MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Studio pro grafický návrh podlahových krytin v interiéru

BAKALÁŘSKÁ PRÁCE

Marek Pasičnyk

Brno, podzim 2007

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Mgr. Petr Tobola, Ph.D.

Poděkování

Chtěl bych poděkovat Mgr. Petru Tobolovi, Ph.D. za vedení bakalářské práce a jeho cenné rady při řešení této práce. Také bych rád poděkoval celé své rodině i své přítelkyni za neocenitelnou podporu a trpělivost.

Shrnutí

Tato práce se zabývá problematikou návrhu a kalkulace plovoucích podlah v interiérech a následnou implementací v CAD systémech. Cílem bylo vyvinout uživatelsky jednoduchou aplikaci pro rychlý návrh interiérů napodobující klasické metody pokládky. Tím posléze docílit co nejpřesnější kalkulace materiálu a vizuálně věrného zobrazení v cílovém prostoru.

Klíčová slova

AEC, CAD, OpenGL, plovoucí podlahy, návrh interiéru, vizualizace interiéru, kladečské plány, FloorPlanner

Obsah

Kapitola 1

Úvod

Různé kreslicí metody patří v architektuře již odedávna k základním metodám přenesení představ na papír či jiné médium. Vznikají tak dvojrozměrné nákresy, které jsou nepostradatelné pro veškerá odvětví architektury. Nákresy plní velmi důležitou úlohu při plánovaní i realizaci staveb. Jsou totiž jakýmsi spojením mezi architekty a lidmi, kteří daný projekt realizují. Ruku v ruce s dvojrozměrným nákresem se často vytváří trojrozměrný model. A protože lidský mozek zpracovává sérii bokorysů, půdorysů a nárysů dost obtížně, zhmotňuje trojrozměrný model lidské představy podstatně lépe. S příchodem počítačů s pokročilejšími grafickými jednotkami bylo možné tyto dvě, do této doby oddělené, metody návrhu spojit a zároveň přidat čtvrtý rozměr – animaci. Od průlomového vynálezu světelného pera se tyto systémy a aplikace velice rychle vyvíjely a získaly z anglického *computer-aided design* zkrácené označení *CAD*. Tato zkratka však nezahrnuje jen aplikace určené pro stavebnictví, ale i pro ostatní odvětví, používající při práci rýsovací prkno. Pro systémy, navržené speciálně pro architekturu a stavebnictví, se pak vžila zkratka *AEC*, z anglického *Architecture, Engineering, Construction*.

CAD se zpočátku ve stavitelství používal výhradně k návrhům exteriérů. Bylo tomu tak zejména kvůli nízké kvalitě zobrazení, která byla pro exteriérové prostory dostačující, avšak pro interiéry nikoli. Dnes je již však situace jiná a existuje celá řada nástrojů a aplikací pro návrh interiérů. Dokonce v podstatě každá velká firma zabývající se vybavením interiérů poskytuje ke svému sortimentu i nějakou více či méně sofistikovanou aplikaci pro zobrazení kolekce svých výrobků v prostoru.

Vizualizace interiérů bývá často zásadní, ale zdaleka ne jediným využitím těchto aplikací. Pomáhají totiž zároveň přesněji specifikovat náklady na celý projekt. Proto je zde vždy snaha o vytvoření co nejpřesnějšího modelu, který následně může ušetřit velké množství financí. Tento problém existuje i u návrhu plovoucích podlah. Jednoduché kalkulace materiálu zde, kvůli předem těžko předvídatelnému odpadu, často selhávají. Proto je vytvoření aplikace pro řešení této problematiky velkým přínosem pro toto odvětví.

Tato práce se věnuje vývoji grafického studia pro vizualizaci a kalkulaci plovoucích podlah v interiéru. Cílem je vytvořit aplikaci, která jednoduchým způsobem vytvoří a zobrazí uživatelem nakreslenou místnost včetně zvolené plovoucí podlahy. Vzhled plovoucí podlahy je při tom co nejvíce přiblížen reálnému vzhledu v dané místnosti. Plovoucí podlaha je totiž vykreslena podle stejných pravidel, jakými se řídí reálná pokládka podlah. Tím lze docílit přesného zobrazení podlahy v prostoru a zároveň získat velmi přesné údaje o nákladech na realizaci.

Na začátku práce je podrobněji rozvedena problematika návrhu a kalkulace interiérů. Kromě toho je zde také popsán vývoj *CAD* aplikací a jejich využití v oblasti návrhu interiérů. Obsahem další kapitoly je popis a zhodnocení některých aplikací, zabývajících se podobnou problematikou. Ve čtvrté kapitole se práce věnuje algoritmům a postupům, které byly stěžejní pro vývoj aplikace *FloorPlanner*. Ta je popsána v následující kapitole. Vysvětlení implementací modifikovaných algoritmů jsou obsahem předposlední šesté kapitoly. Práci uzávírá závěrečná kapitola se zhodnocením přínosu aplikace.

