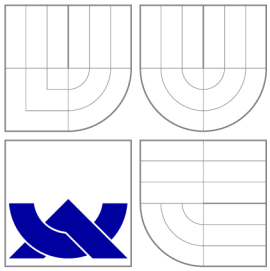
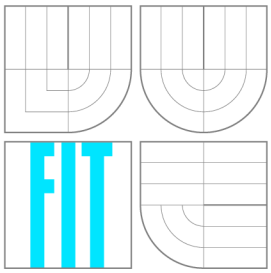


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SKETCHUP JAKO NÁSTROJ PRO NÁVRH NÁBYTKU

SKETCHUP WOODWORKING PLUGIN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ALEŠ MATĚJ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ZDENĚK VAŠÍČEK, Ph.D.

BRNO 2016

Zadání bakalářské práce/17827/2015/xmatej48

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Matěj Aleš**
Obor: Informační technologie
Téma: **Sketchup jako nástroj pro návrh nábytku**
Sketchup Woodworking Plugin
Kategorie: Počítačová grafika

Pokyny:

1. Seznamte se s aplikací Sketchup určenou k tvorbě 3D modelů objektů. Zaměřte se zejména na způsob reprezentace jednotlivých entit, uchování atributů a možnosti tvorby plugin modulů v jazyce Ruby.
2. Seznamte se s typickými operacemi prováděnými při návrhu nábytku složeného z prefabrikovaných desek (LTD, MDF apod.) od návrhu až po fyzickou realizaci. Navrhněte sadu nástrojů pro aplikaci Sketchup, které pomohou návrhový proces zefektivnit.
3. Zpracujte studii na výše uvedené téma.
4. Navržené nástroje implementujte formou tzv. plugin modulu.
5. Diskutujte dosažené výsledky a možná rozšíření.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Vašíček Zdeněk, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Cílem této práce je seznámení se s aplikací SketchUp, postupy návrhu nábytku z prefabrikovaných desek a navžení i implementace nástrojů pro zefektivnění těchto procesů. Nástroje jsou ve formě rozšíření pro SketchUp a využívají jeho API v jazyce Ruby. Text samotný tak především rozebírá základy počítačové 3D grafiky, její použití v aplikaci SketchUp, třídy a metody API aplikace SketchUp, základy tvorby nábytku z prefabrikovaných desek a samotné rozšíření. Popisovnané nástroje pro zefektivnění jsou zaměřeny na tvorbu modelů desek, jejich zprávu pomocí uživatelského rozhraní a generování hranění hran.

Abstract

Main goals of this work are to understand the SketchUp application, learn the basic tasks of fiberboard furniture design and the design and implementation of tools, which should increase efficacy of these tasks. Tools are created as plug-ins for SketchUp using its Ruby API. The text it self is focused on fundamentals of 3D computer graphics and use of them in SkechtUp, fiberboard furniture design processes and the woodworking plug-in it self. Described tools are used for creation of board models, their management through user interface and generation of edging for boards.

Klíčová slova

SketchUp, API, Ruby, rozšíření, modelování, návrh nábytku, prefabrikované desky, nástroje, MDF, hranění

Keywords

SketchUp, API, Ruby, plug-in, modelling, designing furniture, fiberboard, tools, MDF, edging

Citace

Aleš Matěj: Sketchup jako nástroj pro návrh nábytku, bakalářská práce, Brno, FIT VUT v Brně, 2016

Sketchup jako nástroj pro návrh nábytku

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Zdeňka Vašíčka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Aleš Matěj
17. května 2016

Poděkování

Děkuji panu Ing. Zdeňku Vašíčkovi, Ph.D. za odbornou pomoc a vedení práce.

Obsah

Kapitola 1

Úvod

Cíl práce je navrhnout a vytvořit sadu nástrojů ve formě zásuvných modulů (rozšíření) pro aplikaci SketchUp, které pomohou zefektivnit návrh nábytku složeného z prefabrikovaných desek.

Dnes se design, nejen nábytku, dělá pomocí CAD nástrojů. Jedním z nich je i SketchUp, která je dostupný jak komerčně tak zdarma. SketchUp si díky své jednoduchosti a přístupnosti vybudoval početnou uživatelskou základnu. Ačkoliv již existuje nepřeberné množství rozšíření, proces návrhu nábytku je těžkopádný. Uživatel musí opakovaně vytvářet modely kvádrů (desek) s přesnými rozměry a přitom počítat s pozdější tvorbou hranění. Vytvořené nástroje by proto měli umožňovat efektivnější a rychlejší tvorbu desek i jejich provázání se zdrojovým materiálem. Zjednodušil by se tak i postup tvorby nářezového plánu, tedy rozvržení řezů zdrojové desky materiálu, což je jeden z hlavních výstupů návrhu nábytku z prefabrikovaných desek. Další nástroj bude automaticky vytvářet hranění na základě nastavení uživatele. Následuje úvod k rozebíraným tématům v této práci.

Abychom měli šanci podrobněji porozumět činnosti aplikace SketchUp, jakožto softwaru k 3D modelování, je nutné se seznámit s obecnými principy 3D grafiky. Tato znalost se ale určitě uplatní i při návrhu a zpracování samotného rozšíření. Protože i zde bude bezpochyby nutné pracovat s geometrií modelu, provádět její úpravy a transformace. Kapitola 2, která se zabývá touto problematikou, tak tvoří naprostý základ práce.

Další v pořadí je kapitola 3, zpracovávající problematiku programu SketchUp. Protože SketchUp poskytuje poměrně obsáhlé a komplexní API v jazyce Ruby, bude využití jeho objektů, modulů a funkcí velmi časté. Z hlediska efektivity je také mnohem výhodnější používat API, protože nativní implementace poskytuje vyšší výkon, než totožná funkcionalita v Ruby. Je proto důležité porozumět základní filozofii a mechanismům fungování této aplikace a jejího API.

Tvorba nábytku z prefabrikovaných desek má jisté specifika. V kapitole 4 bychom se měli dozvědět s jakými materiály se běžně pracuje a co za operace se s nimi provádí. Tyto informace nám pomohou v návrhu nástrojů rozšíření. Jedna z podkapitol se také zabývá již existujícími rozšířeními. Přínosné je pak zejména porovnání informací o nejčastěji prováděných operacích při modelování nábytku a existujících rozšířeních.

Kapitola 5 obsahuje popis ve formě specifikace sady nástrojů. Co konkrétně by měli dělat a k čemu by měli sloužit. Další kapitola 6 je obsáhlejší a podrobně rozebírá implementaci rozšíření. Každý modul je nejprve shrnut v obecnějším přehledu a později v patřičných podkapitolách jsou zkoumány komplikovanější části implementace. Vysvětleno je i uživatelské rozhraní a jeho vnitřní mechanismy.

Poslední kapitola pak podává stručný přehled o vyhodnocení rozšíření.

Kapitola 2

Základy 3D geometrie

2.1 Trojrozměrný prostor

Mějme posloupnost n čísel z $\mathbb{R}$. Tu můžeme chápat jako pozici v n-rozměrném prostoru. Označme prostor jako $\mathbb{R}^n$ Pokud se n rovná třem, množina všech takovýchto pozic se nazývá trojrozměrný Eukleidův prostor [10]. Jinak řečeno trojrozměrný prostor je geometrické prostředí, kde jsou potřeba tři hodnoty, abychom určili pozici prvku. Tento prostor se nejčastěji používá jako model fyzického světa a tak je chápán i v této práci.

Trojrozměrný prostor definuje souřadné osy. Jednotlivé osy jsou navzájem kolmé. Bod, kde se všechny osy protínají, nazýváme počátek souřadnicového systému a typicky jej označujeme *(0,0,0)*. Relativně k těmto osám můžeme určit pozici bodu jako uspořádanou trojici reálných čísel. Každé číslo udává vzdálenost bodu od počátku podél patřičné osy. Této variantě systému souřadnic se říká Kartézská soustava souřadnic. Nicméně existuje nekonečně mnoho metod jak popisovat pozici bodu v trojrozměrném prostoru, například pomocí sférické či válcové soustavy souřadnic.

V této práci budeme nejčastěji pracovat s prvky 3D prostoru jako jsou body, roviny, vektory a přímky.

2.1.1 Bod

Za bod můžeme označit prvek prostoru, tedy $p \in \mathbb{R}^3$. V Eukleidově prostoru je bod důležitým základem geometrie. To znamená, že bod nelze definovat pomocí dříve uvedené terminologie. Bod je definován pomocí axiomů, které musí splňovat. V podstatě bod nesmí mít žádný rozměr jako je délka, objem nebo plocha. Můžeme říci, že body reprezentují unikátní místo v 3D prostoru.

2.1.2 Přímka

Přímka, stejně jako bod, tvoří základ geometrie a je definována pomocí postulátů. Nicméně na rozdíl od bodu má jeden rozměr. Můžeme si ji představit jako nekonečně tenkou a dlouhou čáru, která je v trojrozměrném prostoru určena například dvěma různými body p_1 a p_2.

Dva body ležící na přímce vymezují úsečku. Úsečka je tak část přímky a obsahuje všechny body, které leží mezi jejími koncovými body. Úsečka v 3D modelu slouží k určení hrany.

2.1.3 Rovina

Na roviny můžeme nahlížet jako na rovné dvojrozměrné povrchy nekonečných rozměrů. V 3D prostoru je rovina jednoznačně určena například pomocí přímky a bodu, který nesmí ležet na dané přímce, nebo dvěma různýma, nerovnoběžnýma přímkami. Roviny v trojrozměrném modelu určují prostorovou orientaci stěn.

2.1.4 Vektor

Vektory používané v 3D prostoru, někdy také zvané geometrické nebo prostorové vektory, jsou geometrické objekty, které mají směr a velikost. Operace s nimi formálně definuje lineární algebra. V prostoru pak vektor představuje přenesení bodu A do bodu B. Latinské slovo *vector* doslova znamená nosič. Vektory jsou obvykle reprezentovány šipkou a zapisují se pomocí souřadnic. Existují dva základní typy. Volný vektor, používaný i v aplikaci SketchUp, nevychází z žádného pevného bodu v prostoru, k jeho vyjádření stačí opravdu velikost a směr. Vázaný vektor má počátek, tedy vychází z konkrétního bodu.

2.2 Reprezentace modelu

Pro modelování a práci s 3D grafikou musíme porozumět jakým způsobem pracuje s 3D geometrií počítač. Trojrozměrné povrchy objektů jsou popsány matematicky a uloženy v paměti počítače jako modely[12]. Tento model můžeme za pomocí různých projekcí zobrazit na 2D plochu (monitor), nebo jej rovnou použít třeba na výpočet simulace fyzikálního jevu.

Většina modelů spadá do jedné ze dvou kategorií: hraniční a objemové modely. Objemové, pevné modely popisují vnitřní strukturu objektu, z čeho se skládá. Jsou realističtější ale náročnější na tvorbu a zpracování. Používají se především v medicíně, geologii a strojírenství pro různé simulace.

Naproti tomu hraniční modely reprezentují objekty popsáním jejich povrchu. Využívají k tomu základní stavební kameny geometrie: body, hrany a stěny[14]. Většina vizualizovaných modelů použitých pro filmy, hry či jinou grafiku (SketchUp) využívá hraniční modely. Podmnožinou hraničních modelů jsou drátové modely. Ty reprezentují model jen pomocí bodů a hran, díky tomu ale obsahují nedostatek topologických informací pro jednoznačnost modelu. Jsou tak vhodné pro rychlé a nenákladné zobrazení modelu.

