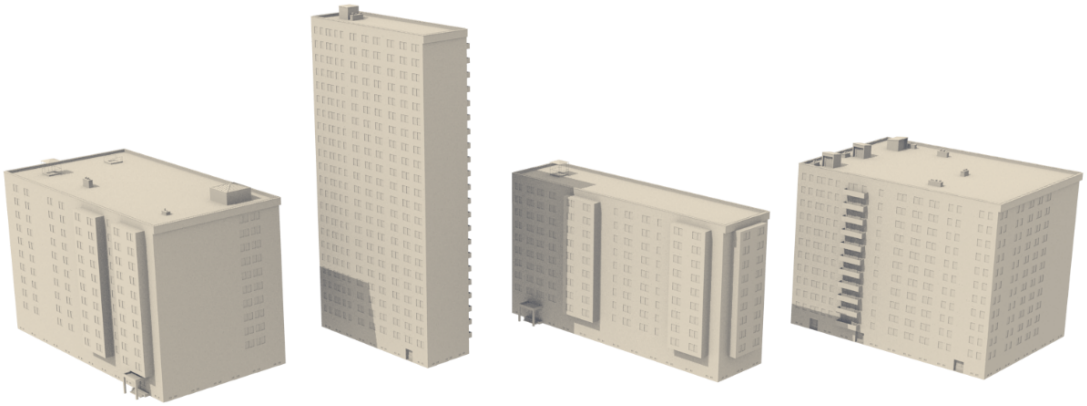
totiž mají stejnou šířku. Parametry „X“ a „Y“ jsou tedy násobky této šířky. Názorně vše ilustruje obrázek 37, kde šířku vybraného modelu označujeme W.



Obrázek 37: Parametry obyvatelných budov.

Stejně jako u nastavení mrakodrapu, i zde je pod parametry zobrazeno jejich zjednodušené schéma. Hodnoty parametrů musí být z intervalu $\langle 1, 1000 \rangle$. Doporučuji však používat hodnoty blízké původnímu nastavení, budovy tak vypadají více realisticky. Příliš veliké budovy se také nevejdou do vyhrazených míst podél cest (ta zůstanou prázdná, nebo budovy budou přečnívat)!

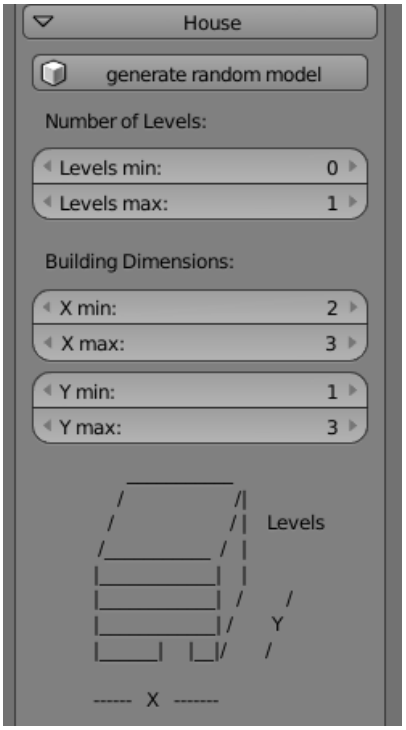
Na obrázku 38 je opět pár příkladů náhodně vygenerovaných budov. Pro všechny zobrazené modely bylo použito identické nastavení parametrů.



Obrázek 38: Náhodně vygenerované obytné budovy za použití stejného nastavení.

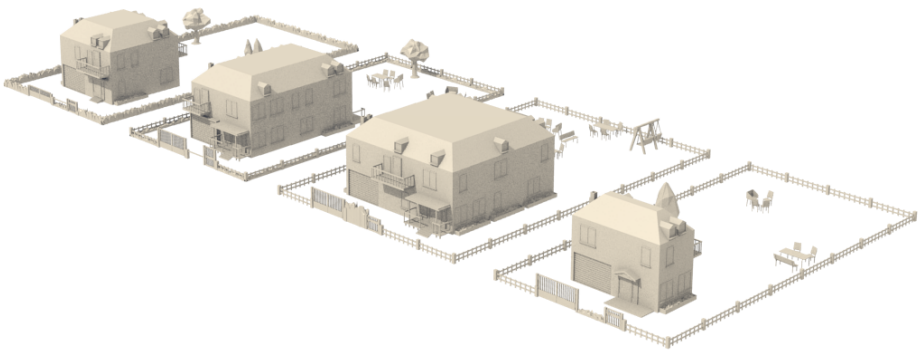
5.3.3 House – nastavení pro rodinné domy

Tato sekce obsahuje nastavení parametrů pro rodinné domy. Vše funguje analogicky jako nastavení pro obytné domy (všimněte si podobností obrázku 35 a obrázku 39). Jediný rozdíl je omezenější hodnota parametru „Y max“, která



Obrázek 39: Apartment building – nastavení pro rodinné domy.

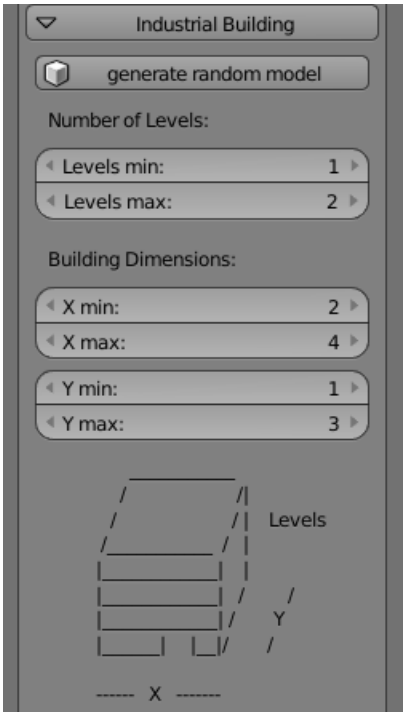
musí být z intervalu $\langle 1, 6 \rangle$. Důvodem je přednastavený rozměr budovy na ose „Y“, který počítá také se zahradou (vytvořená budova musí být uvnitř zahrady).



Obrázek 40: Náhodně vygenerované rodinné domy za použití stejného nastavení.

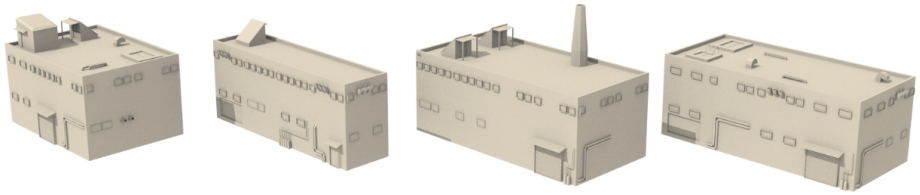
5.3.4 Industrial Building – nastavení pro průmyslové budovy

Zde jsou parametry pro průmyslové budovy. Stejně jako u rodinných domů, vše opět funguje analogicky jako nastavení pro obytné domy.



Obrázek 41: Apartment building – nastavení pro průmyslové budovy.

Na obrázku 42 je opět pár příkladů náhodně vygenerovaných budov (opět za použití stejného nastavení parametrů).



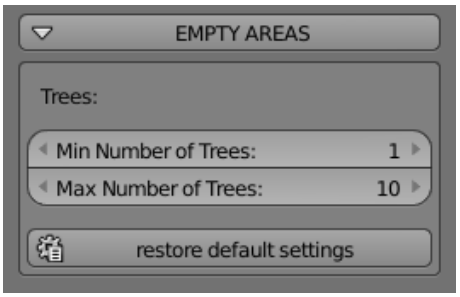
Obrázek 42: Náhodně vygenerované průmyslové budovy za použití stejného nastavení.

5.3.5 Obnovení výchozího nastavení

Po kliknutí na tlačítko „restore default settings“ se obnoví výchozí nastavení všech parametrů sekce „BUILDINGS“, tedy i všech parametrů budov.

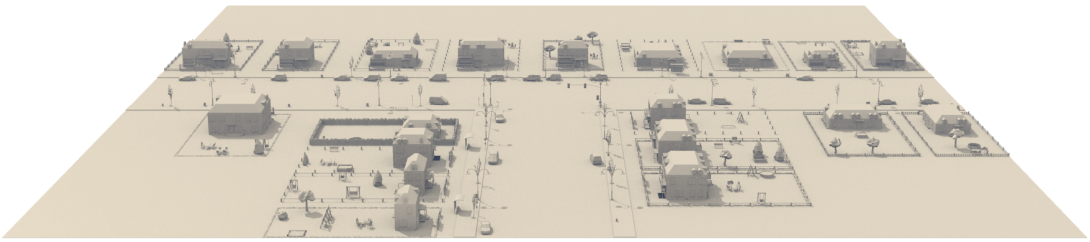
5.4 EMPTY AREAS – Prázdná místa v sektoru

Poslední sekce v ovládacím panelu je „EMPTY AREAS“ – nastavení pro prázdná místa sektoru.



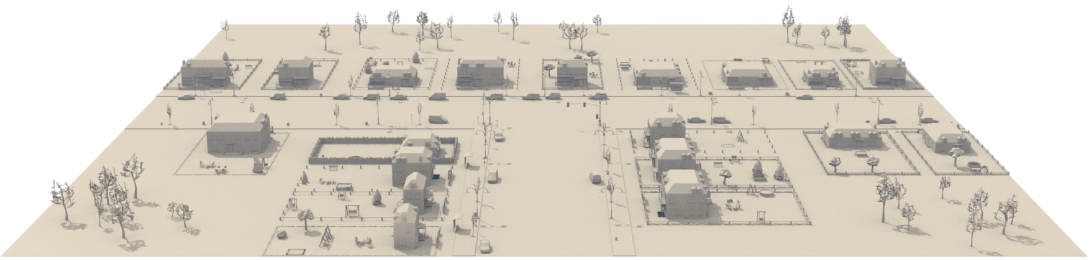
Obrázek 43: Ovládací panel – EMPTY AREAS

Generátor měst si ve své základní podstatě vystačí pouze s generováním cest a budov kolem nich. Vzniklé sektory však mohou působit poněkud „řídce“, jestliže okolo budov bude příliš mnoho prázdných míst (viz obrázek 44).