Kapitola 2

Problematika návrhu interiéru

2.1 Teorie pokládky a kalkulace plovoucích podlah

Plovoucí podlahy jsou moderním prvkem současného architektonického prostoru. Jejich popularita stále roste a jsou vyvíjeny nejrůznější nové technologie a materiály pro jejich výrobu. Vzniká tak široká škála plovoucích podlah z celého světa, které se kvalitativně velmi liší. Cena těch nejinovativnějších a nejkvalitnějších řešení se pohybuje v řádu několika tisíc korun za metr čtvereční. Vzniká proto potřeba co nejpřesnějšího odhadu nákladů na realizaci daného projektu. Tyto moderní podlahy bývají navíc často pokládány v moderních atypických prostorech, kde bývá odhad dosti obtížný. Řešení nabízí aplikace *FloorPlanner*, jež vznikla v rámci této práce. Jejím úkolem je přiblížit odhad skutečným nákladům.

Odhad se většinou provádí jednoduchým přičtením prořezu k ploše místnosti. Prořez tvoří ořezané části podlahových dílců, které již nemohou být použity pro pokládku a tvoří nutný odpad. Většinou je odhadovaný prořez 5 až 7 %. Reálně však může kolísat mezi 3 a 15 %. Množství prořezu je závislé na mnoha faktorech:

- **Velikost místnosti** — množství prořezu je nepřímo závislé na velikosti místnosti. Ve větších místnostech, kde se dá položit větší procento celých podlahových lamel, bývá mnohem méně odpadu.

- **Velikost lamel** — s rostoucí velikostí podlahových dílců se rapidně zvyšuje i odpad. Navíc právě nejdražší celodřevěné plovoucí podlahy jsou vyráběny ve velmi dlouhých formátech podlahových lamel. Tato skupina podlah tvoří nejproblematičtější skupinu při odhadu nákladů.

- **Orientace lamel** — směr pokládky se ve většině případů řídí jednou z obvodových zdí. U zdí, které nemají shodný, nebo kolmý úhel ke zdi řídící, vzniká u každé řezané lamely nezanedbatelný odpad.

- **Balení podlah** — plovoucí podlahy nejsou dodávány po jednotlivých kusech, nýbrž v baleních po několika kusech. To se stává problémem zvláště u velkoformátových lamel, kdy ve středně velké místnosti může rozdíl jednoho balení přesáhnout i 15 % z finálních nákladů. Proto je třeba k tomuto faktu přihlédnout a odhadnout množství co nejlépe.

- **Styl pokládky** — existuje několik způsobů, jak lze lamely vůči sobě pokládat. Nejstriktnější *selský styl*, který dovoluje pouze přibližně poloviční překryv lamel v sousedících řadách, dosahuje největšího odpadu. Následuje *anglický styl*, kdy musí být překryv u všech lamel shodný. Nejběžnější je však *rustikální styl*, u kterého je nutné dodržet pouze minimální překrytí. Spotřebuje se tak i mnohem méně materálu.
- **Optický vzhled podlah** — plovoucí podlahy jsou dnes ozvláštňovány různými prvky, které mění jejich optické vlastnosti v ploše. Jejich hrany mohou být například zkoseny a tak vytvářet podélné či oběžné spáry. Různá orientace pokládky pak může prostor opticky zúžit, nebo rozšiřovat. Často se podlahy pro opticky harmonický vzhled orientují podle dveří či oken.
- **Ostatní aspekty** — kromě přibližně předvídatelných aspektů však existují i faktory, které při návrhu předvídat nelze. Mezi ně patří nedokonalý tvar zdí místnosti, lamely s výrobními vadami nebo chyby a nepřesnosti podlahářů při realizaci. Proto je vždy třeba brát výsledky kalkulace s rezervou.

2.2 CAD aplikace

Computer Aided Design neboli počítačem podporovaný návrh je souhrnným názvem pro aplikace nebo obor zabývající se projektováním či konstruováním na počítači. Jedná se tedy o přenesení rýsovacího plátna na obrazovky počítačů. Proto je zde zahrnuta celá řada oborově různorodých aplikací od elektrotechniky, strojírenství až po stavebnictví.

2.2.1 Historie a současný stav

Kreslicí aplikace se začaly vyvíjet už v padesátých letech po vynálezu světelného pera, avšak první *CAD* aplikace vznikla až roku 1962 díky *Ivanu Sutherlandovi*[1]. Větší rozšíření přišlo až v sedmdesátých letech s příchodem minipočítačů. I přes značný pokrok však grafika stále zůstávala vektorová a práce s aplikacemi byla uživatelsky dosti nepřátelská. Zlom přišel až v roce 1978 s příchodem rastrové grafiky. Přicházely i nové algoritmy v počítačové grafice, a tak například v roce 1980 vzniká modul pro *AEC/CAD* aplikaci *Arch Model*[2], který dokáže poprvé renderovat plochy objektů.