2.3 Eulerova charakteristika

Každý konvexní mnohostěn má Eulerovu charakteristiku rovnou dvěma.

$$2 = F - E + V \tag{2.1}$$

Jedná se o vztah mezi počtem vrcholů (*vertices* - V), hran (*edges* - E) a stěn (*face* - F). Toto je zejména užitečné, když potřebujeme klasifikovat geometrii nebo ověřit, že se model nachází v určitém stavu.

2.4 Projekce

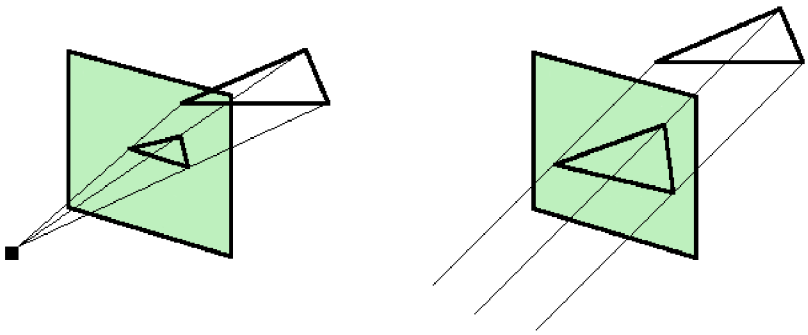
Abychom mohli zobrazit model a případně celou 3D scénu, potřebujeme znát tyto prvky: samotné objekty (jejich materiál, geometrii), zdroj světla a nastavení kamery[13]. Po vyhodnocení scény, kde například umístíme, transformujeme objekty do globální souřadného

systému scény, umístíme kameru a vše transformujeme do souřadného systému kamery, můžeme aplikovat projekci. Projekce převádí ze 3D do 2D prostoru, pomocí projekčního paprsku, který promítá body na průmětnu (plochu na kterou je promítáno). Základní dva druhy jsou paralelní a perspektivní projekce.

Paralelní projekce tvoří v zásadě nárys scény. Využívá rovnoběžných paprsků, čímž zachovává rovnoběžnost hran v modelu. Vzdálenost od průmětny neovlivňuje velikost průmětu (promítnutého modelu). Je vhodná především pro výkresy, technická schémata a dokumentace, kde je nutné zachovat specifikované velikosti a vzdálenosti.

Perspektivní projekce je naopak nelineární, všechny paprsky vycházejí z místa pozorovatele. Nezachovává tak rovnoběžnost v modelu a zprostředkovává vzdálenost pomocí velikosti průmětu. Tím odpovídá více realitě a je tak vhodnější pro napodobování reality ve hrách a architektuře. Existuje jednobodová, dvoubodová a tříbodová perspektivní projekce.

Porovnání použití paralelní a perspektivní projekce můžeme vidět na obrázku 2.1, kde perspektivní projekce promítaný trojúhelník zmenší, na základě vzdálenosti od průmětny, kdežto paralelní projekce zachová jeho velikost.



Obrázek 2.1: Perspektivní projekce a paralelní projekce

2.5 Transformace

Transformace ve 3D prostoru patří mezi často prováděné operace v počítačové grafice. Dovolují nám měnit pozici v prostoru. Obecně můžeme transformace aplikovat dvěma způsoby. Můžeme posunout jednotlivé vrcholy objektu v jeho souřadném systému. Nebo změníme celý souřadný systém modelu, čímž jej dostaneme do jiné pozice. Základní druhy transformací jsou: posunutí, rotace, změna měřítka, zkosení. Transformace ve 3D mohou být popsány maticí o 16 prvcích. Při skládání transformací násobíme matice mezi sebou. Důležité je, že záleží na pořadí.

Kapitola 3

SketchUp a jeho API

SketchUp je aplikace vyvinutá pro účely tvorby 3D modelů, zaměřená především na jednoduchost použití a přístupnost široké uživatelské základně. Uplatní se tak v mnoha odvětvích jako je architektura, interiérový design (například návrh nábytku), strojní inženýrství ale i při tvorbě video her či filmů. Výhodou je jeho rozšiřitelnost pomocí tak zvaných *extensions* psaných v Ruby. Existuje již například řada rozšíření (*extensions*) pro realistické renderování, čímž se SketchUp dostává na úroveň profesionálních nástrojů. SketchUp je dostupný ve freeware verzi jako SketchUp Make, která byla použita pro tvorbu této práce. Firma Trimble, současný vlastník, také poskytuje 3D Warehouse, který obsahuje doslova miliony modelů. Na sesterské platformě Extension Warehouse můžou uživatelé a vývojáři publikovat zdarma a prodávat své rozšíření.

SketchUp integruje rozhraní pro programování aplikací v dynamickém jazyce Ruby, čímž umožňuje uživatelům přístup k jeho vnitřní reprezentaci modelu, tvorbu vlastních nástrojů nebo automatizaci opakujících se úloh.

Instalace rozšíření je možná několika způsoby. Lze využít uživatelské rozhraní přímo v aplikaci SketchUp a jen zvolit konkrétní *extension* s koncovkou *rbz*. Další možností je pak ruční nakopírování zdrojových souborů do složky *Plugins* v adresářích aplikace.

3.1 Základní geometrické třídy

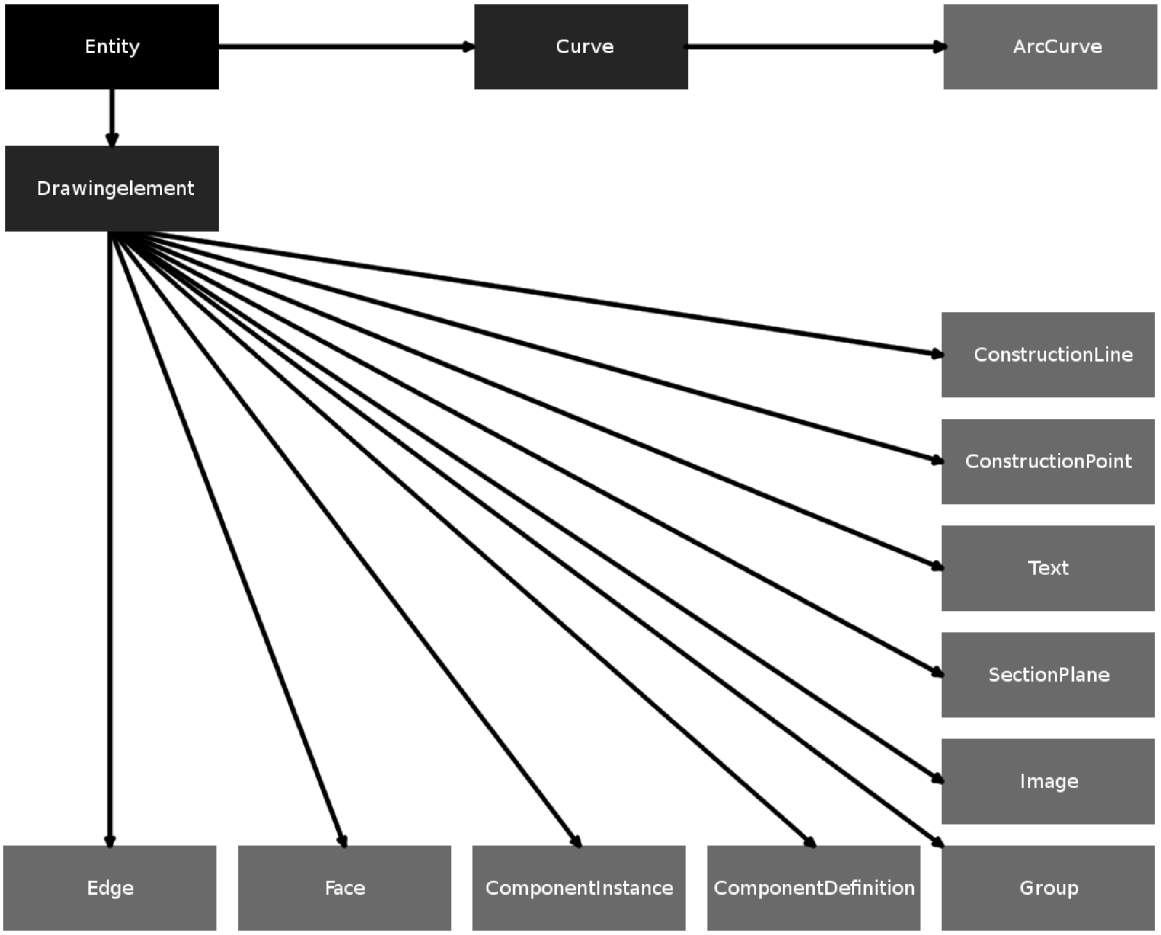
SketchUp zpřístupňuje veškerou geometrii, se kterou uživatel může pracovat, skrze třídu *Model*. Ta tedy slouží jako vstupní bod pro manipulaci s jejími dílčími částmi. Metoda *entities* třídy *Model* vrátí objekt třídy *Entities*, který představuje pole všech entit modelu. Dále je možné entity do sebe zanořovat pomocí skupin (*Group*) a komponent. Komponenty jsou přístupné skrze metodu *definitions* opět z třídy *Model*. Tato metoda předá volajícímu objekt třídy *DefinitionList*, který obsahuje seznam všech definic komponent.

Hierarchie nejčastěji používaných geometrických tříd vychází z *Entity*, z ní dále, kromě spousty dalších, dědí třída *Curve* a především *Drawingelement*. *Drawingelement* je základ pro cokoliv, co lze v aplikaci SketchUp zobrazit. Jedná se především o hrany, stěny, konstrukční body a čáry, viz diagram tříd 3.1. *Curve* a *ArcCurve* nepatří pod *Drawingelement*, protože se nejedná přímo o objekty, jenž by bylo možné vykreslit. Jsou to pouze kontejnery pro hrany (*Edge*) [4]. Tedy každá křivka, kružnice nebo výseč kružnice je v reprezentaci aplikace SketchUp chápána jako posloupnost hran. Z toho vyplývá, že SketchUp nezná ani zakřivené plochy, ty jsou vždy tvořeny různě poskládanými stěnami (*face*), za pomoci měkkých (*soft*) hran. Měkké hrany nejdou v modelu vidět a vytváří plynulý přechod mezi

dvěma stěnami.

Do základních geometrických třídy dále patří *ConstructionLine* a *ConstructionPoint*, které slouží jako pomocné značení při modelování. Dále *Text* a *Image* umožňují přímo v modelu zobrazit text a obrázek. Poslední třída *SectionPlane* dovoluje rozdělit model rovinou a schovat jednu vzniklou část, čímž usnadňuje prohlížení komplexních modelů.

S třídou *Entity* je možné spojit slovník atributů. Díky *AttributeDictionary* tak můžeme k různým entitám ukládat libovolná data ve formátu klíč/hodnota, kde klíč je řetězec. Každá entita může mít víc různých slovníků, specifikovaných jménem dané instance *AttributeDictionary*. Kromě entit je možné slovníky připojovat pouze k třídě *Model*.