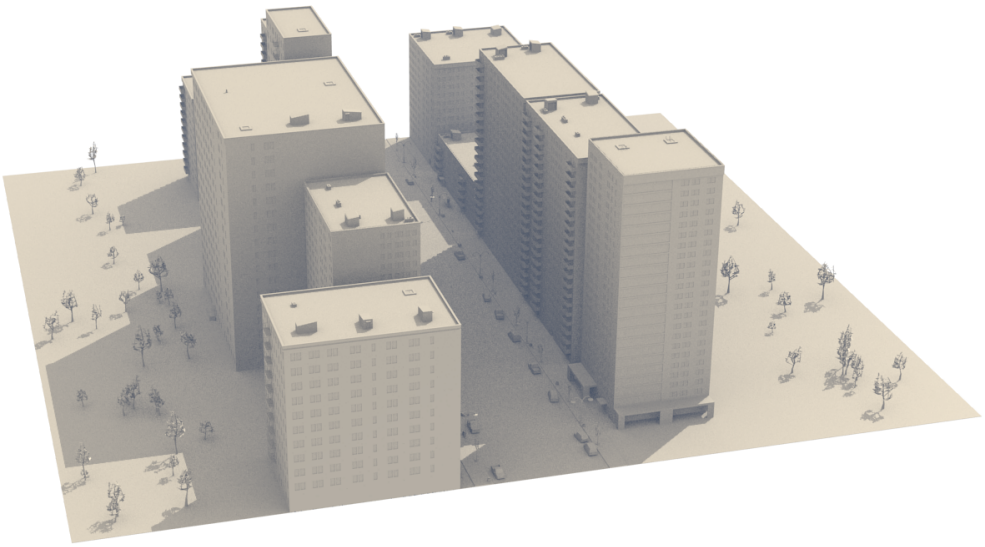
Obrázek 44: Sektor s mnoha prázdnými místy

Tento nedostatek odstraňuje právě tato sekce. Zde můžeme pomocí parametrů „Min Number of Trees“ a „Max Number of Trees“ nastavit minimální a maximální počet stromů, které se použijí pro vyplnění prázdných čtverců sektoru. Když srovnáte obrázek 44 a obrázek 45, je vidět, že prázdná místa jsou nyní za-



Obrázek 45: Sektor s vyplněnými prázdnými místy

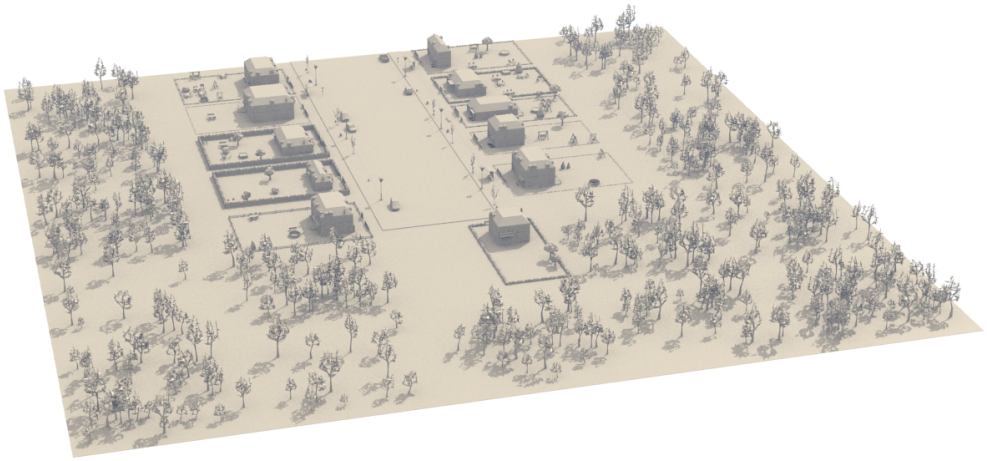
plněna stromy. Zejména u sektorů s rodinnými domy se jedná o důležitý doplněk (vznikají tak „lesíky“). Stromy se však hodí i do sektorů s obytnými domy, či



Obrázek 46: Sektor s obytnými domy a s vyplněnými prázdnými místy

do centra města (zde se jich pravděpodobně nevygeneruje mnoho, protože tudy většinou vede mnoho cest a prázdných míst tedy není tolik, což je žádoucí).

Hodnota parametru „Number of Trees“ je opět náhodně generována podle zadaných mezí. „Min Number of Trees“ a „Max Number of Trees“ musí být z intervalu $\langle 0, 100 \rangle$ a platí pro právě jeden čtverec sektoru.



Obrázek 47: Více stromů vytváří lesíky

5.4.1 Obnovení výchozího nastavení

Kliknutím na tlačítko „restore default settings“ lze obnovit výchozí nastavení všech parametrů sekce „EMPTY AREAS“.

5.5 GENERATE 3D CITY – Tlačítko pro vytvoření 3D modelu města

Nyní se dostáváme k hlavnímu tlačítku tohoto ovládacího panelu. Po kliknutí se začne generovat 3D model města podle zadaných parametrů.

Obrázek 48: Ovládací panel – GENERATE 3D CITY

Bere tedy v úvahu nastavení sektorů (ze kterých ve výsledku složí město), nastavení cest a jejich příslušenství (zaparkovaná auto, odpadkové koše apod.), nastavení mezery mezi budovami a mezní parametry budov a v neposlední řadě také počet stromů, kterými vyplní prázdná místa.

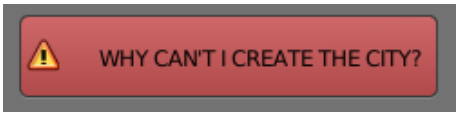
5.5.1 Průběh generování města

Bohužel se zde projevuje další nevýhoda Blenderu. Jakmile totiž začne na něčem pracovat, chová se stejně, jaky když „zamrzne“. To je velice nežádoucí chování, protože uživatel si nikdy nemůže být jistý, jestli pracuje, nebo ne. Uživatel také nemá možnost proces přerušit, ani sledovat jeho průběh.[11]

Obzvláště při generovaní většího města je možné, že Blender dlouho nebude reagovat (dokud nedokončí město). Abych alespoň částečně napravil tento problém, musel jsem narušit automatické vykreslování 3D scény v Blenderu tak, aby se scéna překreslila vždy po dokončení 3D modelu sektoru. Při generování města se tedy postupně objevují nové sektory.

5.5.2 Chybové hlášení

Existuje také možnost, že namísto textu „CREATE 3D CITY“, bude tlačítko indikovat chybu některého z parametrů a zobrazovat text „WHY CAN’T I CREATE THE CITY?“.



Obrázek 49: Chyba při zadávání parametrů

Tento případ může ve skutečnosti nastat pouze v jediném případě – při špatně zadaných volitelných vstupech prvního sektoru, o kterých jsme se bavili dříve.

[11] Tento problém jsem částečně obešel vypisováním informací do systémové konzole, nedá se ovšem počítat s tím, že běžný uživatel bude mít konzoli zobrazenou. Jedná se spíše o pomůcku při programování.

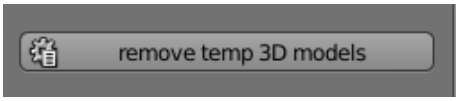
Pro více informací se tedy můžete vrátit na stranu 27. Po opravení vstupů na přijatelné hodnoty (nebo vypnutí možnosti použít vlastní vstupy) se tlačítko promění zpět, a opět umožní vytvořit 3D model města.

5.6 Remove temp 3D models – Tlačítko pro vymazání dočasných 3D modelů

Generátor měst potřebuje pro svoji práci 3D modely základních částí budov, cest a podobně. Všechny tyto modely si sám importuje při stisknutí tlačítka pro generování města (nebo budovy či části cesty). Tlačítko je původně „vybledlé“

Obrázek 50: Indikace, že v projektu nejsou další objekty ke smazání

(viz obrázek 50; indikuje tak, že v projektu žádné tyto modely nejsou), jakmile je však jeho funkce možná, je opět k dispozici (51, plně viditelné). Po kliknutí na toto tlačítko se tedy smažou všechny pomocné 3D modely. Doporučuji je mazat až