Po příchodu procesorů *Intel x86* dochází ke standardizaci hardware a grafické aplikace se postupně přesouvají i na osobní počítače. Vznikají editory jako *AutoCAD*, *VersaCAD* a jiné. Zvětšil se také význam trojrozměrného modelu, který původně tvořil spíše presentační funkci. V dnešní době je návrh ve dvou a třech rozměrech propojen, což přispívá k větší přesnosti finálních návrhů. Během této éry vznikají v počítačové grafice nové pokročilé metody jako jsou osvětlovací modely s podporou radiozity, sledování paprsku nebo kvalitní mapování a filtrování textur. Díky jejich implementacím v *CAD* aplikacích jsou dnes vizualizace modelů téměř fotorealisticky přesné.

1. Kreslicí program *Sketchpad* byl předmětem Sutherlandovy disertační práce.

Výrazným přínosem je i intuitivnější interakce člověka s počítačem. Už samotný přechod z rýsovacího prkna na počítač znamenal podstatnou úsporu času. Zpočátku dosti rozdílné aplikace se dnes vlivem konkurenčního prostředí stávají velice podobnými. Existují však dvě velice rozdílné kategorie *CAD* systémů.

- **Obecné CAD systémy** představují vývojově starší metody návrhů a svou koncepcí připomínají klasické rýsovací prkno. Jde o nejvíce rozšířenou skupinu produktů. Jejich hlavní výhodou jsou vzájemně přenositelné formáty a jednoduché ovládání. Nově se již ovšem objevuje snaha implementovat do obecných *CAD* systémů moduly, které je rozšíří o objektově orientovaný přístup. Typickými aplikacemi této třídy jsou *AutoCAD* nebo *Microstation*.
- **Objektově orientované CAD systémy** naproti tomu nabízejí diametrálně odlišný přístup k návrhu. Mají své vlastní formáty a zpětná kompatibilita s obecnými *CAD* systémy je jen částečná. Je zde mnohem kvalitněji provázáno dvojrozměrné zobrazení s trojrozměrným. Jednotlivé prvky návrhu jsou reprezentovány objekty a jejich skládáním vzniká výsledný model. Do této kategorie patří především velice kvalitní *AEC/CAD* systémy *ArchiCAD* nebo *ALLPLAN FT*.

2.2.2 AEC/CAD systémy

Tato kapitola se věnuje kategorii *CAD* systémů, do které spadá i aplikace *FloorPlanner*. Jedná se o *CAD* aplikace pro projektování v oblasti architektury a stavebnictví, které využívají objektově orientovaného přístupu. *AEC/CAD* aplikace nabízejí širokou škálu objektů, které se pak navzájem kombinují do výsledného modelu. Příkladem může být aplikace *ArchiCAD*. Zde jsou objekty umístěny do hierarchických struktur. *ArchiCAD* má pro tyto struktury vlastní jazyk *GDL*[2]. Objekty uvnitř těchto struktur na sebe inteligentně reagují, čímž dochází například ke spojování zdí, k výřezům ve zdech pro okna či dveře. Zároveň je možné veškeré atributy objektů kdykoli měnit. Výsledný model se přizpůsobuje.

Důležitou vlastností objektů jsou jejich nevektorové informace. Z nich lze totiž následně generovat výkazy či soupisy materiálů potřebných pro celou stavbu, nebo jen pro některou z jejich částí. Tímto lze stanovit předběžnou cenu projektu.

2. Geometric Description Language je "BASIC-like"programovacím jazykem.

Kapitola 3

Stávající aplikace pro návrh interiéru

Vizualizací interiérů se dnes zabývá celá řada aplikací. Konkurence mezi výrobci vybavení pro interiéry je velká a představivost zákazníků malá. Proto se tito výrobci snaží poskytnout zákazníkům možnost vizualizace jejich představ. Konkrétně v oblasti plovoucích podlah jde většinou o webové aplikace, které pouze určitým způsobem pracují s fotografiemi místností.

Aplikace pro přesnější návrh podlah, ale i obkladů, se dnes běžně používají pouze pro keramické dlažby. Dovolují vytvářet přesné kladečské plány a odhadnout i náklady. Obdobné aplikace pro plovoucí podlahy prozatím buď neexistují, nebo nejsou volně přístupné.

Další možností je využít jedno ze sofistikovaných *AEC/CAD* řešení. Aplikace této kategorie poskytují nástroje pro návrh kompletních staveb, tedy i interiérů a jejich podlah. Lze je doplnit i nábytkem a ostatním vybavením a tím vytvořit reálnou kopii realizované místnosti. Tyto aplikace sice většinou obsahují nástroje pro generování výpisů materiálů, ovšem u podlah jde pouze o plochu místnosti a tak problém s množstvím prořezu zůstává nevyřešen. Následuje několik aplikací, které se vizualizací plovoucích podlah zabývají. Jsou zde jednoduché webové aplikace i komplexní objektově orientované *AEC/CAD* řešení.