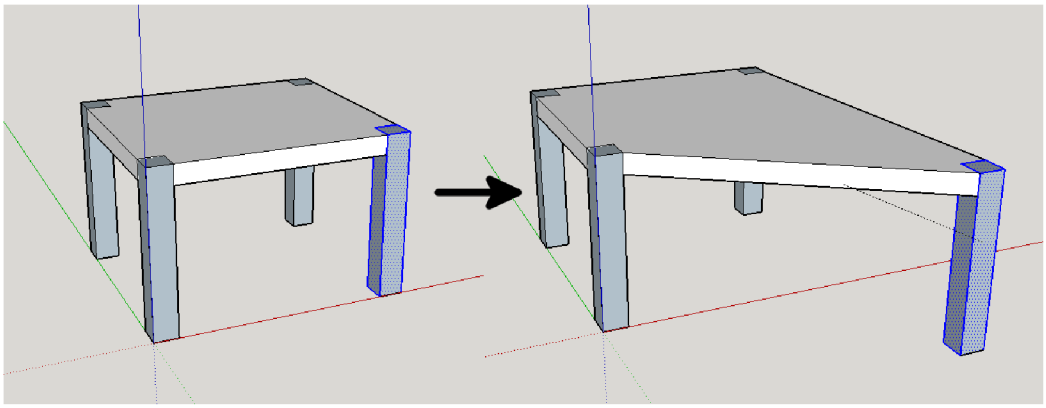
Obrázek 3.1: Základní geometrické třídy a jejich vztahy

3.2 Komponenty, skupiny a jejich podpora v API

Důležitou součástí práce v aplikaci SketchUp jsou komponenty a skupiny. Už názvy nám napovídají k čemu tyto entity slouží, dovolují nám rozdělit model na jeho dílčí části. Tím nám nejen usnadňují orientaci, ale také pomáhají lépe modelovat realitu. Pokud si vezmeme třeba jednoduchý stůl, můžeme ho rozdělit například na pět částí. Čtyři nohy a vrchní desku. Dává proto smysl, abychom si tímto způsobem dekomponovaly i náš model. Navíc při vytváření složitějších modelů, je jejich použití naprosto nezbytné, zabraňují totiž "slepování"

stěn a hran. Toto může být častý problém především při pozdějších úpravách modelu.

Na obrázku 3.2 můžeme vidět problém slepení nohy a desky stolu, kdy posunutí nohy deformuje geometrii vrchní desky. Pokud by byl model rozdělen do komponent zachovala by si vrchní deska původní tvar i po přesunutí nohy, protože komponenty nesdílejí hrany a stěny mezi sebou.



Obrázek 3.2: Slepené stěny

I přestože SketchUp vnitřně reprezentuje skupiny jako kombinaci definice a instance komponenty[6], z pohledu uživatele se jejich chovaní podstatně liší. Každá komponenta má svoji definici(*ComponentDefinition*), ze které vznikají instance (*ComponentInstance*). Komponenty se dají částečně přirovnat k objektům a třídám tak, jak s nimi pracuje objektově orientované paradigma. Vytvořené definice můžeme instanciovat a v aplikaci SketchUp tak získáme více komponent se stejnou definicí. Při vytvoření první komponenty se automaticky generuje její definice a současně se vytvoří jedna instance, která odpovídá původním označeným entitám. Hlavní výhodou je aktualizace definice, při libovolné úpravě jakékoliv její instance. To znamená, že se zároveň upraví i všechny instance vycházející z této definice. Jednou úpravou tak modifikujeme více míst v modelu, pokud by například všechny nohy stolu byly instancí jedné definice, po úpravě jedné nohy by se změna projevila na všech. Toto je velice výhodné při modelování designů, kde se často opakují jisté části, což nastává velmi často.

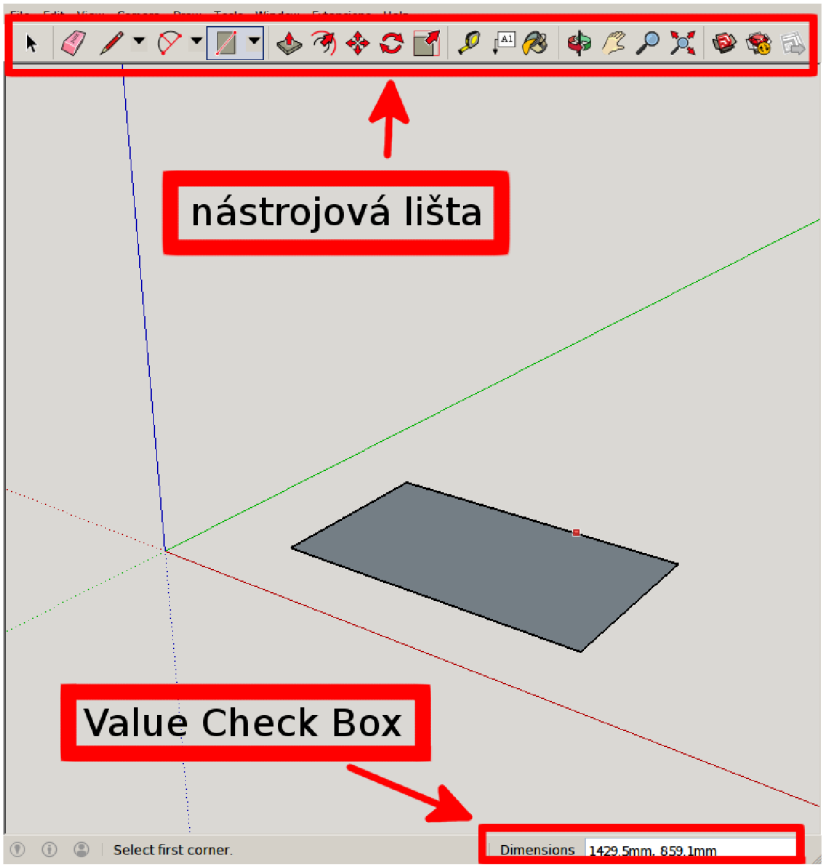
Naproti tomu každá skupina (*Group*) je unikátní a hodí se tak spíše pro neopakující se části geometrie. Ostatní vlastnosti mají podobné jako komponenty.

3.3 Nástroje

Uživatel s modelem manipuluje především pomocí nástrojů. Skrze ně se model vytváří a mají největší zásluhu na intuitivnosti ovládání. Ruby API umožňuje jejich tvorbu skrze rozhraní *Tool*. Mezi jeho metody patří mimo jiné události myši, klávesnice a samozřejmě možnost překreslovat aktuální náhled na model (*View*). Každý nástroj je třída, která implementuje právě zmíněné metody událostí. Vytvořením objektu třídy a jeho předání aplikaci SketchUp skrze metodu *select_tool* třídy *Model* se nástroj aktivuje v uživatelském rozhraní.

3.4 Value Check Box

SketchUp poskytuje funkci zadání přesné hodnoty do pole označeného jako *Value Check Box* (VCB). Použití je vyhrazené pro nástroje. Pokud se například upravuje rozměr, otáčí nebo třeba tvoří podstava, VCB nám umožňuje zadat požadovanou hodnotu číslem. Nástroje jsou připravené tak, aby uživatel, tam kde to dává smysl, mohl pouze přímo napsat číslo, to se automaticky rozpozná a vloží do VCB, poté stačí potvrdit klávesou *ENTER*. Ukázka *VCB* je na obrázku 3.3. *Value Check Box* také zobrazuje aktuální velikosti, pokud například zrovna tvoříme hranu, najdeme zde její délku.



Obrázek 3.3: Ukázka VCB a nástrojů v aplikaci SketchUp

3.5 Inference engine

Inference engine pomáhá uživateli pracovat ve 3D. Ukazuje tak například barevně označený bod, pokud jsem uprostřed hrany. Nebo může zobrazit popis (*tooltip*) označující, že tvoříme kolmou hranu k jiné hraně. SketchUp rozpozná řadu typů těchto skutečností a předává o nich uživateli informace. Nejčastěji pomocí barevných hran, bodů, popisků a pomocných čar[7]. Často může nastat situace, že se více typů kombinuje dohromady.

3.6 Uživatelské rozhraní

V případě, že potřebujeme interagovat s uživatelem formou dialogu, nabízí SketchUp v zásadě dvě možnosti tvorby dodatečného rozhraní. Skrze modul *UI* je možné zavolat funkci *inputbox*, která vytvoří jednoduchý dialog box. Zde je možné vytvořit vstupní pole pro data *input field* nebo výběr jedné z předdefinovaných hodnot *drop down selection*. Tento způsob neumožňuje komplikovanější chování.

Druhou cestou je využití třídy *WebDialog*, také z modulu *UI*. Zde jsou naopak možnosti takřka neomezené. K dispozici máme celé dynamické HTML. Za pomoci JavaScriptu, HTML a kaskádových stylů (CSS) tak můžeme vytvořit komplexní uživatelské rozhraní. Komunikace mezi rozhraním a aplikací probíhá oboustranně. V Ruby můžeme pomocí metody *add_action_callback* přidat funkci, která jde volat z JavaScriptu s volitelným množstvím parametrů. Tímto mechanismem komunikuje rozhraní s Ruby skriptem a předává mu data. Z Ruby můžeme zjistit hodnotu libovolného vstupního pole pomocí volání metody *get_element_value*. Stránce může SketchUp předávat informace pomocí *execute_script*. Tato metoda očekává na vstupu přímo řetězec kódu JavaScript, který vykoná nad uživatelským rozhraním. Mimo to má *WebDialog* mnoho dalších metod umožňujících nastavit parametry okna jako je velikost a pozice okna, barva, modalita, minimální a maximální velikost a podobně.

3.7 Observers

SketchUp umožňuje připojovat pozorovatele ke většině svých prvků. Tak zvaný *Observer* je objekt, který se sváže s konkrétním prvkem, instancí a reaguje na události. Je pak možné implementovat metody, které se spustí vždy při konkrétní akci. Například existuje *ViewObserver*, ten dovoluje doplnit metodu *onViewChanged*. Pokud ji implementujeme, vytvoříme instanci této třídy a pomocí metody *addObserver* přidáme tuto instanci objektu *View*. Zajistíme tím, že vždy když se *View* změní, to znamená například přemístění kamery, zavolá se i naše metoda.

Mimo *ViewObserver* nabízí SketchUp řadu dalších užitečných pozorovatelů, kteří zefektivní sledování změn a událostí, například *DefinitionObserver*, který může hlídat vytváření a mazání instancí definice komponenty nebo *SelectionObserver*, jenž zase monitoruje aktuální selekci v aplikaci SketchUp. Je nutné poznamenat, že pozorovatelé jako objekty tříd *observer* jsou nestabilní a můžou způsobovat pád celé aplikace SketchUp.

Kapitola 4

Modelování nábytku

V poslední době se často dává přednost nábytku připravenému ke složení z prefabrikovaných desek před masivem. Při práci s prefabrikovanýma deskami je hlavní výhodou především jejich dostupnost, pestrá škála povrchových úprav a cena. Jedná se o speciální, uměle vytvořené materiály, které se liší podle svých mechanických vlastností: pevnost, tvrdost, pružnost a odolnost. Každý druh prefabrikovaných desek je tak specializovaný pro jiné použití. Při konstruování nábytku je jejich hlavní cíl nahradit drahé masivní dřevo.