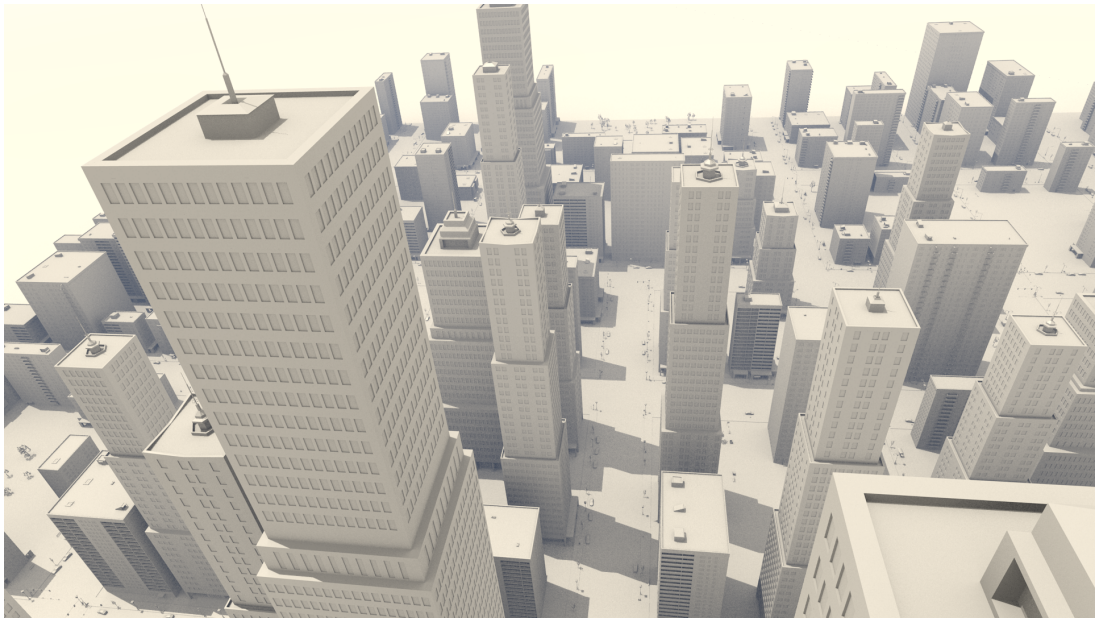


Obrázek 51: Tlačítko pro vymazání dočasných 3D modelů

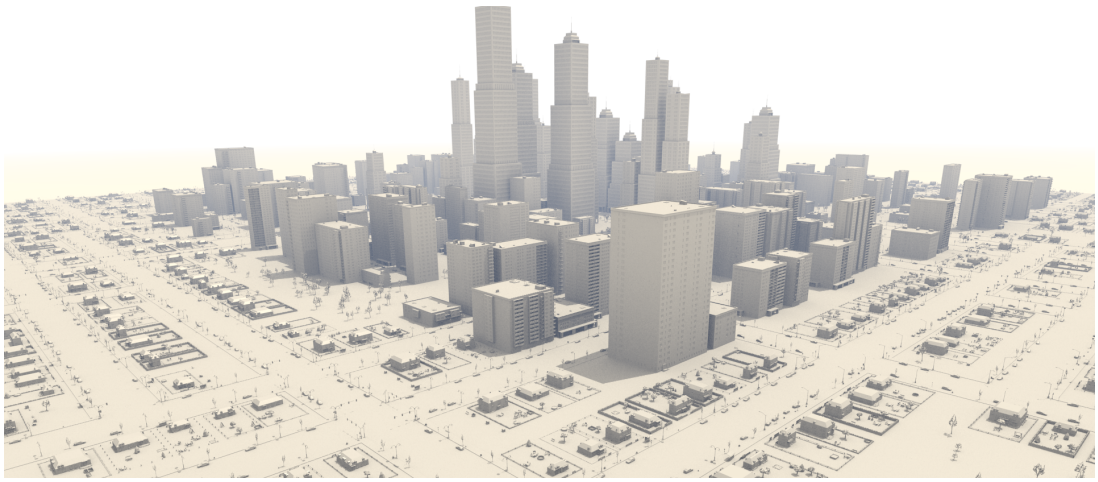
po vytvoření celého (konečného) města, aby se nemusely opakovaně importovat. Nejedná se sice o příliš velké zrychlení, ale minimálně se tak pohodlněji pracuje. Pokud tyto modely ovšem nechcete mazat vůbec, je to také možné. Na následnou práci s městem už nemají žádný vliv.

5.7 Ukázky náhodně vygenerovaných měst

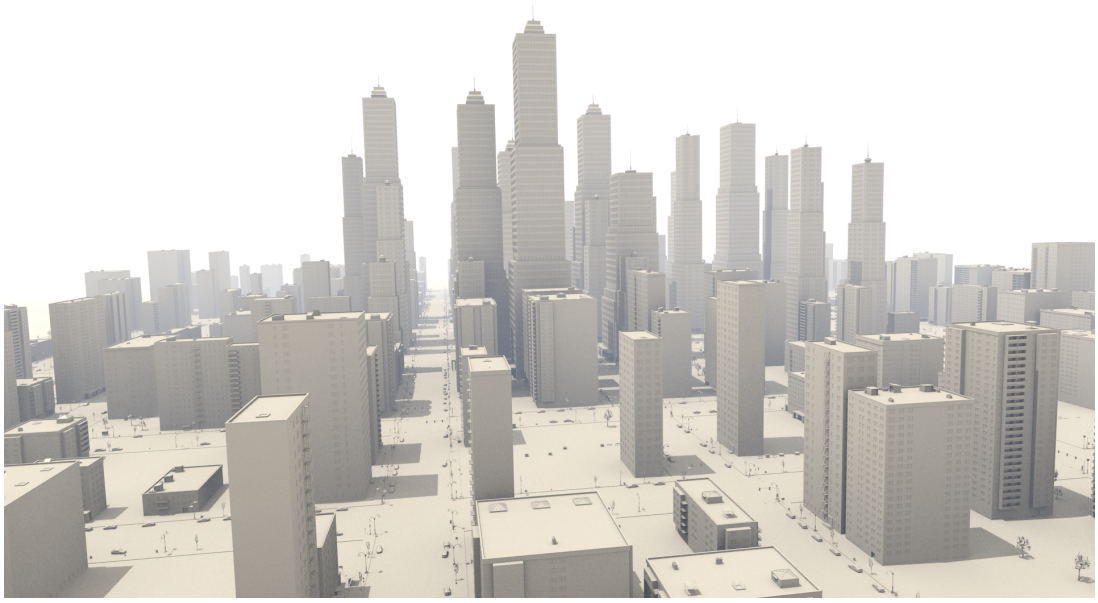
V této kapitole jsou k vidění ukázky měst, které byly vygenerovány pomocí tohoto generátoru bez jakýchkoli dalších úprav.



Obrázek 52: Ukázka náhodně vygenerovaného města 1

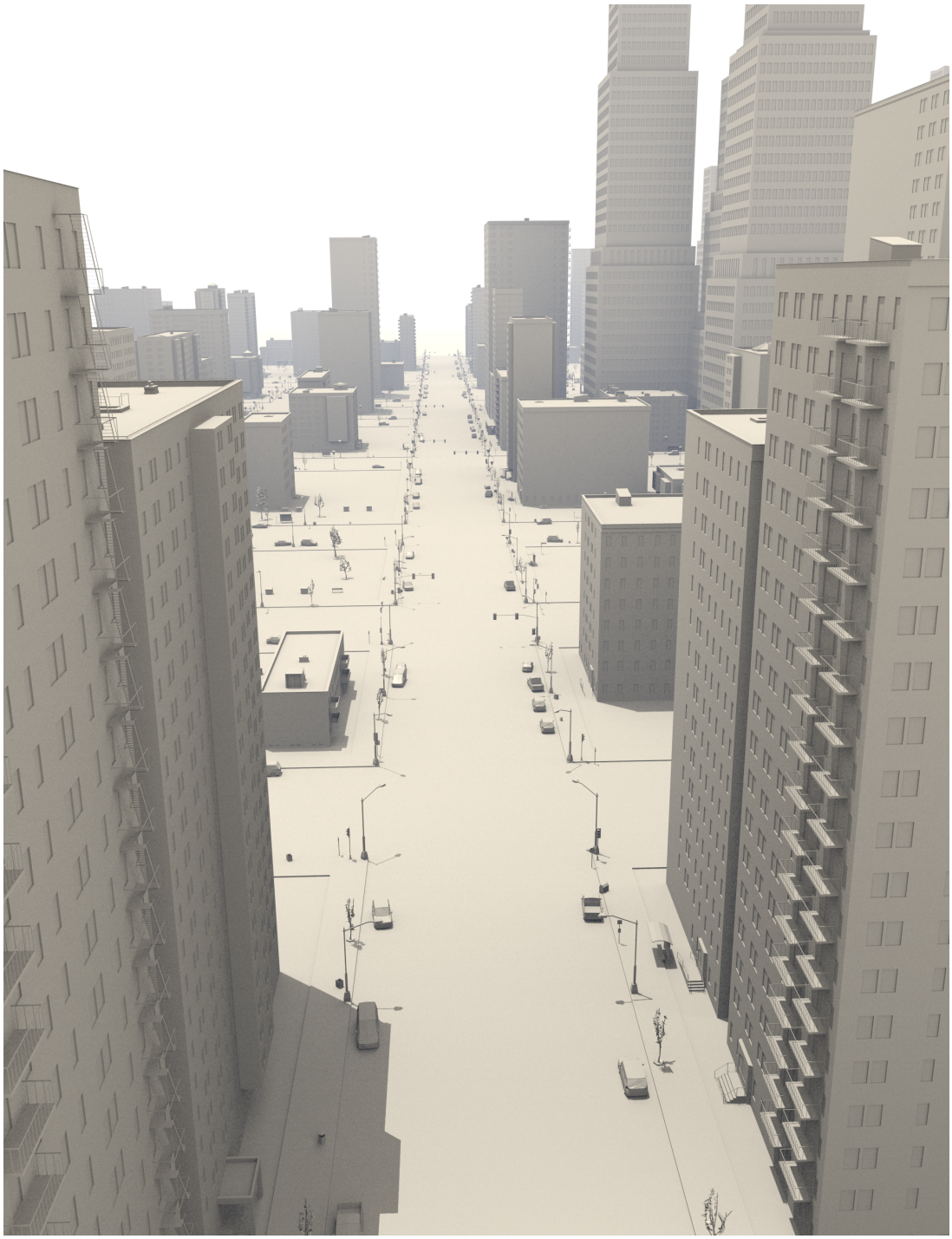


Obrázek 53: Ukázka náhodně vygenerovaného města 2



Obrázek 54: Ukázka náhodně vygenerovaného města 3

Obrázek 55: Ukázka náhodně vygenerovaného města 4



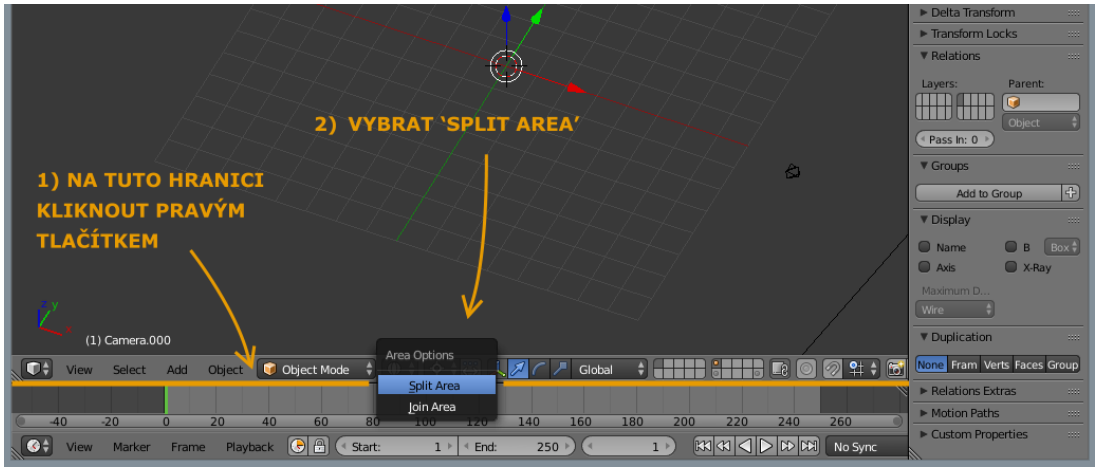
Obrázek 56: Ukázka náhodně vygenerovaného města 5