3.1 eShowroom

Aplikace *eShowroom*[1], která byla vyvinuta pro švédskou společnost *Pergo*, patří právě do kategorie jednoduchých webových aplikací. Byla vyvinuta pro prezentaci kolekce plovoucích podlah *Pergo*. Uživatel má na výběr ze tří různých místností, přičemž má možnost měnit barvu zdí, některého z doplňků a samozřejmě vzor plovoucí podlahy. Tyto tři místnosti jsou pouze tři fotografie, na které jsou pomocí masky nanášeny různé vzory či barvy. Uživatel tak nijak nemůže přizpůsobit prostor ani pohled v prostoru. Přínosem však je možnost barevného sladění a hlavně zobrazení plovoucí podlahy v ploše, což dává uživateli rozhodně lepší představu než jeden samostatný vzorek. Aplikace dokonce disponuje jednoduchým kalkulátorem, který po zadání několika údajů o místnosti přibližně vypočítá potřebné množství balení podlah a doporučí i barevně sladěné lišty. Tento výpočet lze však považovat za správný jen v místnostech střední velikosti a navíc obdélníkového tvaru.

1. Dostupná na adrese <http://www.europe.pergo.com/Consumer/Templates/eshowroom/CS_eshowroom.asp> (ověřeno k 6. lednu 2008).

Obrázek 3.1: Vizualizace plovoucí podlahy v aplikaci *eShowroom*.

3.2 Room Designer

Pokročilejší webovou aplikací je bezesporu *Room Designer*[2], vyvinutý pro německou firmu *Parador*. Místo přednastavených fotografií totiž umožňuje vložit vlastní. Pomocí krátkého průvodce, během něhož jsou jednoduchými kliknutími označeny oblasti podlahy a zdí, je fotografie připravena. Pro uživatele je k dispozici kompletní kolekce firmy *Parador*. Korekci perspektivy je možné provést pomocí malého lichoběžníku uprostřed. Posuvníkem pro nastavení průhlednosti podlahy lze na novou podlahu přenést odlesky té původní. Kromě podlahy lze označením stěn pozměnit i malbu na stěnách.

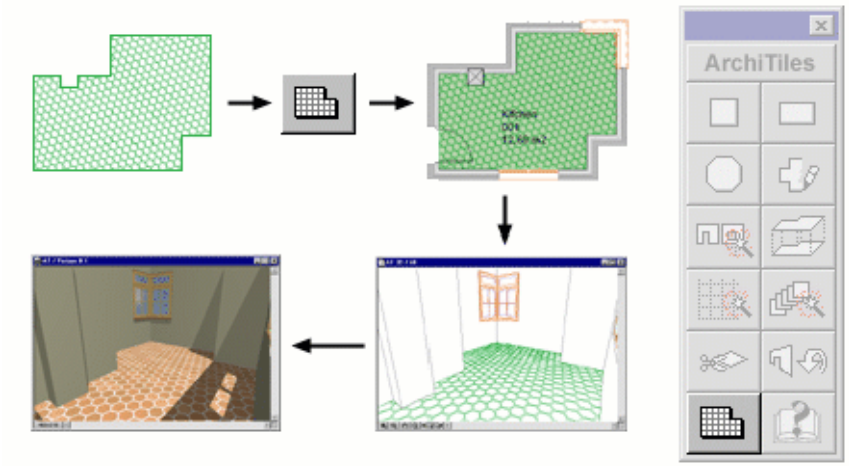
Obrázek 3.2: Porovnání místností s původní podlahou a podlahou vytvořenou aplikací *Room Designer*.

2. Dostupná na <http://www.parador.de/swf/raumdesigner/de.php> (ověřeno k 6. lednu 2008).

Aplikace sice umožňuje zobrazit podlahu v zamýšleném prostoru, avšak zpracování podlah není příliš kvalitní. Při nastavení perspektivy je obraz deformován bez následného vyhlazení, a tak se vzor podlahy spíše ztrácí v šumu. Bohužel tato aplikace neumožňuje jakýmkoli způsobem stanovit náklady na realizaci.

3.3 ArchiCAD a ArchiTiler

Aplikace *ArchiCAD* od společnosti *Graphisoft* je dnes jedním z nejpoužívanějších AEC/CAD systémů. Pro konstrukci budov využívá objektově orientovaného přístupu *BIM*[3]. V ArchiCADu se tato technologie používá pod názvem *Virtuální budova*. Celý model se sestává ze vzájemně propojeným konstrukčních prvků, jako jsou zdi, dveře, okna apod. Ty jsou pak v různých režimech zobrazení vykreslovány odpovídajícím způsobem. Díky tomu lze okamžitě zobrazovat reálný model v 3D, okótované výkresy ve 2D, ale i automaticky aktualizované tabulky s výkazy výměr a výpisy prvků. Součástí aplikace jsou rozsáhlé knihovny těchto prvků. Jsou popsány jednoduchým jazykem *GDL* a není tedy problém přidat prvky, které uživateli v knihovnách chybí. Výsledný projekt lze exportovat v mnoha formátech. Trojrozměrné vizualizace lze provádět renderingem přímo v aplikaci, nebo pomocí některého z externích rendererů. Pro vložení speciálních konstrukčních prvků lze využít jednoho z mnoha plug-inů.