Při praktickém použití je většinou nutná povrchová úprava, nejčastější je použití dýhy, laminátu nebo lakování [5]. Nutné je také ohranit řezné hrany, například pomocí ABS (*Acrylonitrile butadiene styrene*) plastu, ten existuje v různých tloušťkách, nebo pomocí levnější varianty krycím papírem.

Desky se běžně prodávají v základních rozměrech, specifické pro daný materiál, v případě MDF je to například: 2.750 x 1.840 mm. Nicméně jsou dostupné i v dalších různých velikostech a tloušťkách. Desky jsou dodávány ve formě ploten, které je nutné nařezat na potřebné rozměry. Mezi největší prodejce u nás patří například firmy Egger a Kronospan.

4.0.1 OSB deska

Tento druh je vytvořen lisováním velkých dřevěných hoblin od 2 do 7 centimetrů v několika vrstvách. Desky jsou vodě odolné. Díky své nevzhlednosti a nevhodnosti pro povrchové úpravy se používají spíše ve stavebnictví, na tvorbu nábytku nejsou moc vhodné.

4.0.2 Dřevotřísková deska (DTD)

Nejrozšířenější druh je dřevotřísková deska. Má podobné vlastnosti jako MDF, ale díky větší hrubosti nelze povrch přímo lakovat nebo upravit potahovou PVC folii. Dřevotříska má také velmi malou odolnost proti vlhkosti. Z těchto důvodů se nejčastěji upravuje jejich povrch laminátem, vzniká tak laminátovaná třísková deska (LTD).

4.0.3 Polotvrdá dřevovláknitá deska (MDF)

Materiál s homogenní strukturou, jemnější jak dřevotříska, který se vyrábí bez použití lepidel, jedná se tak o přírodní materiál. Jeho hlavní výhody spočívají především v ceně, která je mnohem menší než u masivu, konzistentní pevnosti a velikosti, stabilních rozměrech, jednoduchém opracování a vhodnosti pro aplikaci dýh. Na druhou stranu jsou MDF náchylné na přímý kontakt s vodou a vrtání, především do okrajů, které často způsobuje poškození desky [2], proto je nutné při práci s nimi dbát určitých zásad.

4.1 Operace při modelování

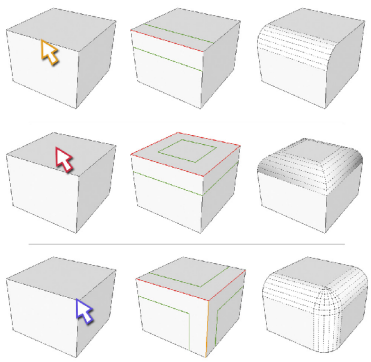
Modely desek je nutné zejména vytvářet, často specifické velikosti, a umísťovat na požadované pozice. Mnohokrát jsou také desky, konkrétně jejich rohy, upravovány do zaoblených tvarů, kvůli bezpečnosti. Dále je také někdy nutné počítat v modelech s povrchovou úpravou. Moření, lakování a aplikace PVC folií většinou není potřeba v modelu zohledňovat, ale použití dýhy by se už mělo projevit. Dýha je tenká vrstva nebo list masivního dřeva (0,3 mm až 5 mm), který se nalepí na povrch desky. Dalším důležitým prvek je ohranění hran, to může podstatně ovlivnit rozměry desek. V propracovanějších modelech se pak dají najít i druhy a pozice spojů jednotlivých desek, které je také potřeba v geometrii vytvořit. Pokud už je model hotový, je další logický krok jeho rozklad na komponenty a jejich umístění na přířez. Vytvoří se tak nářezový plán. [1] [3]

4.2 Existující rozšíření

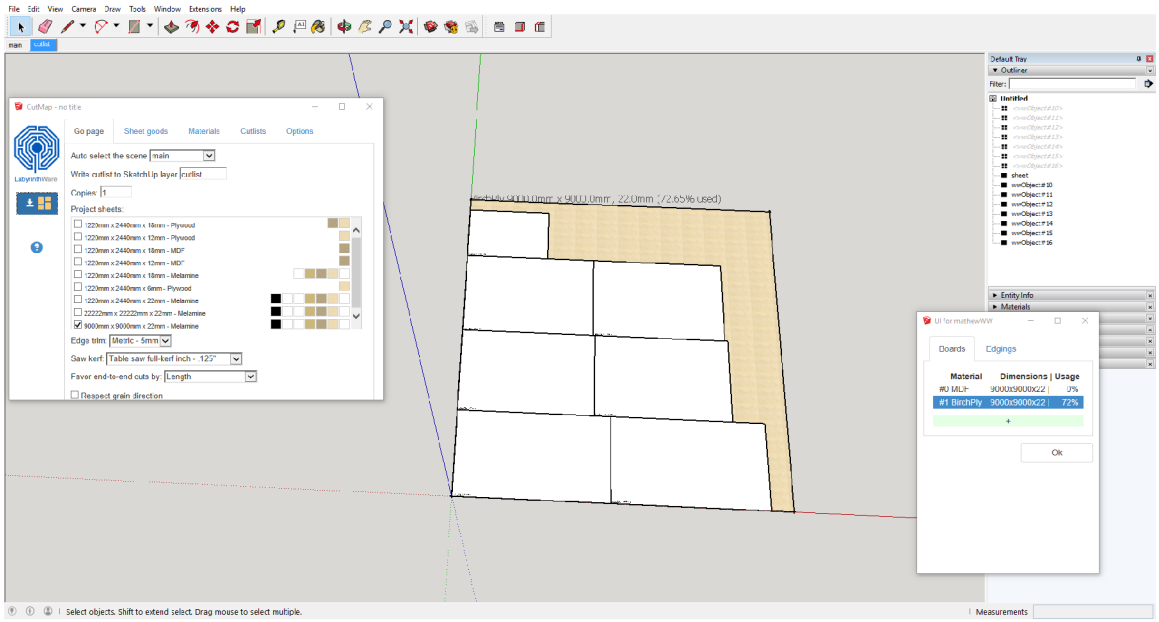
Na stránkách pro SketchUp je k dispozici řada rozšíření od jednodušších jako *3D Grid Tool* nebo *Align* umožňujících vytvořit mřížku pomocných čar nebo zarovnat označené entity, až po komplexní balíky přidávající sady nových nástrojů. Pro modelování nábytku lze například nalézt celou řadu nástrojů, které poskytují již předem vytvořené modely. Uživatel si tak může sestavit kuchyňskou linku z již dostupných produktů. Nejpropracovanější rozšíření, které ale není dostupné na Extension Warehouse, je *WUDWORX*. Toto rozšíření se skládá se sady nástrojů pro modelování práce s masivním dřevem. Například lze vytvořit po celé délce desky čepy a v protějším kusu žlábky, tyto dva protikusy pak do sebe zapadnou. Tento spoj je pevný, ale náročný na výrobu. Rozšíření implementuje modelování dalších druhů spojů i nástroj pro vytváření desek.

Další poměrně komplexní balíky jsou *CutMap*, *CutList* a *Builder*. Všechny tyto nástroje se specializují na tvoření nářezového plánu. *CutList* a *Builder* prezentují plán a nastavení čistě pomocí vestavěného HTML formuláře *WebDialog*. *CutMap* zase využívá přímo prostředí modelu a navíc dovoluje manipulaci s jednotlivými částmi nářezu. Je tak možné rozvržení ručně předělávat.

Další užitečné rozšíření je *RoundCorner*, jak napovídá název, zjednodušuje především zdlouhavý a často opakující se proces zaoblování hran. *RoundCorner* obsahuje celou řadu možných nastavení pro různé situace. Některé metody využití můžeme vidět na obrázku 4.1. Na obrázku 4.2 pak vidíme nářezový plán vytvoření rozšířením *CutMap* přímo uvnitř aplikace SketchUp.



Obrázek 4.1: Rozšíření *RoundCorner* [9]



Obrázek 4.2: Nářezový plán vytvořený pomocí *CutMap*

4.3 Důsledky

Našim cílem jsou tedy nástroje, které budou zrychlovat nejčastější operace při modelováni nábytku z prefabrikovaných desek. Z již existujících nástrojů je pro naši úlohu vhodné rozšíření *RoundCorner* na tvorbu zaoblených hran a některé z rozšíření starající se o tvorbu nářezového plánu, zde si můžeme dokonce vybrat. Nicméně neexistují žádně nástroje, které by nám pomohli tvořit modely samotných desek a jejich hranění. Potřebujeme nástroj, jenž nám pomůže s hlídáním tlouštěk u desek a nejlépe celkově zrychlí jejich tvorbu. Od vytvoření geometrie desky až po její spojení v komponentu. Tvorba hranění by také měla jít podstatně zefektivnit. Pokud uživatel zadá jeho tloušťku, mělo by celé hranění jít vygenerovat podle označené stěny, kde se má nalézat.

Kapitola 5

Návrh

Abychom usnadnili práci při modelování nábytku z prefabrikovaných desek vyjdeme z typického postupu. Uživatel si volí prefabrikované desky s tím, že jejich materiál a rozměry jsou dány výrobcem. To znamená, že i jejich tloušťka je pevně daná a všechny dílčí desky vytvořené z této zdrojové desky ji musí respektovat.

Hlavní koncept navrhovaného rozšíření spočívá v práci se zdrojovou deskou, která představuje modelářem zakoupenou prefabrikovanou desku. Pokud si ji poté uživatel zvolí, může vytvářet za pomocí nástroje na tvorbu desek její jednotlivé dílčí desky v modelu. Ty jsou automaticky propojeny se svojí zdrojovou deskou a zároveň získávají její materiál. Všechny dílčí desky (dále jen desky) musí být možné vyřezat ze zdrojové desky.

5.1 Uživatelské rozhraní

Uživatelské rozhraní musí především dovolovat vytvářet zdrojové desky různých materiálů a výběr mezi nimi při tvorbě desek. Pro lepší přehled by pak rozhraní mělo ukazovat procentuální využití zdrojových desek nebo přehled vytvořených hranění, konkrétně kolik bylo použito jakého hranění. Užitečnou funkcí rozhraní by také mohlo být vybrání všech desek jedné zdrojové desky přímo v modelu aplikace SketchUp.

5.2 Nástroj na tvorbu desek

Tento nástroj je specializovaný na tvorbu modelů desek. Typický postup, bez využití dalších rozšíření, tvorby libovolného kvádru, kterým je i dílčí deska, vyžaduje nejméně dva kroky. Je nutné definovat podstavu, základní stěnu, a tu poté vytáhnout *PushPull* nástrojem. Cílem tohoto nástroje je zjednodušit a zrychlit tvorbu takovýchto desek. Základní funkčnost tedy odpovídá tomuto postupu. Logika byla navržena následovně, po zvolení nástroje prvním kliknutím do modelu definuje uživatel počáteční bod. Poté se, stejně jako při použití *Rectangle* nástroje, začne mezi pozicí kurzoru a počátečním bodem tvořit obdélník. Ten je určený délkou uhlopříčky, vzdáleností mezi počátečním bodem a pozicí kurzoru, a rovinou se kterou uživatel pracuje. Pokud uživatel tvoří podstavu na existující stěně v modelu, bere se jako výchozí rovina určená touto stěnou. Jestliže je podstava ve volném prostoru, automaticky se zarovnává ve směru os souřadného systému modelu. Druhým kliknutím se tedy vytvoří podstava. Nástroj se tak posune do poslední fáze, jenž odpovídá *PushPull* nástroji a uživatel si může zvolit výšku své desky, kvádru.