6 Popis použitých algoritmů

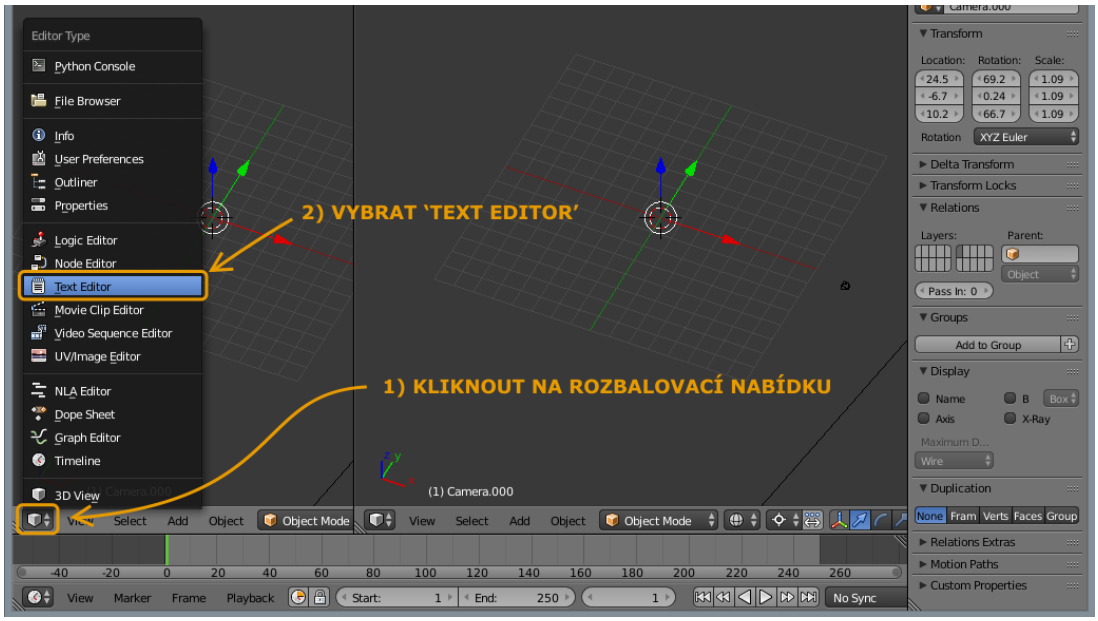
V této kapitole se podíváme, jak generátor měst tvoří jednotlivé budovy, cesty, sektory a nakonec celé město. Použitý jazyk je Python.

6.1 Vývojové prostředí

Jako vývojové prostředí lze použít textový editor uvnitř Blenderu, který je přímo určen pro psaní Python skriptů. Využijeme možnosti Blenderu poskládat si



Obrázek 57: Rozdělení 3D pohledu GUI

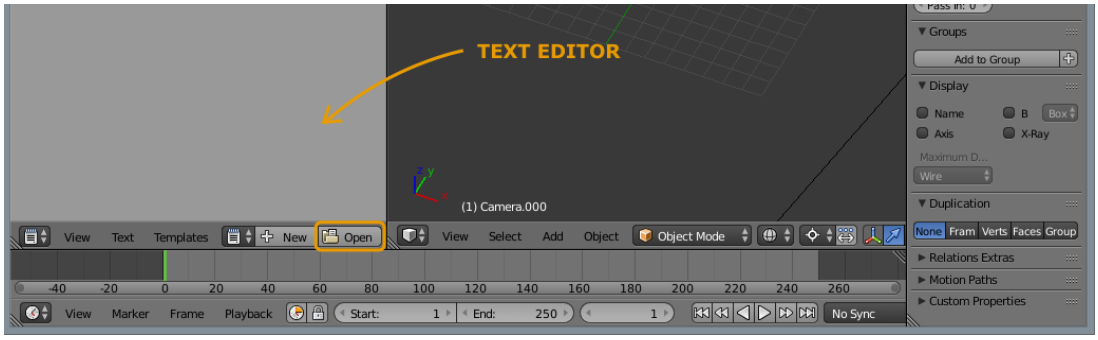


Obrázek 58: Změna typu části GUI na „Text Editor“

vlastní rozložení GUI. Kliknutí pravým tlačítkem myši na libovolný horizontální

okraj 3D pohledu vyvolá nabídku (viz obrázek 57), ze které vybereme možnost „Split Area“.

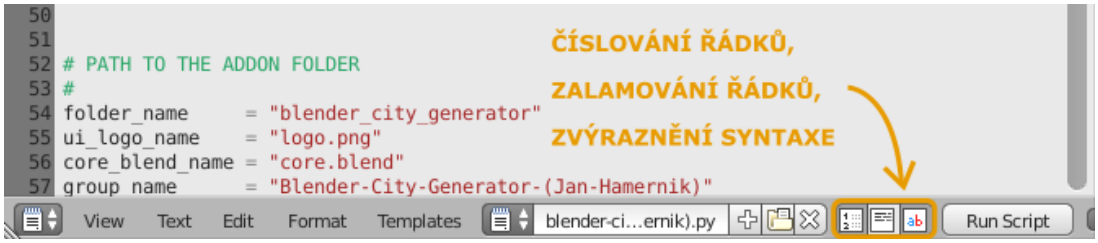
Myš posuneme na místo, kde chceme rozdělující hranici, a potvrdíme levým tlačítkem myši (pravým tlačítkem se dá operace zrušit). Nyní nově vzniklému oknu vlevo nastavíme „editor type“ na „Text Editor“ (viz obrázek 58). Nyní



Obrázek 59: Text Editor

už jen stačí otevřít skript pro generátor měst. Klikneme na tlačítko „Open“ (zvýrazněné na obrázku 59), najdeme příslušný skript („blender-city-generator-(jan-hamernik).py“) a otevřeme jej. Skript se zobrazí v „Text Editoru“ jako čistý text.

Pro lepší čitelnost můžeme zapnout zvýrazňování syntaxe a číslování řádků (je také možné zapnout zalamování řádků, osobně mi ale kód potom přijde nepřehledný, takže tuto možnost nepoužívám), jak je vidět na obrázku 60.



Obrázek 60: Text Editor

Pomocí tlačítka „Run Script“ pak můžeme skript spustit. V našem případě se po spuštění přidá do GUI Blenderu ovládací panel pro generátor měst.

Klávesovou zkratkou „Ctrl + T“ s kurzorem myši nad „Text Editorem“ lze vyvolat panel s dalším nastavením (například velikost písma, hledání a nahrazení slov v textu a další), ten však nyní není potřeba.

6.2 Pomocné funkce

Před prací na samotném generátoru měst bylo zapotřebí definovat si řadu obecných funkcí, které později využiji. V této kapitole uvedu pouze pár vybraných funkcí, začneme tedy s tou nejdůležitější, což je funkce pro importování 3D modelů.

6.2.1 Funkce pro importování 3D modelů do Blenderu

Bez této funkce by se generátor měst neobešel. Jak jsem již zmínil dříve v textu, budovy jsou poskládány z menších 3D modelů – a ty je právě potřeba vložit do Blenderu před samotným generováním měst. K tomu slouží tato funkce:

```
1  def import_models():
2   try:
3     # jestliže už jsou objekty vloženy, ukonči svoji činnost
4     for group in bpy.data.groups:
5       if group.name == group_name:
6         return
7
8     # cesta k .blend projektu s požadovanými 3D modely
9     filepath = os.path.join(addon_folder , core_blend_name)
10
11    # načti data z projektu
12    with bpy.data.libraries.load(filepath) as (data_src, data_dst):
13      data_dst.groups = [group_name]
14
15    # pole určující na které vrstvy se objekty vloží
16    # (vloží se na poslední vrstvu)
17    layer_list = [ False, False, False, False, False,
18       False, False, False, False, False,
19       False, False, False, False, False,
20       False, False, False, False, True]
21
22    # projdi data a načti je na zvolené vrstvy
23    for group in data_dst.groups:
24      instance = bpy.data.objects.new(group_name, None)
25      instance.dupli_type = 'GROUP'
26      instance.dupli_group = group
27      bpy.context.scene.objects.link(instance)
28      instance.layers = layer_list
29
30  except:
31    print("An error occured while import temporary 3D models!")
```

Zdrojový kód 1: Funkce pro importování 3D modelů do Blenderu

Nejedná se o typické importování 3D modelů ve formátu .obj nebo .3ds. Modely totiž nejsou z Blenderu exportovány, ale jsou uloženy v .blend projektu, který je umístěn v „addons“ složce samotného Blenderu.

Blender má funkci „append“, která umožňuje „spojit“ vybrané části určitého .blend (Blender) projektu s aktuálním projektem. Funkce „import_models“ pracuje na podobném principu.

6.2.2 Funkce pro zjištění rozměrů objektu

Možnost zjistit rozměry zvoleného objektu je rovněž nezbytná pro správnou funkci generátoru měst. Každý objekt má vlastní tzv. „mesh“, který má in-

```
1  def get_mesh_dimensions(mesh):
2    # deklarace proměnných
3    x_right = 0
4    y_depth = 0
5    z_top = 0
6
7    # počátek objektu je vždy v nule (důležité)
8    x_neg = 0
9    y_neg = 0
10   z_neg = 0
11
12   # projdi vrcholy meshe a přenastav proměnné
13   for vertex in mesh.vertices:
14     x = vertex.co[0]
15     y = vertex.co[1]
16     z = vertex.co[2]
17     if x > x_right:
18       x_right = x
19     if y > y_depth:
20       y_depth = y
21     if z > z_top:
22       z_top = z
23     if x < x_neg:
24       x_neg = x
25     if y < y_neg:
26       y_neg = y
27     if z < z_neg:
28       z_neg = z
29
30   # vypočítej a vrat' rozměry
31   return [abs(x_neg) + x_right, abs(y_neg) + y_depth, abs(z_neg) +
         z_top]
```

Zdrojový kód 2: Funkce pro zjištění rozměrů meshe

formace o tvaru objektu (uchovává souřadnice bodů, hran a stěn). Pomocí meshe tedy zjistíme rozměry objektu od jeho počátku. Ten je vždy v bodě $[0, 0, 0]$ (důležité!).