Obrázek 3.3: Kladečské plány dlažby v plug-inu *ArchiTiler* aplikace *ArchiCAD*.

Pro práci s plovoucími podlahami lze jednoduše použít texturu na prvek podlahy. Pro kalkulaci a reálnější vzhled lze použít propracovaného plug-inu *ArchiTiles* firmy *Cigraph*. Ten je primárně určen pro kladečské plány a kalkulaci pokládky dlaždic a dlažby. Změnou tvaru a textury však lze rozšířit použití i na plovoucí podlahy. Je možné nastavit posun mezi dílci v jednotlivých řadách, jenž však může být pouze konstantní a tak nelze docílit rustikálního

3. Building Information Modeling.

stylu pokládky, kdy je dosaženo nižší spotřeby materiálu. Bohužel také vestavěný algoritmus nepočítá s použitím ořezaných dílů v dalších řadách. Jinak však *ArchiTiles* dokáže automaticky generovat výkazy materiálu včetně kalkulace prořezu.

Toto řešení je nejlepší ze zde presentovaných aplikací pro vizualizaci a kalkulaci plovoucích podlah. Dovoluje použití i pro jiné typy podlah. Díky rozsáhlým knihovnám *ArchiCADu* lze interiér vylepšit dle svých představ. Fotorealistické zobrazení a kalkulace včetně výpočtu prořezu jsou dalšími z mnoha výhod. Je zde ale i několik záporů. Aplikace *ArchiCAD* je primárně určena pro návrh kompletních budov a komplexnosti aplikace odpovídá i její cena, která se při zakoupení jedné licence i s plug-inem *ArchiTiles* pohybuje kolem 140 000 Kč. To je však pro firmy, hledající pouze program pro vizualizaci podlah, příliš nákladná a těžko vratná investice. Další nevýhodou je komplikovanost *ArchiCADu*, čímž je tato aplikace pro nezkušeného uživatele v podstatě nepoužitelná.

Kapitola 4

Použité algoritmy, metody a techniky

V průběhu vývoje aplikace *FloorPlanner* bylo zapotřebí vyřešit několik úskalí. Nejzásadnější problémy a hlavně metody pro jejich řešení jsou obsahem právě této kapitoly.

4.1 Triangulace konkávních polygonů

Triangulací se v počítačové grafice rozumí rozklad polygonu na trojúhelníky. Trojúhelníky se navzájem nesmějí překrývat a jejich sjednocení je rovno původnímu polygonu. Triangulací můžeme komplikované konkávní polygony rozdělit na jednoduché trojúhelníky, na nichž lze provádět některé operace, které bychom s původním polygonem nebyli schopni provádět. Trojúhelníky jsou totiž vždy rovinné, jednoduše se rasterizují a jsou jednoznačně interpolovatelné.

Triangulačních algoritmů je celá řada. Liší se časovou i paměťovou složitostí, komplikovaností implementace, ale i kvalitou podávaných výsledků. Některé vytvářejí aproximace rovnostranných trojúhelníků. Triangulace může také rozdělit polygon buď na trojúhelníky, které mají pouze vrcholy shodné s vrcholy původního polygonu, nebo na trojúhelníky s vrcholy kdekoli uvnitř nebo na okraji polygonu. V aplikaci *FloorPlanner* bude použita prvně zmiňovaná metoda, neboť je implementovatelná pomocí nástavbové knihovny *GLU* knihovny *OpenGL*. Právě v souvislosti s *OpenGL* je častěji používán obecnější pojem *teselace*[1], který označuje rozklad polygonu na lépe zpracovatelné struktury, většinou právě trojúhelníky. Prostorové objekty bývají děleny na čtyřstěny, nebo šestistěny.

4.2 Alias a jeho řešení

4.2.1 Alias

Alias je jedním z nejčastějších a nejdiskutovanějších problémů v současné počítačové grafice. Jde o obecný pojem pro vadu vznikající podvzorkováním vstupního signálu. Dochází k němu při porušení *Nyquistova kritéria*, které říká, že vzorkovací frekvence musí být nejméně dvakrát větší, než je největší frekvence ve vstupních datech. Toho však často na monitorech s běžným rozlišením nelze dosáhnout. Podvzorkování pak způsobuje zobrazení vysokých frekvencí do frekvencí nižších, což vede ke vzniku nepříjemných rušivých elementů. V aplikaci *FloorPlanner* je však kladen důraz na kvalitní zobrazení textur. Zároveň

1. Z latinského *tessella* neboli kousek mozaiky.

je také většinou používán pohled, kdy je *alias* velice markantní a nežádoucí. Existuje několik projevů *aliasu*, z nichž dva jsou v počítačové grafice nejběžnější:

- **Jaggies**[2]— pojmem *jaggies* se obvykle označuje jev příliš ostrého a zubatého přechodu na okrajích polygonu. Vzniká rasterizací polygonu do konečně velkého rozlišení. Vzhledem k tomu, že většina dnešních scén, včetně scén v aplikaci *FloorPlanner*, je tvořena polygonálními modely a následně převedena do rastru, vypadají jejich okraje, zvláště v kritických úhlech, dosti nevzhledně.
- **Moiré**[3] — při vzorkování textur s pravidelným vzorem může docházet k nepříjemnému efektu moiré, který se projevuje zobrazováním vzorku s vyšší frekvencí jako vzorku s frekvencí nižší. Vzorkování s vyšší vzorkovací frekvencí tento problém pouze odsune do frekvencí vyšších. V aplikaci *FloorPlanner* se textury s pravidelným vzorem asi příliš neobjeví. Avšak v případě vykreslení velkého počtu shodných polygonů vedle sebe se v místech, relativně hodně vzdálených od pozorovatele, tyto rušící artefakty běžně zobrazují.

Následující dvě části popisují metody, jak tyto dvě vady dostatečně eliminovat.

4.2.2 Antialiasing

Antialiasing je obecným pojmem pro odstraňování *aliasu*. Nejběžnější cestou je odstranění příliš vysokých frekvencí některým z vysokofrekvenčních filtrů. Tím však dojde k jistému rozostření obrazu. Možností je využít metodu *Multisamplingu*, která je implementována přímo v *OpenGL API*. Ta se hodí k odstranění zubatých hran a efekt moiré dokáže odsunout do vyšších frekvencí. Využívá při tom akumulační paměť. Obraz je tak vykreslen několikrát, přičemž se pohled po každém vykreslení změní o vzdálenost menší než jeden pixel. Výsledné obrazy jsou sečteny a zprůměrovány. Tato metoda je sice kvůli několikanásobnému vykreslování dosti náročná na výkon, avšak je jednoduché ji pomocí *OpenGL* implementovat a rovněž ji podporuje většina dnešních grafických karet.

4.2.3 Filtrování textur a mipmapping

Velkým problémem je i vznik *aliasu* na texturách. V prostorových scénách totiž většinou jednomu pixelu neodpovídá právě jeden *texel*[4]. Filtrování textur provede smíchání více texelů na jeden pixel. Bez použití filtrování je pixelu přiřazena barva jen jednoho texelu a tak při pohybu dochází k nepříjemnému mihotání a vzniku *jaggies*. Filtrování textur je výkonově levnější operací než použití *antialiasingu*. Pro další zrychlení lze zmenšení textury předpočítat. Slouží k tomu technika *mipmappingu*, která předem ke každé textuře vytvoří její zmenšené již filtrované verze. Takto je vytvořeno několik textur od původního rozlišení

2. Z anglického *jagged edges* neboli „zubaté“ hrany.
3. Podle tkaniny mající vzhled zčeřené vodní hladiny.
4. *Texel* neboli *texture element* je, obdobně jako pixel u obrazu, základní jednotkou textury.

až po velikost jedna. Při následném mapování textury se dle úrovně detailů zvolí příslušná *MIP map* úroveň. *OpenGL* podporuje pouze textury o velikosti mocniny čísla 2. To ovšem znamená, že pro každou *MIP map* texturu je zapotřebí čtyřnásobek paměti oproti původní textuře. Náročnost na paměť je i přes částečné rozmazání textury vyvážena mnohem lepším zobrazením. Nespornou výhodou je také výše zmiňované ušetření výpočetního času. Není totiž zapotřebí při každém nanesení textury provádět filtrování na textuře s původním rozlišením.

4.3 Řádkové vyplňování se seznamem aktivních hran

4.3.1 Specifikace požadavků

Nejdůležitější částí aplikace bylo bezesporu najít a implementovat vhodný algoritmus pro vyplnění oblastí. Jeho modifikovaná podoba by byla následně použita při vyplňování místnosti ve tvaru libovolného polygonu dílci plovoucí podlahy ve tvarech libovolných obdélníků. Algoritmus musel splňovat následující podmínky:

- **Nezávislost na typu polygonu** — algoritmus by neměl rozlišovat, zda je vyplňovaný objekt polygonem konvexním či konkávním.
- **Geometricky určené hranice** — základní podmínkou bylo také vybrat algoritmus, který vyplňuje oblast zadanou sérií propojených bodů.
- **Vyplňování v libovolném úhlu** — je nezbytné, aby bylo možné provádět vyplňování v obecném úhlu, neboť v aplikaci bude většinou úhel vyplňování shodný s jednou z hran polygonu.