Samozřejmostí je možnost zadávat hodnoty do VCB, pro vytvoření přesné míry, vzdálenosti.

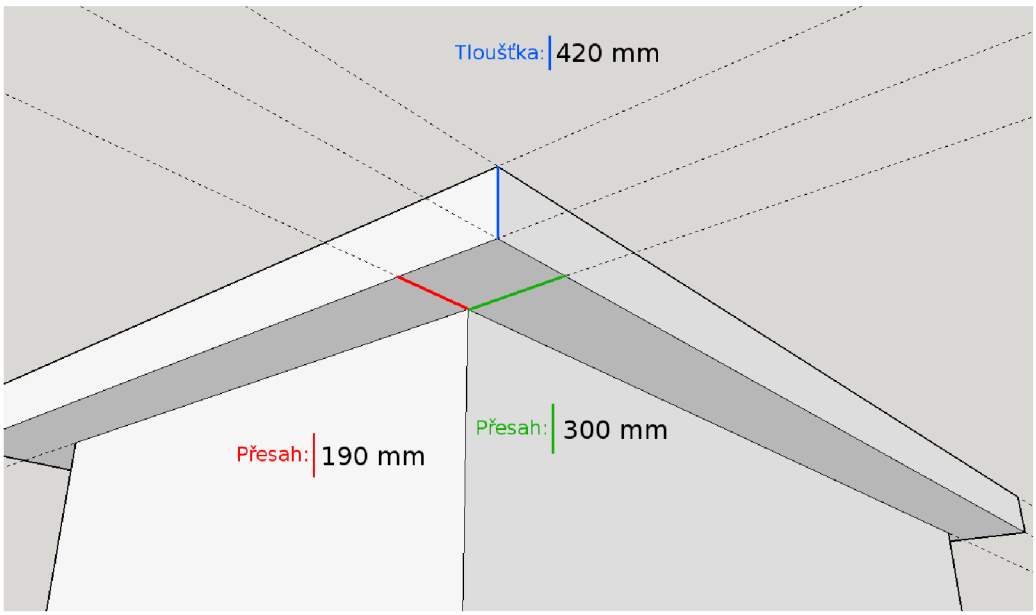
Abychom předešli problému slepování stěn a hran, viz kapitola 3.2, vytvoří se automaticky z desky komponenta.

Protože mají zdrojové desky určené tloušťky, může s nimi nástroj pracovat. Při tvorbě dílčích desek se sám přichytává na tyto míry, pokud se uživatel pohybuje v jejich okolí.

5.3 Automatická tvorba hranění

Při tvorbě nábytku z prefabrikovaných desek je nutné používat hranění pro ošetření řezných hran. Protože hranění vždy kopíruje hranu desky a má konstantní tloušťku, lze jeho tvorbu automatizovat. Cílem tohoto nástroje je umožnit uživateli vybrat požadované stěny, které budou mít jedno celistvé hranění, a po nastavení jeho parametrů jej jediným kliknutím vytvořit. Nástroj by také měl umožňovat nastavení přesahujících hran. Hranění bude vytvořeno jako skupina uvnitř komponenty.

Návrh uživatelského rozhraní prošel několika iteracemi. Jedna z možností byla zobrazovat uživateli všechny potřebné informace uvnitř samostatného okna s parametry. Původní myšlenka byla zobrazovat část nebo celý model, na kterém se vytváří hranění, uvnitř dialogu pomocí HTML5 komponenty *Canvas*. SketchUp, v případě operačního systému Windows, používá pro vykreslování webového rozhraní Internet Explorer. Nelze se však spolehnout, že všichni, dokonce ani většina uživatelů, bude mít verzi prohlížeče podporujícího HTML5. Toto omezení by však šlo překonat pomocí JavaScriptu, ale bylo by nutné vytvořit v podstatě novou projekci modelu a tu vykreslit pomocí klasických HTML elementů. Uživatelské rozhraní pak mohlo vypadat podobně jako na obrázku návrhu 5.1.



Obrázek 5.1: Návrh uživatelského rozhraní

Toto řešení však bylo nahrazeno jiným z několika důvodů. Jeho hlavní nevýhodou by byla nutná implementace nového vykreslení modelu. Dalším problémem by byla diverzita modelů. Každý model může vypadat zcela odlišně a může mít různý počet přesahů. Bylo

by proto komplikované realizovat rozhraní tak, aby vždy dávalo smysl a bylo přehledné, protože musíme předpokládat proměnlivý počet vstupních polí.

Obecně se jako lepší řešení návrhu uživatelské rozhraní ukázalo modifikovat přímo model aplikace SketchUp a *WebDialog* naprogramovat jen jako pole vstupních polí. Uživatel může dostat zpětnou vazbu změnou geometrie modelu a není nutné implementovat novou projekci.

Jediná překážka tohoto přístupu je, jakým způsobem spojit vstupní pole s jeho úsekem geometrie tak, aby uživatel nemusel zkoušet různé kombinace. Při samotné implementaci byly otestovány dvě možnosti řešení. První byly použity barvy pro rozlišení, tedy patřičná část geometrie byla při otevřeném UI přebarvena stejně jako její odpovídající vstupní pole. Toto řešení fungovalo, ale uživatelské rozhraní hrálo všemi různými barvami, při větším počtu přesahů. Jako elegantnější se však ukázalo, vždy při zapisování hodnoty do vstupního pole, označit upravovanou geometrii přímo v aplikaci SketchUp.

Kapitola 6

Implementace

Jak bylo popsáno v minulé kapitole, celé rozšíření stojí na konceptu zdrojových desek, které určují rozměry (tloušťku) a materiál jejich dílčích desek. Tyto vztahy jsou reprezentovány následovně. Jednotlivé instance třídy *Material*, které odpovídají existujícím materiálům v modelu aplikace SketchUp, uchovávají informace o všech zdrojových deskách, které jsou tvořeny právě tímto materiálem. Komponenty, představující dílčí desky, si zase pamatují ke které konkrétní zdrojové desce patří. Jak *Material*, tak i *ComponentDefinition* jsou potomci třídy *Entity*, je tedy možné k nim připojit slovníky. V *AttributeDictionary* jsou pak uložena veškerá potřebná metadata.

6.1 Uživatelské rozhraní

Hlavní uživatelské rozhraní propojuje celé rozšíření dohromady. Jeho vizualizaci můžeme vidět na obrázku 6.1. Vzhledem k nárokům na množství zobrazených informací a způsobu příjímání dat od uživatele, se jako jediná možnost jevilo použít *WebDialog* v kombinaci s několika objekty tříd *observer*. Už při startu aplikace SketchUp, kdy probíhá načtení rozšíření, se vytvoří instance uživatelského rozhraní. To je potřeba, protože každou komponentu, která patří k nějaké desce materiálu, hlídají objekty třídy *EntityObserver* a *EntitiesObserver*. Ty potřebují referenci na rozhraní, aby mu mohli dát vědět o případné změně.

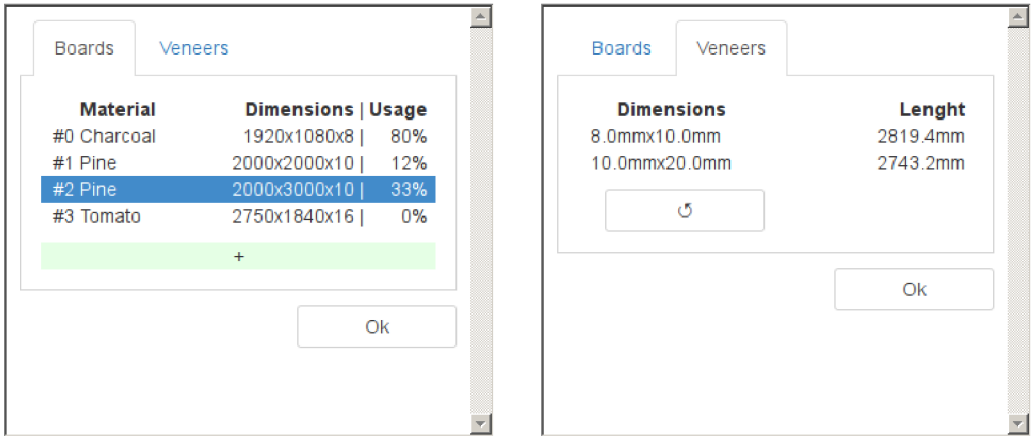
Objekty *EntitiesObserver* dávají pozor na úpravy v geometrii definic komponent dílčích desek. Po dokončení každé změny se přepočítá Eulerovo číslo, popsáno v kapitole 2.3, ověří se jestli je stále všechna geometrie propojena a zdali je dílčí deska stále kvádr. *EntityObsever* zase hlídá materiál komponenty a zdali nebyla smazána. Pokud změněná dílčí deska nemůže dále patřit do zdrojové desky materiálu, odebere patřičný *observer* z komponenty materiál a nastaví ji jako samostatnou. Uživatel tak ihned pozná, jaké dopady mají jeho změny.

Aby nemuselo dlouhodobě běžet mnoho objektů *observer*, kontroluje pouze jeden *SelectionObserver* aktuální výběr uživatele. Jakmile je označena alespoň jedna komponenta desky aktivují se všechny objekty *observer* pro hlídání geometrie.

Uživatelské rozhraní je vytvořeno v HTML, JavaScriptu a CSS s malou pomocí frameworku *Bootstrap*, jenž poskytuje třídy CSS pro menu. Celé rozhraní je pak postaveno jako dvě záložky, tvořící menu, které přepínají stránky. Hlavní obsah je seznam vytvořených desek, konkrétně jejich materiálem, rozměry a procentuálním využitím. Je zde také možné přidat novou desku pomocí tlačítka. To otevře nové okno, další *WebDialog*, kde je možné nastavit parametry nové desky a vytvořit ji. Druhá záložka menu ukáže seznam použitých hranění, jejich rozměry a délku. Funkcionalitu uživatelského rozhraní zajišťuje

JavaScript. Je implementováno jak přepínání obsahu v menu, tak možnost vybrat jednotlivé desky. Desky jdou označit jedním kliknutím, čímž informujeme Ruby skript, že další tvořená deska pomocí nástroje na tvorbu desek, by měla patřit právě této zdrojové desce. Nebo dvojklikem pro vybrání všech dílčích desek, jenž už do zdrojové desky patří. GUI na vybírání také reaguje a patřičně mění barevné označení položek. JavaScript zařizuje tyto funkce především měněním připravených tříd u elementů a voláním Ruby.

Abychom se vyhnuli dalšímu zatěžovaní běhu aplikace SketchUp více instancemi tříd *observer*, je na záložce s přehledem hranění, použito tlačítko pro aktualizaci obsahu. Není tak nutné hlídat veškerou manipulaci a tvorbu geometrie hranění.