V Pythonu je vestavěná funkce „dimensions“, která pro zadaný objekt vrátí jeho rozměry. Já však tuto funkci nepoužívám, protože, jak jsem se již zmínil,

```
def get_object_dimensions(obj):
  # jestliže v obj není uložen objekt, vrat' nulové rozměry
  if obj is None:
    return [0,0,0]

  # v opačném případě vrat' rozměry jeho meshe
  return get_mesh_dimensions(obj.data)
```

Zdrojový kód 3: Funkce pro zjištění rozměrů objektu

pro funkci generátoru měst potřebuji počítat rozměry vždy z počátku souřadnic ($[0, 0, 0]$). Dalším důvodem je fakt, že se tato funkce dá zavolat pouze s objektem. Některé funkce však pracují pouze s meshem, kdy zjišťují jeho rozměry ještě před vytvořením samotného objektu.

6.2.3 Funkce pro vytváření objektů

Neméně důležité je také vytvářet objekty. Například podle zadaného meshe, či pouze podle jeho jména.

```
# vytvoř objekt ze zadaného meshe
def create_object_from_mesh(mesh, final_obj_name):
  # vytvoři kopii meshe
  mesh_copy = mesh.copy()
  mesh_copy.name = "mesh_copy"

  # vytvoř nový objekt (z kopie meshe)
  obj = bpy.data.objects.new(final_obj_name, mesh_copy)

  # spoj objekt se scénou
  bpy.context.scene.objects.link(obj)

  # vrat' vytvořený objekt
  return obj


# vytvoř objekt z meshe zadaného jménem
def create_object_from_mesh_name(mesh_name, final_obj_name):
  # získej mesh podle zadaného jména
  mesh = bpy.data.meshes[mesh_name]

  # vytvoř z tohoto meshe objekt, a vrat' jej
  return create_object_from_mesh(mesh, final_obj_name)
```

Zdrojový kód 4: Funkce pro vytváření objektů

Jak si můžete všimnout, funkce „create_object_from_mesh“ vytváří kopii meshe, místo aby jej použila přímo. Pokud by totiž funkce vytvořila objekt přímo

z předaného meshe, všechny jeho transformace skrze takto vytvořený objekt by se promítly na všechny objekty, které tento mesh používají. V principu by se tedy všechny takové objekty odkazovaly na jeden mesh, a vzájemně se tak „rozbíjely“.

Pro zjednodušení algoritmů je výhodné místo „Null“ (a neustálého testování na „Null“) předávat prázdný objekt. Kód 5 ukazuje definici funkce, která ho umožňuje vytvořit.

```
1  def create_blank_object():
2    # vytvoř nový mesh (je prázdný)
3    mesh = bpy.data.meshes.new("empty")
4
5    # vytvoř z něj nový objekt
6    obj = bpy.data.objects.new("empty", mesh)
7
8    # spoj objekt se scénou
9    bpy.context.scene.objects.link(obj)
10
11   # vrat' vytvořený objekt
12   return obj
```

Zdrojový kód 5: Funkce pro vytvoření prázdného objektu

Jelikož je celé město generováno náhodně, pro úplnost zde ještě uvádím kód pro vytvoření náhodných objektů.

```
1  # náhodně vyber mesh
2  def get_random_mesh(meshes):
3    # zjisti kolik meshů pole meshes obsahuje
4    rand_limit = len(meshes)-1
5
6    # vyber mesh na náhodném indexu
7    return meshes[randint(0,rand_limit)]
8
9
10 # vytvoř objekt z náhodně vybraného meshe
11 def create_object_from_random_mesh(meshes, final_obj_name):
12   # náhodně vyber mesh (funkce výše)
13   mesh = get_random_mesh(meshes)
14
15   # vytvoř objekt z meshe a vrat' jej
16   return create_object_from_mesh(mesh, final_obj_name)
```

Zdrojový kód 6: Funkce pro vytváření náhodných objektů

6.2.4 Funkce pro transformaci objektů

Aby bylo možné z importovaných 3D modelů poskládat složitější modely (budovy, cesty, ...), je nutné definovat také funkce pro jejich transformace. Jelikož se jedná o poměrně základní posloupnosti operací, uvedu zde pro příklad pouze dvě funkce.

```
1  # posuň objekt podle rozměrů
2  def shift_object_by_dimensions(obj, shift_vector):
3    # jestliže obj je None, vrat' obj bez transformací
4    if obj is None:
5      return obj
6
7    # zjisti rozměry objektu obj
8    dim = get_object_dimensions(obj);
9
10   # přičti k aktuální pozici násobek vektoru posunutí s rozměry obj
11   obj.location.x += shift_vector[0] * dim[0]
12   obj.location.y += shift_vector[1] * dim[1]
13   obj.location.z += shift_vector[2] * dim[2]
14
15   # vrat' objekt
16   return obj
17
18
19 # posuň objekt podle zadané hodnoty
20 def shift_object_by_value(obj, shift_vector):
21   # přičti k aktuální pozici vektor posunutí
22   obj.location.x += shift_vector[0]
23   obj.location.y += shift_vector[1]
24   obj.location.z += shift_vector[2]
25
26   # vrat' objekt
27   return obj
```

Zdrojový kód 7: Funkce pro transformaci objektů

První funkce („shift_object_by_dimensions“) nám umožňuje posunout objekt o násobek jeho rozměrů, což je výhodné pro posun různě velikých modelů tak, aby například vytvořili spojitou stěnu budovy. Díky této funkci tedy snadno poskládáme budovy tak, jak jsme již mohli vidět na obrázku 37, na straně 40.

Druhá funkce („shift_object_by_value“) pak posunuje objekt o zadanou hodnotu. V obou případech hodnoty zadáváme jako tříprvkový vektor (posun po ose X, Y a Z).

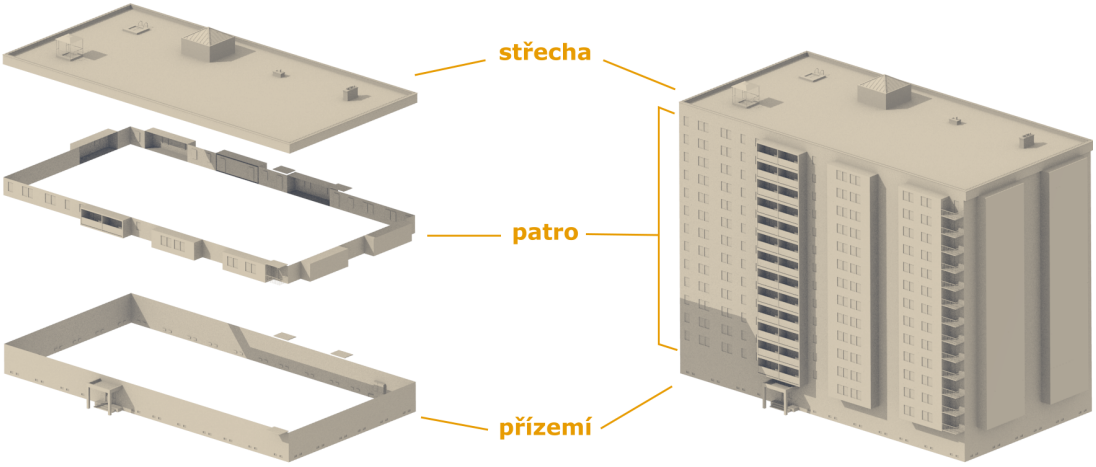
6.3 Generování 3D modelů budov

Jelikož jsou funkce pro generování budov rozsáhlé (a jejich rozdělení by značně zvýšilo nepřehlednost kódu), uvedu zde pouze funkce pro generování obytných budov („apartment building“). Uvedu také pouze několik zástupců těchto funkcí, ostatní však pracují na stejných principech. Pro zjednodušení některé části funkcí nahradím pseudokódem.

```
1  def create_apartment_level(building_meshes, x, y, no_balconies):
2    # získej relevantní meshe pro části budovy:
3    meshes_apartments = find_building_level_meshes_with_specific_name(
         building_meshes, mesh_name_apartments)
4    meshes_empty = # OBDOBNĚ
5    meshes_balconies = []
6
7    # deklaruj pole pro náhodné indexy stěn s balkony:
8    random_ent_indices = []
9
10   # pokud budou použity balkony:
11   if no_balconies > 0:
12     # ZJISTI RELEVANTNÍ MESHE "meshes_balconies"
13     # UMÍSTI DO "random_ent_indices" NÁHODNÉ INDEXY
14
15   # vygeneruj 3D model:
16   level_objs = []
17
18   # vytvoř stěnu "X"
19   for i in range(0, x):
20     if i in random_ent_indices:
21       # vytvoř balkon
22       obj = create_object_from_random_mesh(meshes_balconies, "
             temp_apartments_balcony")
23     else:
24       # vytvoř stěnu s okny
25       obj = create_object_from_random_mesh(meshes_apartments, "
             temp_apartments")
26     wall_transform(obj, i, x, y, 0) # posuň stěnu na místo
27     level_objs.append(obj) # ulož model do pole
28
29   # VYTVOŘ STĚNU "Y" (OBDOBNĚ JAKO "X")
30   # VYTVOŘ STĚNU "X-Neg"
31   # VYTVOŘ STĚNU "Y-Neg"
32
33   # spoj vytvořené objekty do jednoho objektu, a ten vrat':
34   return join_objects_to_new(level_objs, "temp_level_apartments")
```

Zdrojový kód 8: Funkce pro vytvoření jednoho patra obytné budovy

Funkce „create_apartment_level“ (definovaná v kódu 8) slouží pro vytvoření 3D modelu jednoho patra obytné budovy, pro ilustraci si prohlédněte obrázek 61.