Všechny tyto podmínky nejlépe splňuje algoritmus *řádkového vyplňování se seznamem aktivních hran*. Aby navíc fungoval v obecném úhlu, použijeme postup, jenž je podobný šrafování v obecném úhlu.[5]

4.3.2 Popis algoritmu

Je nutné zavést celkem tři datové struktury. Základním datovým typem je *hrana*. O každé hraně se uchovávají tři atributy: souřadnice y koncového bodu, souřadnice x průsečíku s aktuálním řádkem a přírůstek x při přechodu na nový řádek. Někdy se místo souřadnice y koncového bodu používá počet zbývajících řádků. V aplikaci *FloorPlanner* však nebudou souřadnice y celočíselné hodnoty, a proto by kvůli nepřesnostem ve výpočtech mohlo docházet k chybám.

Druhou datovou strukturou je *tabulka hran* obsahující na začátku všechny nevodorovné hrany polygonu. Vodorovné hrany jsou z výpočtu vyloučeny, neboť mají s řádkem nekonečně mnoho průsečíků a pro výpočet nejsou kvůli řádkovému stylu vyplňování důležité. Všechny hrany v tabulce hran jsou orientovány směrem dolů a jsou sestupně uspořádány podle souřadnice y počátečního bodu.

Poslední nutnou datovou strukturou je *seznam aktivních hran*. Při přechodu na nový řádek jsou všechny hrany s počátkem na aktuálním řádku přesunuty z tabulky hran do seznamu aktivních hran. Seznam se udržuje vzestupně seřazený podle aktuální souřadnice x každé z hran. V tomto okamžiku je možné vykreslit úseky mezi lichými a sudými hranami v seznamu. Před přechodem na nový řádek je u všech hran aktualizována hodnota souřadnice x průsečíku. Zároveň jsou hrany, mající souřadnici y koncového bodu shodnou se souřadnicí y aktuálního řádku, ze seznamu aktivních hran vyřazeny.

Algoritmus začíná naplněním a seřazením tabulky hran, přičemž je její první, a tím pádem i největší, souřadnice y použita pro nastavení aktuálního řádku. Přechody na nový řádek jsou poté opakovány, dokud se tabulka hran i seznam aktivních hran úplně nevyprázdní.

Kapitola 5

FloorPlanner

Aplikace *FloorPlanner* vznikla jako stěžejní část této bakalářské práce. Řadí se do kategorie jednoduchých objektově orientovaných *AEC/CAD* aplikací. Byla primárně vytvořena pro vizualizaci a kalkulaci plovoucích podlah. I přes její současnou funkčnost a využitelnost v praxi je možný, a zároveň důležitý, její další vývoj. Cesty, kterými by se mohl vývoj ubírat, jsou naznačeny na konci této kapitoly.

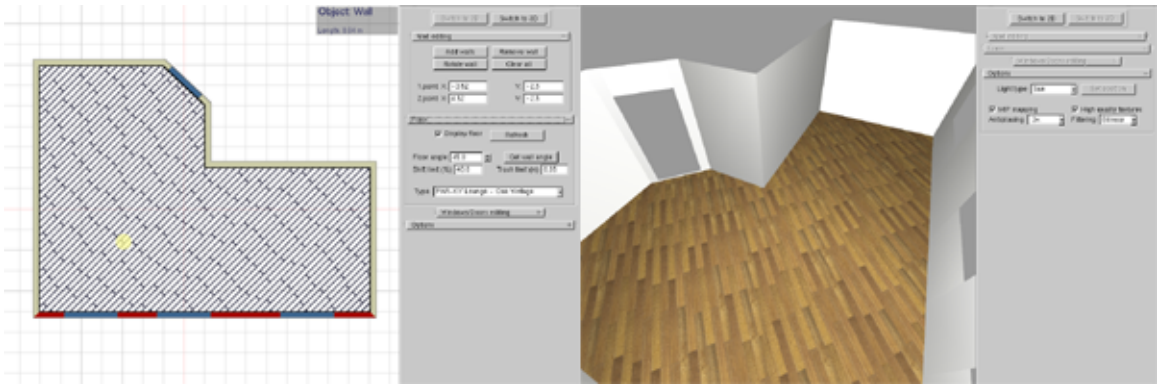
5.1 Uživatelské prostředí

Aplikace je navržena tak, aby uživatel mohl jednoduchými a intuitivními kroky vytvořit téměř libovolnou místnost. Do vytvořené místnosti pak „položil" jednu z nabízených podlah, poupravil pár parametrů pokládky a nakonec se prošel v hotové místnosti. Uživatelské prostředí se skládá ze dvou částí. V pravé části se nacházejí veškeré ovládací prvky. Levá část různými způsoby zobrazuje grafická data.

Ovládací prvky byly vytvořeny platformově nezávislou knihovnou *GLUI*. Jsou rozděleny do několika rozvinovacích panelů. Pomocí panelu *Wall editing* lze přidávat, odebírat i editovat zdi. Panel *Floor* se stará o nastavení a zobrazení podlahy. Panel *Windows/Doors editing* vytváří a odstraňuje otvory ve zdech pro okna a dveře. V panelu *Options* se nalézají ostatní nastavení, například pro změnu světla či kvality zobrazení. Pravá část mimo jiné obsahuje i tlačítka pro volbu jednoho ze dvou módů. Změnou módu se mění způsob zobrazování grafických dat v levé části.