Obrázek 6.1: Ukázka uživatelského rozhraní

6.2 Nástroj na tvorbu desek

Nástroj na tvorbu desek implementuje rozhraní *Tool* z Ruby API aplikace SketchUp, které bylo popsáno v kapitole 3.3. Nástroj je v podstatě konečný automat, kdy jeho stav je určen atributem *state*. Pokud je nástroj ve stavu, kdy uživatel vybírá počátek, tak se při každém pohybu myši vybírá nový aktuální bod, kam myš ukazuje. Pakliže je provedeno kliknutí levého tlačítka, použije se tento bod jako počáteční bod podstavy a přejde se do dalšího stavu. I zde se při pohybu myši najde aktuální bod, pomocí metody *pick* třídy *InputPoint*, a znehodnotí se aktuální pohled na model, čímž se vynutí jeho překreslení. V metodě *draw* je nutné nejprve najít vhodnou podstavu pomocí aktuálního a počátečního bodu, jak je detailně rozebráno v kapitole 6.2.1, jakmile je nalezena proběhne její vykreslení.

Další stisk levého tlačítka nás přenese do následujícího stavu. Nejprve se však z dočasné, pouze kreslené, geometrie stane skutečná a do modelu se přidá nová komponenta s vybranou podstavou. Poté pohyb myši určuje třetí rozměr desky. Nicméně, abychom mohli správně určit délku, je nutné nejprve získat třetí bod, který leží v jedné z rovin stěn desky, algoritmus určení bodu je detailně popsán v kapitole 6.2.2. Jakmile je vypočítaný i třetí bod, opět se znehodnotí pohled, aby mohlo proběhnout vykreslení drátěného modelu. Ten poskytuje uživateli potřebnou zpětnou vazbu při konstrukci desky, stejně jako se děje při výběru podstavy. Uživatel tak vidí, jak bude výsledek vypadat, aniž by se muselo překreslovat velké množství geometrie.

Poslední, levé kliknutí pak vytvoří 3D model desky. To znamená, že dokončí geometrii komponenty. Nástroj také ve druhém a třetím stavu patřičně aktualizuje hodnoty ve VCB

a umožňuje jejich přímé zadání číslem, skrze metodu rozhraní *onUserText*.

Zatím bylo toto rozšíření popsáno jako vhodné pro tvoření libovolných boxů. Pro tvorbu desek, pokud má uživatel vybranou zdrojovou desku, se využívá její známá tloušťka. Podle úrovně přiblížení pohledu na model se kolem dané tloušťky vytvoří interval, ve kterém se přichytí kurzor myši. Pokud poté vytvářím podstavu nebo třetí rozměr (výšku), automaticky se použije středová hodnota intervalu, pokud je délka, kterou určuje okamžitá pozice myši, právě v tomto intervalu. V praxi je pak jednodušší vytvořit několik desek stejné tloušťky. Toto přichytávání je možné vypnou/zapnout opakovaným stiskem nebo podržením/puštěním klávesy *CTRL*.

6.2.1 Nalezení vhodné podstavy podle počátečního a aktuálního bodu

K jednoznačnému určení podstavy bychom potřebovali tři body, jak ale můžeme vidět na existujícím nástroji *Rectangle*, není nutné určovat všechny tři body, stačí získat jen dva. Oba body, počáteční i aktuální, jsou vybrány pomocí třídy *InputPoint* která vybírá entity, jenž leží pod současnou pozicí kurzoru. Samotné vybrání realizuje metoda *pick* třídy *InputPoint*, ta může být volána několika způsoby, podle parametrů. Základní vybere bod jen podle získaných souřadnic. Druhá možnost je dát metodě jako parametr navíc i další instanci *InputPoint* s už vybranými souřadnicemi. Metoda pak například využije jeden z vektorů globálních os a najde nový bod ve stejném směru od předaného bodu. Důležité je, že metoda *pick* vybírá 3D bod na základně dvou souřadnic, proto není jasné který bod se má vrátit. SketchUp využívá svůj *inference engine* aby uživateli napověděl, který bod zvolil.

Pakliže máme samotné dva body, už jen stačí najít vhodnou rovinu na které oba leží. Zde si algoritmus nejprve ověří, zdali body leží na už existující rovině. Pokud ano, vezme se tato rovina jako základ pro podstavu. Nicméně když neleží na, v modelu se vyskytující, rovině, najde se podstava ve směru některé z hlavních os modelu. Největší prioritu má osa x, poté y a poslední se pracuje s osou z.

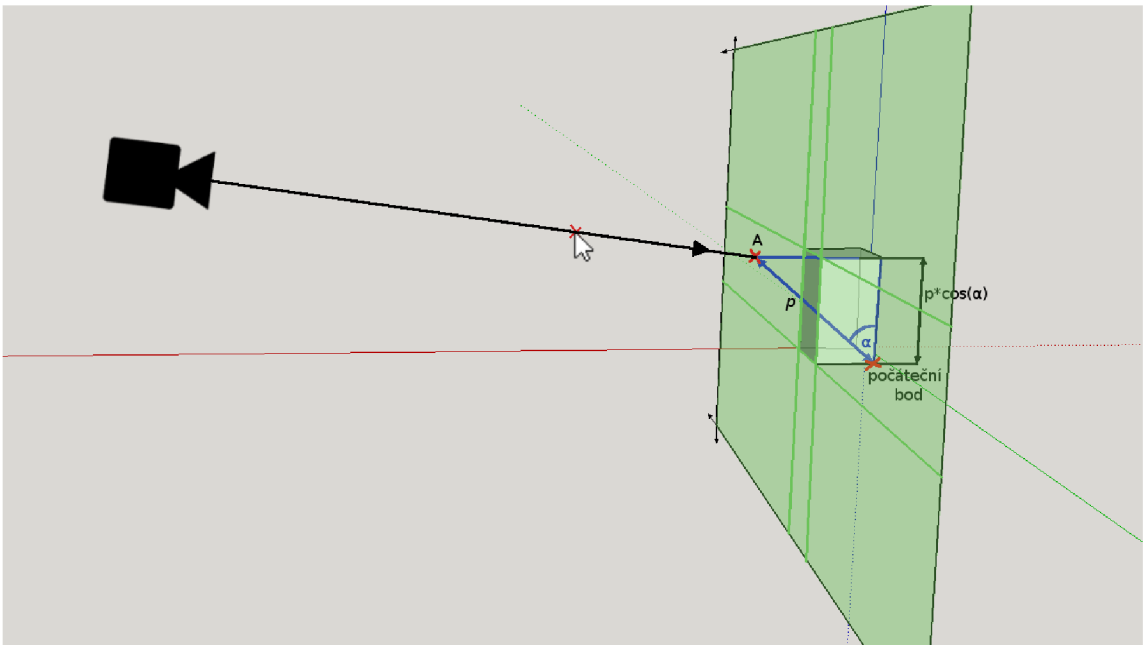
6.2.2 Nalezení třetího bodu - promítání paprsku z kamery do roviny

Jako zajímavý problém, z hlediska určení pozice v 3D prostoru, se ukázalo vybrání třetího bodu pro zjištění výšky. Pokud budeme bod hledat stejným způsobem jako doposud, tedy pomocí *pick* z třídy *InputPoint*, nastane problém, jestliže kolem modelu desky nebudou žádné entity jako hrany nebo stěny. *Pick* totiž může získat bod, který leží daleko od modelu desky, navzdory tomu, že jej uživatel vidí ve stejné rovině. Změna pozice kamery odhalí, že bod leží zcela mimo. Kdybychom pak počítali vzdálenost takového bodu od podstavy, získáme obrovské nepoužitelné hodnoty.

Jako nejlepší nalezené řešení se jevilo spočítat nový bod pomocí průniku roviny některé stěny desky a paprsku vrženého z pozice kamery, ve směru pohledu, skrze pozici kurzoru. Ilustrováno na obrázku 6.2, kde paprsek z kamery, procházející pozicí kurzoru, protne rovinu, jedné ze stěn tvořeného kvádru, v bodě A. Počáteční bod a bod A tak určují úsečku p a úhel α, který svírá úsečka s normálou podstavy. Pomocí vzorce $v = p * cos(\alpha)$ pak můžeme vypočítat odpovídající výšku tvořeného tělesa.

6.2.3 Komponenty, jejich geometrie a tvorba

Komponenty, stejně jako skupiny, mají vlastní systém souřadnic. Při tvoření komponenty skrze Ruby API je nutné dávat pozor na geometrii, kterou do ní přidáváme. Cílem je mít geometrii uvnitř komponenty co nejblíže počátku a celou komponentu poté transformovat na



Obrázek 6.2: Vizualizace nalezení bodu pro výpočet výšky

určené místo ve vnějším systému souřadnic. Když tedy tvoříme podstavu desky, počáteční bod má nulové souřadnice, leží v počátku os, a místo kam uživatel klikl v globálním systému se použije k získání transformace do toho bodu.

Vzhledem k ostatním nástrojům a obecně další práci s komponentou je žádoucí, aby deska ležela v kladných poloosách vnitřního modelu komponenty. Tedy je potřeba geometrii uvnitř komponenty přesunout v případě, že je vytvořena v záporném směru některé osy. Toho je docíleno pomocí zrcadlení skrze osy nebo rotací pro třetí kvadrant.

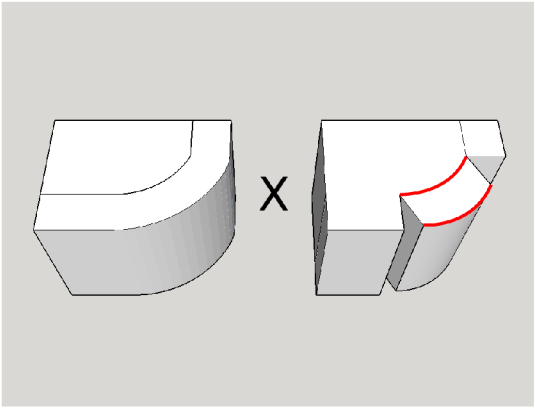
Ve výskledku se pak obě transformace sečtou a výsledná transformace se použije na celou komponentu.

6.3 Tvorba hranění

Hranění je generováno samostatným nástrojem. Základní vstup Ruby skriptu tvoří aktuální selekce uvnitř aplikace SketchUp. Na první pohled by se mohlo zdát, že stačí pouze na všechny vybrané stěny použít *PushPull* nástroj a vytáhnout je požadovaným směrem o patřičnou hodnotu. Nicméně toto řešení má zásadní nedostatek, protože nefunguje na libovolně zaoblené plochy, což je dáno čistě limitací *PushPull* nástroje. Problém, pokud zaoblenou plochu přímočaře zkopírujeme, je že nemůže plynule navazovat na vedlejší zkopírované roviny, zobrazeno na obrázku 6.3. Chybějící geometrie by pravděpodobně šla dopočítat. Bylo by potřeba dotvořit zahnuté hrany tak, aby navazovaly na vedlejší stěny. Nicméně lepší řešení je pracovat přímo s jednotlivými body, jenž určují hrany rovin. To je možné díky způsobu, kterým SketchUp reprezentuje křivky.