Obrázek 61: Generované části obytné budovy

Funkce pro vytvoření střechy je rozdělená na dvě části. První část (funkce v kódu 9) „roztáhne“ importovaný model střechy na potřebné rozměry, a druhá část (funkce v kódu 10) střechu náhodně zaplní „příslušenstvím“, jako jsou vstupy na střechu, ventilace apod.

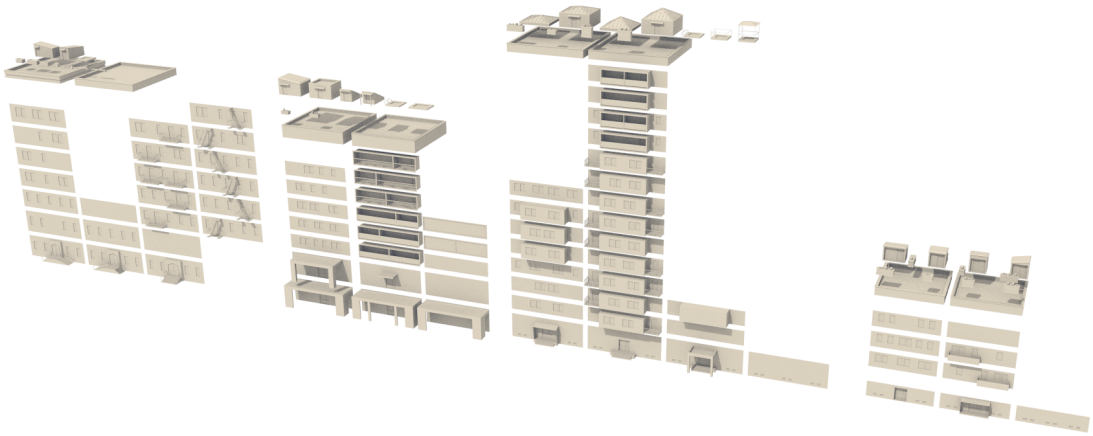
```
1  def create_roof_level(building_meshes, x, y):
2    # získej relevantní meshe pro střechu:
3    meshes_roof = find_building_level_meshes_with_specific_name(
         building_meshes, mesh_name_roof)
4
5    # vytvoř objekt z náhodně vybraného meshe a zjisti jeho šířku:
6    roof_obj = create_object_from_random_mesh(meshes_roof, "
         temp_roof_level")
7    width = get_object_dimensions(roof_obj)[0]
8
9    # roztáhni mesh objektu na požadované rozměry x, y:
10   for vertex in roof_obj.data.vertices:
11     if vertex.co.x > width/2: vertex.co.x += (x-1)*width
12     if vertex.co.y > width/2: vertex.co.y += (y-1)*width
13
14   # vrať objekt střechy:
15   return roof_obj
```

Zdrojový kód 9: Funkce pro vytvoření střechy budovy

```
def create_roof(building_meshes, x, y):
  # vytvoř 3D model střechy:
  roof_level = create_roof_level(building_meshes, x, y)
  roof = [roof_level]

  meshes_roof_entrance = # RELEVANTNÍ MESHE VSTUPŮ NA STŘECHU
  meshes_air_vents = # RELEVANTNÍ MESHE VENTILACÍ

  # v cyklech vyber náhodný objekt, a náhodně ho transformuj:
  for i in range (0, x):
    for j in range (0, y):
      # zaručené vložení vstupu na střechu (i == 0), dále náhodně
      if i == 0 or randint(0,20) == 0:
        obj = create_object_from_random_mesh(meshes_roof_entrance, "
            temp_roof_entrance")

      # pravděpodobné vložení "ničeho"
      elif randint(0,6) > 1:
        obj = create_blank_object()

      # jinak vložení ventilace
      else:
        obj = create_object_from_random_mesh(meshes_air_vents, "
            temp_air_vent")

      # ulož objekt do pole a přesuň počátek objektu do jeho středu:
      roof.append(obj)
      center_obj_origin(obj)

      # náhodná rotace
      rand_rot = randint(0,4)*90
      rotate_object_degrees(obj, (0,0,rand_rot))

      # pozice
      step = get_object_dimensions(roof_level)[0]/x
      shift_object_by_value(obj, [step/2 + i*step, step/2 + j*step,
          0])

      # náhodný posun
      rand_x = float(randint(-10,10)/1000)
      rand_y = float(randint(-10,10)/1000)
      shift_object_by_value(obj, [rand_x, rand_y, 0])

  # vrat' objekty spojené do jednoho objektu
  return join_objects_to_new(roof, "temp_roof")
```

Zdrojový kód 10: Funkce pro zaplnění střechy budovy

Další na řadě je finální funkce (zdrojový kód 11), která pomocí předešlých funkcí vytvoří 3D model budovy podle zadaných parametrů. Mezi ty patří například „design_index“, se kterým uživatel nemá možnost manipulovat. Jak jsem se již několikrát zmínil, budova je poskládána z menších 3D modelů. Jelikož tyto 3D modely mohou mít různé rozměry, je potřeba je roztřídit do skupin, které jsou určeny právě „design indexem“.



Obrázek 62: Příklad 3D modelů pro tvorbu obytných budov

Na obrázku 62 jsou k vidění příklady 3D modelů, ze kterých se tvoří obytné budovy. Každá ze skupin 3D modelů zastupuje právě jeden „design index“. Počet těchto indexů je uložen v proměnné v kódu, takže algoritmy přesně vědí, z kolika takových skupin mohou modely vybírat. Je také snadné přidat nový design budovy. Stačí vytvořit nové 3D modely[12] a přepsat v kódu počet designů pro daný typ budovy.

Je také nutné vytvořit pro příslušný design minimálně jeden 3D model pro každou potřebnou část budovy – v tomto případě tedy musíme vymodelovat tyto části:

1. vchodové dveře (ground entrance),
2. stěnu bez vchodových dveří (ground),
3. okna (apartments),
4. balkon (apartments balcony),
5. stěnu bez oken (apartments empty),
6. střechu (roof),
7. ventilace (air vents),
8. vchod na střechu (roof entrance).

[12]Názvy meshů nových modelů ale musí odpovídat určitému formátu, který model zařadí pod příslušný „design index“.

```
def create_apartment_building(design_index, no_levels=1, x=2, y=1,
    no_entrances=1, no_balconies=2):

  building_meshes = # RELEVANTNÍ MESHE BUDOVY
  building = []

  #vytvoř základ budovy (vchodové patro):
  ground = create_ground_level(building_meshes, x, y, no_entrances)
  building.append(ground)

  # vytvoř patra budovy:
  if no_levels > 0:
    # vytvoř jedno patro:
    apartment_level = create_apartment_level(building_meshes, x, y,
        no_balconies)

    # funkce "create_apartments" patro navrší na potřebný počet:
    apartments = create_apartments(apartment_level, no_levels)

    # ulož patra do modelů budovy a posuň je nad vchodové patro
    building.append(apartments)
    shift_object_by_value(apartments, (0,0, get_object_dimensions(
        ground)[2]))

  # vytvoř střechu budovy:
  roof = create_roof(building_meshes, x, y)
  building.append(roof)

  # posuň střechu:
  shift_object_by_value(roof, (0,0, get_object_dimensions(ground)
      [2]))
  if no_levels > 0:
    shift_object_by_value(roof, (0,0, get_object_dimensions(
        apartments)[2]))

  # vrat' budovu jako jeden objekt:
  return join_objects_to_new(building, "apartment_building")
```

Zdrojový kód 11: Funkce pro vytvoření obytné budovy

Nyní již máme vše potřebné pro generování obytné budovy, ostatní budovy se generují obdobně. Poslední funkcí, kterou zde uvedu, je funkce pro vytvoření obytné budovy podle náhodně zvolených parametrů.

```
1  def create_random_apartment_building():
2    # získej uživatelovo nastavení parametrů:
3    min_no_levels = settings.rand_apartment_building_min_no_levels
4    max_no_levels = settings.rand_apartment_building_max_no_levels
5    min_x = settings.rand_apartment_building_min_x
6    max_x = settings.rand_apartment_building_max_x
7    min_y = settings.rand_apartment_building_min_y
8    max_y = settings.rand_apartment_building_max_y
9
10   # náhodně vyber design index (apartment_building_designs je promě
        nná obsahující počet vymodelovaných designů pro obytné budovy):
11   design_index = randint(0, apartment_building_designs-1)
12
13   # náhodně vyber parametry podle uživatelem zadaných mezí:
14   no_levels = randint(min_no_levels, max_no_levels)
15   x = randint(min_x, max_x)
16   y = randint(min_y, max_y)
17
18   # náhodně vyber počet vchodů a balkonů (vchod musí vždy být minimá
        lně jeden):
19   no_entrances = randint(1,x)
20   no_balconies = randint(0,x)
21
22   # vytvoř a vrat' obytnou budovu:
23   return create_apartment_building(design_index, no_levels, x, y,
        no_entrances, no_balconies)
```

Zdrojový kód 12: Funkce pro vytvoření obytné budovy s náhodnými parametry

Jak jsme již viděli při popisu ovládání na straně 39, volbu parametrů uživatelé mohou ovlivnit nastavením příslušných mezí (obrázek 35). Funkci „create_random_apartment_building“ (zdrojový kód 12) dále používá algoritmus pro vytvoření sektoru.

6.4 Náhodné cesty v sektoru

Cesty jsou ukládány pomocí matic, proto je nejprve potřeba definovat funkce pro práci s maticemi.

```
1 def create_empty_matrix(size, fill = 0):
2   matrix = []
3   row = []
4   for i in range(0,size):
5     row.append(fill)
6   for i in range(0,size):
7     matrix.append(row[:]) # "[:]" vytvoří kopii řádku
8   return matrix
```

Zdrojový kód 13: Funkce pro vytvoření matice

Ve zdrojovém kódu 15 je uvedena funkce pro vygenerování náhodné cesty maticí. Jelikož je zapotřebí ošetřit spoustu případů, jak může právě vytvářená cesta kolidovat s již existujícími cestami, funkce obsahuje mnoho podmínek. Z tohoto důvodu opět nahradím některé části pseudokódem. Výsledkem může být následující matice („1“ značí cestu):

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Plný zápis funkce je samozřejmě k dispozici ve zdrojovém kódu generátoru měst, tam jsou také přehledně okomentovány všechny ošetřené případy se zjednodušenými schématy cest. Parametry této funkce jsou:

1. matrix ... matice,
2. size ... velikost matice,
3. row ... index řádku, na kterém má cesta začínat,
4. col ... index sloupce, na kterém má cesta začínat,
5. vertical ... má být cesta vodorovná/svislá,
6. mark ... značka, která v matici označí cestu (výchozí je 1),
7. cannot_end ... cesta nesmí náhodně skončit.

```
1  def create_path(matrix, size, row, col, vertical = True, mark = "1",
       cannot_end=False):
2    # deklarace proměnných:
3    distance_to_corner = 0
4    previous_direction = 0 # 0 = dolů, 1 = doprava
5
6    # pravděpodobnost, že cesta skončí:
7    will_end = False
8    if not cannot_end and randint(0, 100) < 90: will_end = True
9
10   # cesta v matici:
11   while (row < size) and (col < size):
12     matrix[row][col] = mark
13
14     # jestliže cesta skončí, rozhodni zde:
15     if will_end and row > 1 and col > 1 and randint(0, 1000) > 800 or
           row==size-1 or col==size-1: break
16
17     if row>0 and row<size-1 and col>0 and col<size-1:
18       # VYŘEŠ PŘÍPADY, KDY BY VEDLE SEBE VEDLY DVĚ CESTY, NEBO SKONČI
19
20     # uprav vzdálenost k další zatáčce
21     if distance_to_corner > 0:
22       distance_to_corner = distance_to_corner - 1
23
24       if previous_direction == 0: row = row+1 # pokračuj dolů
25       else: col = col+1 # pokračuj doprava
26
27     # zatáčky:
28     else:
29       if col==0 and not vertical: # vertikální cesty začínají dolů
30         col = col+1
31         previous_direction = 1
32
33       # první a poslední řádek/sloupec nesmí končit křižovatkou, nebo
             cestou podél okraje
34       elif (randint(0,100) > 50 or row==0 or row==size-1) and col!=
             size-1:
35         row = row+1
36         previous_direction = 0
37
38       else:
39         col = col+1
40         previous_direction = 1
41
42       # podle podmínky nastav novou vzdálenost k zatáčce
43       if distance_to_corner == 0:
44         distance_to_corner = randint(settings.min_distance_to_corner,
             settings.max_distance_to_corner)
45
46   return True
```

Zdrojový kód 14: Funkce pro vytvoření náhodné cesty v matici

Nyní již stačí vytvořit funkci, která opakovaným voláním funkce „create path“ vytvoří mapu cest pro sektor. Vstupní parametry „road indices row“ a „road indices col“ musí splňovat pravidla uvedená na straně 25!

```
1  def create_road_map_matrix(size, road_indices_row, road_indices_col)
       :
2    matrix = create_empty_matrix(size)
3
4    # vytvoř cesty pro zadané indexy "road_indices_row":
5    for ci in road_indices_row:
6      row = 0
7      col = ci
8      create_path(matrix, size, row, col, vertical = True, mark =
           road_matrix_mark, cannot_end=False)
9      return matrix
10
11   # OBDOBNĚ VYTVOŘ CESTY PRO "road_indices_col"
12
13   # náhodně vytvoř nové cesty (maximálně 3):
14   for i in range(0,3):
15     if randint(0, 100) < 90: # pravděpodobnost vytvoření
16       row = randint(1, size-2)
17       col = randint(1, size-2)
18       i = 0
19       no_of_tries = 5
20
21       # pokud začátek cesty nesplňuje podmínky, zkus to znovu:
22       while(i < no_of_tries and (matrix[row-1][col-1] != 0 or matrix[
             row-1][col+1] != 0 or matrix[row+1][col-1] != 0 or matrix[
             row+1][col+1] != 0) ):
23         row = randint(1, size-2)
24         col = randint(1, size-2)
25
26       if randint(0,1) > 0:
27         create_path(matrix, size, row, col, vertical = True, mark =
               road_matrix_mark, cannot_end=True)
28       else:
29         create_path(matrix, size, row, col, vertical = False, mark =
               road_matrix_mark, cannot_end=True)
30
31   return matrix
```

Zdrojový kód 15: Funkce pro vytvoření matice cest

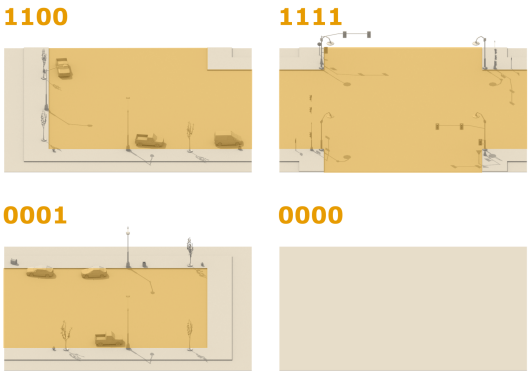
Funkce „create road map matrix“ tedy může vytvořit například matici podobnou této:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Pro práci s maticí je ve zdrojovém kódu generátoru měst definováno ještě mnoho funkcí, které si tu už ukazovat nebudeme. Důležitá je například funkce „get matrix output path indices“, která zjistí výstupy vygenerovaných cest, aby je bylo možné předat vedlejšímu sektoru – díky tomu na sebe cesty sektorů tedy budou navazovat.

6.5 Generování 3D modelů cest podle zadané mapy

3D modely částí cest používají jednoduché čtyřmístné značení, ze kterého lze vyčíst, jak část cesty vypadá. Jedná se o posloupnost 1 a 0, kdy 1 znamená, že z tohoto směru cesta vede, a 0 znamená, že nevede. Směry jsou pak uvedeny v pořadí „shora, zprava, zdola, zleva“, jak je názorně předvedeno na obrázku 63.



Obrázek 63: Příklad značení částí cest

Jestliže budeme mít v matici cest označená prázdná místa 0, a cesty 1, je velice snadné při průchodu maticí určit potřebnou část cesty (stačí kód ve správném pořadí poskládat ze sousedních hodnot).

Aby bylo v rámci jednoho sektoru stejné pouliční osvětlení, je nutné nejprve náhodně vybrat příslušný 3D model, který se poté předá funkci pro vytvoření 3D modelu části cesty.

```
def create_road_3d_model(matrix, road_meshes, road_props_meshes):
  rows = len(matrix)
  cols = len(matrix[0])
  road_objs = []

  # ZÍSKEJ RELEVANTNÍ MESHE PODLE TYPU (road_props_mesh_arrays,
      intersection_props_mesh_arrays)
  # NÁHODNĚ VYBER POULIČNÍ OSVĚTLENÍ (lamp_mesh)

  # zpracuj první a poslední řádek:
  for j in range(0,2):
    if j == 1: j = j*rows-1 # poslední řádek

    for i in range(0, cols):
      part_function = "0000"
      if matrix[j][i] != 0:
        part_function = "1010"

        # slepá cesta:
        if j == 0 and matrix[j+1][i] == 0: part_function = "1000"
        if j > 0 and matrix[j-1][i] == 0: part_function = "0010"

      # vytvoř 3D model části cesty a posuň jej:
      part = create_final_road_part(road_meshes,
          road_props_mesh_arrays, intersection_props_mesh_arrays,
          lamp_mesh, function = part_function)
      shift_object_by_dimensions(part, (i, -j-1, 0))
      road_objs.append(part)

  # OBDOBNĚ ZPRACUJ PRVNÍ A POSLEDNÍ SLOUPEC

  # zpracuj zbytek matice:
  for i in range(1, rows-1):
    for j in range(1, cols-1):
      part_function = "0000"

      if matrix[j][i] != 0:
        # poskládej kód části cesty:
        top = convert_non_zero_mark(matrix[j-1][i])
        right = convert_non_zero_mark(matrix[j][i+1])
        bottom = convert_non_zero_mark(matrix[j+1][i])
        left  = convert_non_zero_mark(matrix[j][i-1])

        part_function = top + right + bottom + left

      # vytvoř model části cesty, a posuň jej:
      part = create_final_road_part(road_meshes,
          road_props_mesh_arrays, intersection_props_mesh_arrays,
          lamp_mesh, function = part_function)
      shift_object_by_dimensions(part, (i, -j-1, 0))
      road_objs.append(part)

  return join_objects_to_new(road_objs, "city_sector_base")
```

Zdrojový kód 16: Funkce pro vytvoření 3D modelu z matice cest

6.6 3D model sektoru

Funkce „create city sector“ vytvoří kompletní 3D model sektoru města. Nejprve náhodně vytvoří mapu cest, ze které vytvoří 3D model. Poté okolo cest naskládá budovy (funkce[13] „create building vertical“ a „create buildings horizontal“). Nakonec se prázdná místa zaplní stromy a funkce vrátí nově vzniklý 3D model sektoru s výstupními indexy jeho cest (aby bylo možné na něj napojit další sektor).

```
1  def create_city_sector(size, road_indices_row, road_indices_col,
       building_types, meshes_array):
2    # vytvoř náhodnou mapu cest:
3    matrix = create_road_map_matrix(size, road_indices_row,
         road_indices_col)
4
5    road_part_mesh = meshes_array[0]
6    street_props_meshes = meshes_array[1]
7    empty_space_meshes_tree = meshes_array[2]
8    road_meshes = meshes_array[3]
9    road_props_meshes = meshes_array[4]
10   sector_3D_models = []
11
12   # vytvoř 3D model cest:
13   road = create_road_3d_model(matrix, road_meshes, road_props_meshes
         )
14   sector_3D_models.append(road)
15
16   # vytvoř bloky budov:
17   buildings_ver = create_buildings_vertical(matrix, road_part_mesh,
         building_types, street_props_meshes, empty_space_meshes_tree)
18   sector_3D_models.append(buildings_ver)
19   buildings_hor = create_buildings_horizontal(matrix, road_part_mesh
         , building_types, street_props_meshes, empty_space_meshes_tree)
20   sector_3D_models.append(buildings_hor)
21
22   # vyplň volná místa stromy:
23   trees = fill_sector_with_trees(matrix, road_part_mesh,
         empty_space_meshes_tree, settings.min_amount_of_trees, settings
         .max_amount_of_trees)
24   sector_3D_models.append(trees)
25
26   # vrať 3D model sektoru a výstupní indexy jeho cest:
27   return [join_objects_to_new(sector_3D_models, "city_sector"),
         get_matrix_output_path_indices(matrix)]
```

Zdrojový kód 17: Funkce pro vytvoření 3D modelu sektoru

[13] Obě tyto funkce mají na vstupu matici cest, ze které zjistí průběh cesty. Po vytvoření budov označí v matici již zaplněná políčka.