Tvorbu hranění pak lze provést následovně. První je potřeba identifikovat ohraničení výběru, podrobněji v kapitole 6.3.1 a pracovat s ním jako se sérií bodů. Tím se ovšem částečně ztratí informace o rozích a stěnách. Pro vykreslení správného počtu vstupů v uživatelském rozhraní, je potřeba zjistit kolik rohů má výsledné hranění mít 6.3.2. Poté už nám nic nebrání vytvořit uživatelské rozhraní, popsané v kapitole 6.3.4. Uživatel si zvolí

velikosti přesahů a tloušťku, tyto hodnoty se načtou do Ruby a uloží do pole. Následně jsou vytvořeny zbylé body, které dohromady tvoří geometrii hranění. Původní body se zkopírují a posunou nejprve tak, aby model hranění měl zadanou tloušťku a potom společně s původními ve směru přesahů o uložené hodnoty získané z uživatelského rozhraní. Změna pozic bodů je detailněji rozvedena v kapitole 6.3.3. K dokončení geometrie už stačí jen správně vytvořit stěny hranění, s touto problematikou se seznámíme také níže v 6.3.5. Nakonec se uloží informace o konkrétním hranění, jako délka a rozměry, do jeho slovníku, aby se mohli snadno zobrazit v přehledové části hlavního uživatelského rozhraní.



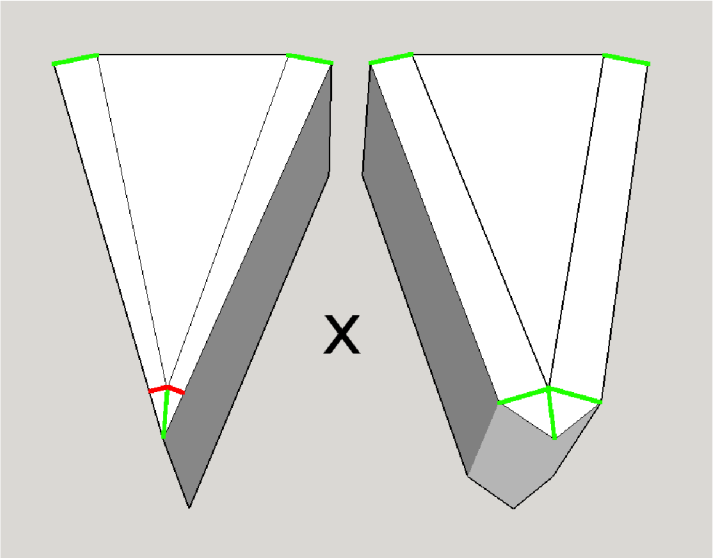
Obrázek 6.3: Ukázka chybějící geometrie, při jednoduchém kopírování stěn

6.3.1 Ohraničení výběru

První se ověří, zdali je celý výběr propojený, tedy jestli uživatel nevybral některé oddělené části. Pokud ano, použije se největší propojený celek, tedy ten který obsahuje nejvíce stěn. Musí se ověřit, že výsledný výběr není kruhově uzavřený. Jestliže je, je nutné ho v jedno místě rozdělit. Skutečné hranění se také nedá vytvořit jako uzavřený kruh. Pak proběhne vyfiltrování entit, které nejsou stěny a samotný algoritmus, jenž najde okrajové hrany označené geometrie. Algoritmus bere postupně každou stěnu a najde její hrany, ale vynechá přitom měkké (*soft*) hrany, ty SketchUp používá na tvorbu prohnutých stěn. Ze získaných hran pracuje dále pouze s těmi, které jsou okrajové, tedy které patří jen jedné stěně z výběru. Takhle se postupně vytvoří pole unikátních hran, které dohromady tvoří ohraničení selekce. Získané pole nemá garantované pořadí.

V dalším kroku se neseřazené pole hran transformuje na pole bodů, jdoucích postupně po sobě. První se náhodně najdou dvě startovní, na sebe navazující hrany. Od nich se pokračuje jedním směrem a vždy se hledá hrana která navazuje, tak aby výsledné pole bylo seřazené. Zároveň se vytváří i druhá křivka, to znamená pole bodů posunuté o tloušťku hranění, a to následujícím způsobem. Pokud se posunuje rohový bod, jenž patří právě jedné stěně, posune se bod ve směru normály stěny na které leží. Jestliže bod leží na hranici dvou stěn, posune se ve směru lineární kombinace normál obou stěn. Je ovšem nutné dávat pozor na úhel, který hrany svírají. V případech kdy je úhel příliš ostrý, to znamená menší než $\pi/4$, je nutné vytvořit dodatečnou geometrii, ilustrováno na obrázku 6.4. Přidáním dvou dalších bodů se rozdělí příliš ostrý úhel. Nové body se současně přidají i do původního pole bodů, kde jsou sice všechny tři stejné, mají stejné souřadnice, ale zachová se tak symetrie mezi křivkami a každý bod má svůj protějšek. Výsledkem jsou tedy dvě uzavřené křivky, jedna

je téměř totožná s původním ohraničením selekce a druhá, nová, leží nad původní ve výšce tloušťky hranění.



Obrázek 6.4: Chyba tloušťky hranění, při ostrých úhlech mezi stěnami

6.3.2 Nalezení rohů

Tento problém je řešen následovně. Postupně se prochází pole bodů, tvoří se vektory z bodů sousedních a kontrolují se úhly mezi nimi. Pokud je úhel příliš ostrý (menší jak $\pi/4$) je označen za roh. Složité je určit správně velikost úhlu, kdy se ještě jedná o zakřivenou plochu a kdy o roh v geometrii. Ve skutečnosti tento úhel nelze zcela správně určit. Zakřivené plochy v aplikaci SketchUp mohou mít libovolně ostrý úhel mezi jednotlivými stěnami, které ji tvoří. Avšak pro běžné použití se jeví jako vyhovující použít úhel $\pi/4$. Možné řešení toho problému by bylo blíže zkoumat původní geometrii. Pokud by v původní geometrii byl zlom součástí zakřivené plochy a nebyl by zrovna na okraji, mohli bychom poměrně bezpečně prohlásit, že se nejedná o roh.

6.3.3 Manipulace s body ohraničení

Celý algoritmus manipulace bodů, podobně jako u hledání rohů, vychází z cyklu, který postupně prochází všechny seřazené body. Protože mají obě křivky stejný počet bodů, můžeme s nimi pracovat zároveň. Vstupní hodnoty míry přesahů jsou uloženy postupně v poli, musíme proto začít na rohu křivek, kde se hodnoty mění. První krok je tedy najít počáteční roh. Od něj postupně procházíme a posunujeme body vždy o hodnotu získanou z pole pro aktuální přesah. Směr posunutí udává lineární kombinace normál dvou sousedících rovin, které vznikají z obou křivek tak, aby posunovaný bod byl právě mezi nimi. Přechod z jednoho přesahu do dalšího je oddělen rohem. Ty opět najdeme pomocí úhlu, který svírají roviny. Pakliže dojde ke změně, tedy narazíme na roh, musíme stejný bod posunout dvakrát. Pokaždé ve směru a o míru daného přesahu, protože rohové body patří vždy do dvou přesahů.

6.3.4 Uživatelské rozhraní

I při vytváření hranění se jako základ uživatelského rozhraní využívá třída *WebDialog*, blíže popsaná kapitole 3.6. Rozhraní vždy nabízí vstupní pole, pro změnu tloušťky hranění. Dále se zobrazuje proměnlivý počet vstupních polí pro přesahy. Jejich počet je určen podle počtu rohů, viz kapitola 6.3.2. Zobrazení proběhne ihned po spuštění skriptu, souběžně s vykreslením dočasného drátěného modelu do geometrie aplikace SketchUp. Drátěný model představuje uživateli jak bude výsledné hranění vypadat. Další výhodou použití modelu, jenž obsahuje jen jednoduché hrany, je jeho rychlé překreslení. Toho je využito, pro poskytnutí okamžité zpětné vazby uživateli. Vždy když dokončí zadávání hodnoty do vstupního pole, model se překreslí, aby respektoval novou hodnotu.

Aby uživatel poznal pro který přesah právě zadává hodnotu, vždy když vstoupí do editačního módu vstupního pole, v modelu aplikace SketchUp se označí patřičná část geometrie. Tato funkcionalita je implementována polem, jehož indexy odpovídají jednotlivým přesahům a hodnoty obsahují seznam geometrie tvořící přesah. Toto pole je vytvořeno už při první konstrukci modelu hranění.

Protože hranění může mít velké množství jednotlivých přesahů, podporuje rozhraní také zrychlené zadaní. Pokud uživatel nechá políčko prázdné, to znamená neobsahuje ani hodnotu nula, použije se hodnota ze souměrného přesahu nebo pokud je například zadaná jen jedna hodnota, použije se pro všechny přesahy.

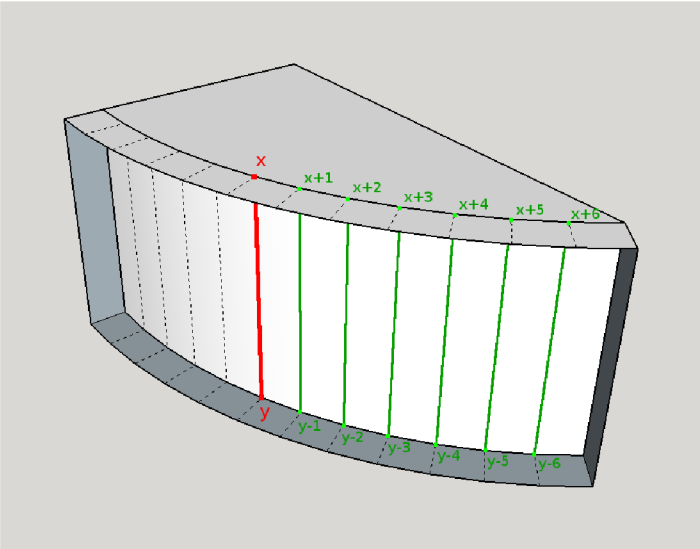
6.3.5 Vyplnění stěn

Poslední krok spočívá ve vytvoření stěn. Máme nachystané dvě uzavřené křivky, pole bodů, které jsou korektně umístěné podle nastavení uživatele. Vyplnit boky hranění, to znamená stěny, které spojují obě křivky, je s ohledem na zbytek řešených úloh vskutku jednoduchý problém. Stačí jít postupně po stejných indexech bodů obou křivek, postupně je spojovat a vytvářet vzniklé stěny. Jediná nepříjemnost může nastat pokud všechny čtyři body nejsou v jedné rovině. Tento případ je nutné testovat a případně rozdělit takový čtyřúhelník na dva trojúhelníky. Trojúhelník vždy tvoří rovinu.

Druhou část boků hranění tvoří přední a zadní stěny, můžeme je sice pro naše potřeby brát jako identické, ale jejich vyplnění je komplikovanější. První se ověří zdali hranění není natolik jednoduché, že jeho stěny tvoří dvě roviny. Pokud nejde vyplnit stěny rovinami, je nutné nejprve zjistit, jak mají být měkké hrany, které budou tvořit zakřivené stěny, orientovány. Toho je docíleno nalezením tak zvané startovní hrany. Startovní hrana je nějaká hrana, jenž je součástí některé z původně vybraných stěn, ale neleží přitom na ohraničení selekce. Využijeme indexy startovní hrany na křivce jako počáteční a vytváříme další, měkké, hrany s tím, že bereme v úvahu dva vzestupné/sestupné po sobě jdoucí indexy, zobrazeno na 6.5. Toto provedeme pro oba směry od startovní hrany i obě stěny hranění. Výhodou tohoto postupu je, že celou funkci tvorby měkkých hran pro opačný směr, získáme jednoduše zavoláním s prohozenými indexy startovní hrany.