6.7 3D model města

Vytváření 3D modelu města funguje na podobném principu jako vytváření 3D modelu cest. Nejprve je tedy nutné vytvořit matici, ve které je zaznamenáno rozložení města (typy sektorů a jejich pozice). Poté se matice projde, a vytvoří se příslušné modely sektorů, ze kterých se nakonec složí finální 3D model města.

Funkci „create city matrix“, která vytvoří matici rozložení města, zde rozebírat nebudeme. Ukážeme si však možný výsledek:

$$\begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 3 & 3 & 3 \end{bmatrix}$$

Povšimněte si, že se jedná o stejnou matici, která se pro nápovědu zobrazuje v ovládacím panelu (viz obrázek 12 na straně 24) pro nastavení sektorů. Funkce pro vytvoření 3D modelu města je opět přehledně okomentovaná ve zdrojovém kódu generátoru měst, proto zde uvedu převážně pseudokód této funkce.

```
1  def create_city(sector_size, no_city_centre_sectors,
       no_city_apartments_sectors, no_city_houses_sectors,
       no_city_industrial_sectors):
2
3    # 1 - JESTLIŽE JSOU VŠECHNY SEKTORY S POČTEM == 0, SKONČI
4    # 2 - VYTVOŘ MATICI MĚSTA
5    # 3 - VYTVOŘ POLE RELEVANTNÍCH MESHŮ "meshes_array" A DALŠÍ
6    # 4 - VYGENERUJ NÁHODNÉ VSTUPY CEST DO MĚSTA
7
8    # 5 - vytvoř sektory:
9    for i in range(0, rows):
10     for j in range(0, cols):
11       # VYTVOŘ 3D MODEL SEKTORU
12       # POSUŇ SEKTOR NA PŘÍSLUŠNÉ MÍSTO
13       # ULOŽ VÝSTUPNÍ CESTY PRO DALŠÍ SEKTOR
14
15       # překresli 3D scénu:
16       bpy.ops.wm.redraw_timer(type='DRAW_WIN_SWAP', iterations=1)
17
18   # 6 - vyčisti vymazané objekty:
19   clear_data_blocks()
20
21   return True
```

Zdrojový kód 18: Funkce pro vytvoření 3D modelu města

Úplně poslední funkcí, kterou zde uvedu, je funkce pro vytvoření města, kterou uživatel vyvolá z ovládacího panelu stisknutím tlačítka „GENERATE 3D CITY“ (viz strana 46). Tato funkce se jednoduše podívá na uživatelem zadané nastavení, se kterým zavolá funkci pro vytvoření města (zdrojový kód 16).

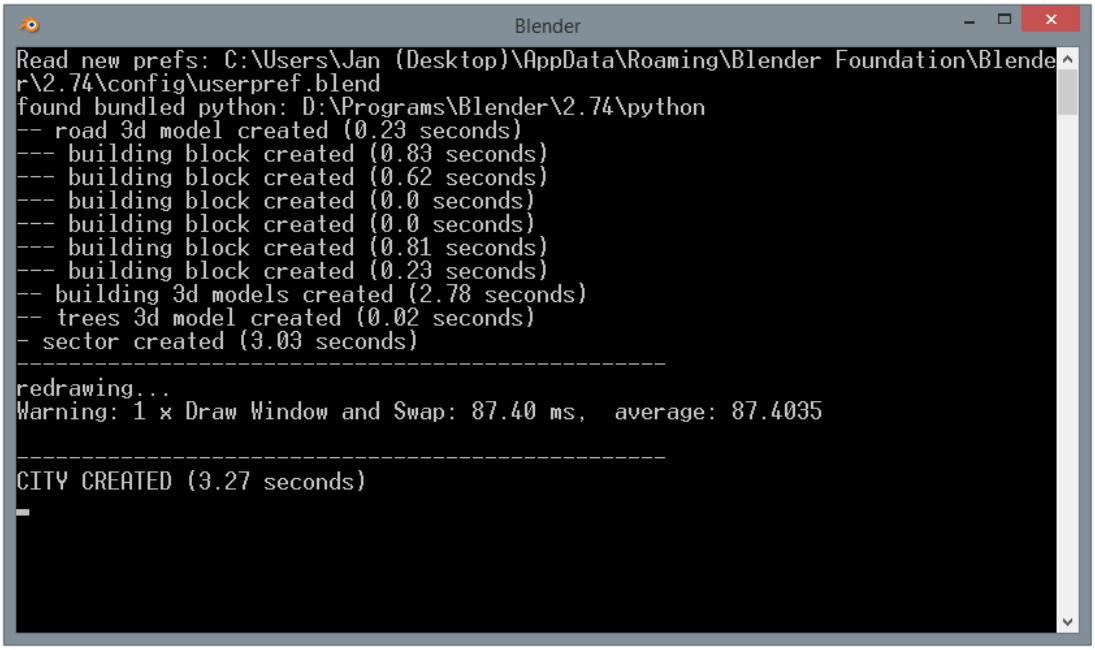
```
def create_city_with_settings():
  # nastavení od uživatele:
  sector_size = settings.sector_size

  no_city_centre_sectors = settings.no_city_centre_sectors
  no_city_apartments_sectors = settings.no_city_apartments_sectors
  no_city_houses_sectors = settings.no_city_houses_sectors
  no_city_industrial_sectors = settings.no_city_industrial_sectors

  # vytvoř město:
  return create_city(sector_size, no_city_centre_sectors,
     no_city_apartments_sectors, no_city_houses_sectors,
     no_city_industrial_sectors)
```

Zdrojový kód 19: Funkce pro vytvoření 3D modelu města

Nyní už tedy známe vše potřebné pro generování měst. Ovládací panel je vytvořen pomocí omezených možností Pythonu generovat části grafického rozhraní pro Blender.

Při testování generátoru je výhodné si otevřít konzoli (ve Windows horní lišta Blenderu > Window > Toggle System Console),[14] protože se do ní vypisují informace během vytváření měst.

[14]Pozor! Konzole se tímto způsobem musí i zavřít, pokud bychom ji totiž zavřeli jako běžné okno, zavře se bez varování také okno Blenderu se současným projektem!



Obrázek 64: Konzole s informacemi o tvorbě města

Závěr

Cílem práce bylo vytvořit doplněk pro program Blender, který měl umožnit generovat 3D modely měst podle zadaných parametrů, a který by se měl ovládat z výchozího grafického prostředí Blenderu. Text práce je pomyslně rozdělen na dvě části.

V první (uživatelské) části jsem čtenáře seznámil s Blenderem a s jeho netradičním ovládáním. Poté jsem popsal, jak se doplněk ovládá. Popis zahrnuje i veškeré pokročilé možnosti tohoto doplňku.

V druhé (programátorské) části jsem vysvětlil, jaké algoritmy používám pro generování města. Rozsah zdrojového kódu tohoto doplňku bohužel neumožňuje zacházet do přílišných podrobností, nejpodstatnější části jsou zde však vysvětleny.

Conclusions

The aim of this thesis was to build a Blender addon, which would be able to generate 3D models of cities by given parameters, and which would be controlled from Blender user interface. The text is notionally divided into two parts.

In the first (user) part, readers got acquainted with Blender and its unconventional controls. Then I have described how to use the addon. The description includes all the advanced capabilities of this addon.

In the second (programmer) part, I have explained the algorithms that I'm using to generate a city. Unfortunately, the scope of the source code of this addon doesn't allow me to go into too much detail, but the most important parts are explained.

A Obsah přiloženého CD

doc/
: Složka obsahuje text práce ve formátu PDF a všechny soubory potřebné pro pro jeho bezproblémové vygenerovaní.

src/
: Složka obsahuje soubory potřebné k instalaci doplňku „Blender City Generator“ do Blenderu (neobsahuje instalátor Blenderu samotného).

readme.txt
: Soubor obsahuje instrukce pro spuštění a instalaci doplňku „Blender City Generator“.

Navíc CD obsahuje:

data/
: Složka obsahuje ukázkové projekty se sektory města vygenerovanými doplňkem „Blender City Generator“.

install/
: Složka obsahuje instalační soubory Blenderu 2.74 pro různé operační systémy.

Bibliografie

[1] JAWORSKI, Witold. Programming Add-Ons for Blender 2.5 [online]. 2011. ISBN 978-83-931754-2-0. Dostupné z: http://www.airplanes3d.net/downloads/pydev/pydev-blender-en.pdf

[2] LARSSON, Thomas. Code snippets.: Introduction to Python scripting for Blender 2.5x [online]. 2011. Dostupné z: https://wiki.blender.org/index.php/Dev:Py/Scripts/Cookbook/Code_snippets

[3] MULLEN, Tony. Mastering Blender. 2nd ed. Indianapolis: Wiley. ISBN 1118275403.

[4] ŽÁRA Jiří, FELKEL Petr, BENEŠ Bedřich a SOCHOR Jiří. Moderní počítačová grafika, 2. vydání.. Computer Press, 2005. ISBN 8025104540.

[5] POKORNÝ, Pavel. Blender: naučte se 3D grafiku. 2., aktualiz. a rozš. vyd. Praha: BEN – technická literatura, 2009, 286 s. ISBN 978-80-7300-244-2.

[6] Dokumentace Blenderu 2.74. Dostupné z: https://www.blender.org/api/blender_python_api_2_74_0/contents.html