6.4 Práce s jednotkami délky

SketchUp má propracovaný systém zprávy jednotek [11]. Jeho vnitřní jednotka jsou palce. Jakákoliv délka či souřadnice je vždy uložena v palcích. Tedy vždy když se jednotky zobrazují převedou se první do aktuálních jednotek modelu. A naopak pokud uživatel nějaké hodnoty zadá, převedou se opět do palců. Při tvorbě rozšíření stačí tato pravidla respekto-



Obrázek 6.5: Tvorba měkkých hran pomocí startovní hrany

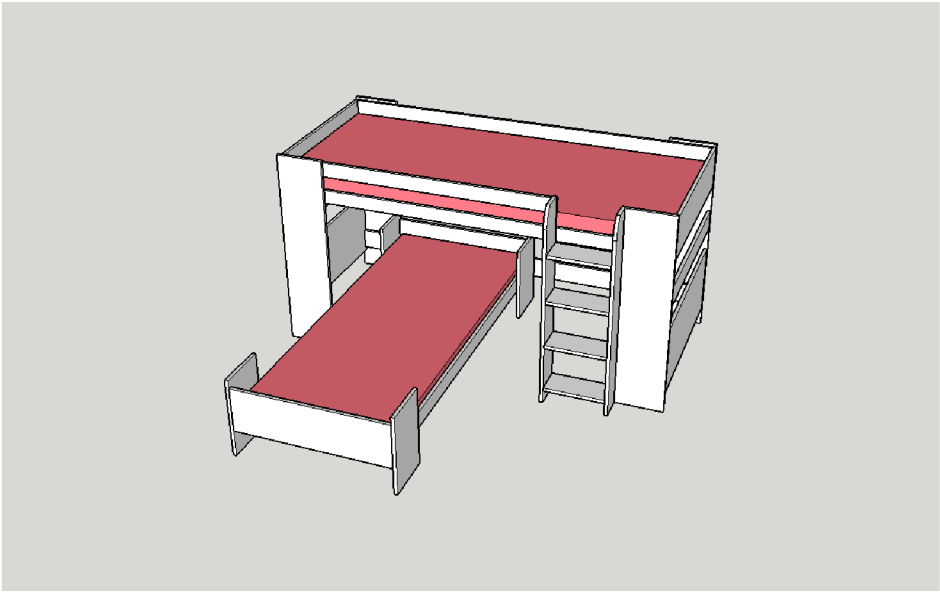
vat. Při zadávání hodnot použít speciální metodu *to_l*, pro převod řetězce na objekt *Length*. *To_l* má několik výhod, u samostatných čísel předpokládá, že jsou v jednotkách modelu, ale pokud je v řetězci dopsaná jednotka, například "50mm" nebo "50´ ", automaticky ji rozpozná a správně převede. *Length* je třída, která vše konvertuje na palce a zvládá základní operace a porovnání. Je také vhodné využít její metodu *to_s* pro převedení zpět na řetězec, opět se nemusíme starat o jednotky, jsou implicitně připojeny.

Ukládat *Length* hodnoty jako řetězce není vhodné. Protože uživatelé mohou používat desetinnou čárku nebo tečku, pro oddělení desetinné části čísla. Správný postup je vždy první číslo převést pomocí *to_f* na hodnotu s pohyblivou řádovou čárkou a až poté použít *to_s*, *float.to_s* vždy používá desetinnou tečku. Stejně tak *string.to_f* očekává desetinnou tečku.

Kapitola 7

Vyhodnocení přínosu rozšíření

Protože je netriviální přímo kvantifikovat přínos navrženého rozšíření ve formě jednoho ukazatele, byl použit následující scénář. Vyhodnocení probíhalo jako porovnání rychlosti a efektivnosti modelování s a bez použití rozšíření. Jako výchozí model byla zvolena dvoupatrová postel, viz obrátek 7.1. Měřil se celkový počet kliknutí a celková uražená vzdálenost kurzoru. Jako validní vzorek samozřejmě nejde považovat jednoho uživatele, protože výsledky jsou hodně ovlivněny schopnostmi a zkušeností modeláře. Abychom předešli tomuto problému, byl zvolen vzorek čtyř uživatelů s různou úrovní znalostí aplikace SketchUp. Výsledky shrnuje tabulka 7.1. Měření probíhalo na systému Windows 10, SketchUp Make 2015. Pro sběr dat kurzoru byl použit program Mousotron [8].



Obrázek 7.1: Model použitý jako vzor pro vyhodnocení

Z výsledků vidíme, že při tvorbě tohoto konkrétního modelu, má skutečně rozšíření pozitivní přínos. U počtu kliknutí došlo k průměrnému procentuálnímu poklesu o 32.822% a u uražené vzdálenosti o 21.592%. Ve všech případech kleslo množství kliknutí i uražená vzdálenost. Nelze však tvrdit, že každému uživateli rozšíření pomohlo stejnou mírou. Například pro uživatele 1 je vidět velké zlepšení, což je ale určitě dáno i jeho zlepšením práce s aplikací SketchUp. Naučení se několika jednoduchých klávesových zkratek, pro nejčastěji

Tabulka 7.1: Výsledky měření

		počet kliknutí			uražená vzdálenost (m)		
uživatel	úroveň	bez	s	zlepšení (%)	bez	s	zlepšení (%)
1	první použití	892	443	50.448%	212,88	99,78	53.129%
2	začátečník	801	604	24.594%	129,21	106,93	17.243%
3	pokročilý	503	381	24.254%	73,40	63,69	13.229%
4	pokročilý	472	321	31.992%	61,11	59,42	2.765%

používané nástroje, může obrovským způsobem ovlivnit celkovou vzdálenost uraženou myší. U pokročilých uživatelů zase hraje velkou roli jejich metodika a postupy tvorby modelu. Faktem je, že každý uživatel může mít naučené různé, odlišné postupy, které můžou přinášet různé výsledky.

Další zajímavé data by mohlo poskytnout měření s odlišnými modely nebo kombinace s jinými rozšířeními.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo blíže se seznámit s aplikací SketchUp, procesem návrhu nábytku a tvorby rozšiřujících modulů, dále také navržení a vytvoření takového modulu. Ačkoliv existuje řada rozšíření, neexistuje žádné, které by zefektivnilo modelování nábytku z prefabrikovaných desek. Protože je v dnešní době nábytek tvořený z prefabrikovaných desek populárnější než drahý dřevěný masív, má takové rozšíření velký potenciál.

Rozšíření tvoří dva nástroje a hlavní uživatelské rozhraní. V hlavním uživatelském rozhraní je možné vytvářet zdrojové desky různých rozměrů a materiálů. První nástroj pak tvoří modely dílčích desek a dokáže převzít informace ze zvolené zdrojové desky jako její tloušťku a materiál i je aplikovat při tvorbě dílčí desky. Druhý nástroj je specializovaný na generování hranění desek.

Rozšíření bylo navrženo s ohledem na existující pluginy, s cílem předejít duplicitě systémů. Dokonce se podařilo zajistit spolupráci vzájemně se doplňujících funkcí. Projekt byl například rozšířen o práci s materiály, aby se zajistila kompatibilita s nově vzniklým rozšířením, pro umístění dílčích desek na přířez.

Při vyhodnocení výsledků rozšíření se ukázalo, že uživatelé, kteří jej používali, dosáhly průměrně procentuální úspory 21.592% uražené vzdálenosti myší a 32.592% v počtu kliknutí.

V plánu je také zveřejnit rozšíření na platformě Extension Warehouse, kde bude dostupné pro uživatele. Ti jej budou moci použít při tvorbě vlastních modelů a zefektivnit tak svou práci.

V projektu jsem se přesvědčil, jak podstatná je dobrá dokumentace k API uzavřených systémů. Přestože SketchUp poskytuje přehlednou a obsáhlou dokumentaci i malé nepřesnosti mohou vést k značným problémům.

Z hlediska budoucího vývoje je rozšíření velmi perspektivní, například ve vylepšování vytvořených nástrojů. Hlavní zastřešující uživatelské rozhraní by mohlo umožňovat zobrazení více detailů nebo například měnit materiál hranění hromadně. Nabízí se také přidání různých nastavení jako třeba možnost pojmenovat každou vytvořenou dílčí desku. Algoritmus na tvorbu samotného hranění by určitě bylo možné optimalizovat, neboť při generování rozsáhlejších modelů se může projevit jeho časová náročnost. Hodně možností je také v přidávání nových nástrojů. Ať už se jedná o automatizovanou tvorbu spojů a šroubů nebo například generování desek se zaoblenými hranami.

Literatura

[1] How to create furniture in Google SketchUp[online].
http://squaresquirrel.ro/mac/how-to-create-furniture-in-google-sketchup/, 2010-04-01
[cit. 2016-05-04].

[2] Medium-density fibreboard[online].
https://en.wikipedia.org/wiki/Medium-density_fibreboard/, 2016-02-29
[cit. 2016-04-11].

[3] Turning SketchUp 3D Furniture into Real Furniture[online].
http://squaresquirrel.ro/miscelaneous/turning-sketchup-3d-furniture-into-real-furniture-into-real-furniture/, 2016-04-02
[cit. 2016-05-05].

[4] Drawingelement[online].
http://www.sketchup.com/intl/en/developer/docs/ourdoc/drawingelement.php, [cit. 2016-03-01].

[5] Zkratky MDF, LTD a DTD na nábytku[online].
http://www.ketyban.cz/clanky/zkratky-mdf–ltd-a-dtd-na-nabytku/, [cit. 2016-04-11].

[6] Group[online]. http://www.sketchup.com/intl/en/developer/docs/ourdoc/group, [cit. 2016-05-08].

[7] Introducing Drawing Basics and Concepts[online].
https://help.sketchup.com/en/article/3000083#know-inference, [cit. 2016-05-11].

[8] Mouse and Keyboard Usage Measurement[online].
http://www.blacksunsoftware.com/mousotron.html, [cit. 2016-05-08].

[9] Chopra, A.: Take the edge off: RoundCorner[online].
https://blog.sketchup.com/sketchupdate/take-edge-roundcorner, 2010-07-16
[cit. 2016-05-08].

[10] Howard, A.: *Elementary linear algebra*. Wiley, 2004, iSBN 978-1-118-43441-3.

[11] Thomassen, T.: Dealing with Units in SketchUp[online].
http://www.thomthom.net/thoughts/2012/08/dealing-with-units-in-sketchup/,
2012-08-17 [cit. 2016-05-08].

[12] Watt, A.: *3D Computer Graphics*. Addison-Wesley Publishing Company, 2000,
iSBN 0-201-39855-9.

[13] Španěl, M.: Základy vykreslování 3D scény, perspektivní a paralelní projekce[PDF]. 2013 [cit. 2016-05-08].

[14] Španěl, M.; Hulík, R.: Reprezentace 3D objektů[PDF]. 2013 [cit. 2016-05-08].

Přílohy

Seznam příloh

Příloha A

Obsah CD

/plugin/ - zdrojové soubory rozšíření

/mathewWWPlugin.rbz - rozšíření připravené na instalaci pro aplikaci SketchUp

/bp.pdf - technická zpráva

/latex/ - zdrojové soubory technické zprávy