Dvojrozměrný mód zobrazuje ortografický náhled půdorysu místnosti a je ideální pro editaci. Je navržen jako nápodoba rýsovacího prkna. Pozadí tvoří pomocná mřížka, na níž se místnost jednoduše kreslí pomocí myši. Pro vytváření zdí pod úhlem 90° a jeho násobku je implementováno, pro tuto činnost standardní, kreslení se stisknutou klávesou **Shift**. Zobrazují se zde i okna s různými doplňkovými informacemi. Po dokončení místnosti a zobrazení podlahy je v tomto módu nejlépe vidět rozložení podlahových dílců.

Druhou možností je trojrozměrný mód plnící presentační a vizualizační úlohu. Pro pohyb je použit pohledový model první osoby. Veškeré zdi jsou viditelné jen z vnitřní strany místnosti. Díky tomu je podlaha interiéru vidět i v případě, že je pozorovatel vně místnosti. V pravé části okna aplikace je možné nastavit několik úrovní kvality zobrazení tohoto trojrozměrného módu včetně typu osvětlení.



Obrázek 5.1: Ukázka uživatelského prostředí aplikace FloorPlanner.

5.2 Logická stavba programu

Aplikace byla naprogramována v jazyce *C++*. Tento jazyk byl vybrán pro snadnou implementovatelnost *OpenGL API*. Pro vytvoření objektově orientované *AEC/CAD* aplikace byl i v programování ve většině případů použit objektově orientovaný přístup. Hlavní třídou aplikace je třída *Room* obsahující veškerá grafická data o místnosti. Tato třída proto disponuje kontejnerem s instancemi třídy *Wall* pro zdi místnosti, instanci třídy *Floor* pro podlahu a instanci třídy *Light* pro osvětlení. Zároveň se stará o vzájemnou interakci komponent místnosti. Třída *Wall*, kromě grafických dat o samotné zdi, obsahuje také kontejner s instancemi třídy *Windows* pro výřezy ve zdech. Ke kalkulaci materiálu slouží třída *MaterialCalculation*. Uvnitř dochází také k výpočtu grafických dat podlahových dílců pro místnosti typu *Room*. Kromě toho jsou součástí aplikace i další dvě důležité třídy mající na starost zobrazení scény a uživatelské vstupy. Třída *Viewport2D* je určena pro vykreslování dat ve dvojrozměrném módu. Ošetřuje vstupy z klávesnice i myši. Vykresluje mřížku pro informace o právě vykonávané činnosti a hlavně půdorysná data instancí tříd *Room* a *MaterialCalculation*. V případě výběru nějaké akce či výskytu chyby zobrazí podrobnější informace. Naproti tomu jsou třídou *Viewport3D* zobrazována trojrozměrná data.

5.3 Implementované knihovny

5.3.1 OpenGL

OpenGL (Open Graphics Library) je víceplatformní rozhraní pro tvorbu především trojrozměrné grafiky, jež je renderována v reálném čase. I když je *OpenGL* nejčastěji interpretováno jazyky *C*, nebo *C++*, existují implementace knihovny pro většinu běžných programovacích jazyků. Stejně tak je implementováno na většině dnešních platforem. Pokud ne hardwarově, tak alespoň pomocí některé ze softwarových implementací. Vzniklo roku 1992 pod zášti-

tou konsorcia *OpenGL ARB*[1]. Od té doby vzniklo již sedm revizí, z nichž poslední verze je *OpenGL 2.1*. Pomocí asi 250 příkazů pro nastavení a definici objektů a scény lze vytvářet široké spektrum grafických aplikací. Veškeré objekty jsou vykreslovány pomocí základních primitiv, jako jsou body, úsečky, trojúhelníky či polygony. *OpenGL* se chová jako stavový automat. Poskytuje proto celou řadu příkazů pro nastavení stavových proměnných, které definují vlastnosti, jež jsou na objekty aplikovány během zpracování v *OpenGL pipeline*[2].

5.3.2 GLU

Knihovna utilit *GLU* (OpenGL Utility Library) je jednou z nástavbových knihoven *OpenGL*. Je standardně obsažena v každé implementaci *OpenGL*. Obsahuje sadu příkazů pro zjednodušení programování některých procedur oproti klasickému *OpenGL*. Nabízí například nástroje pro teselaci polygonů, generování MIP map, tvorbu NURBS křivek a ploch nebo jednodušší nastavení pohledové matice. Přidává taky některá nová primitiva, tzv. kvadriky, pro tvorbu koulí, válců a disků.

5.3.3 GLUT

GLUT (OpenGL Utility Toolkit) je další sadou nástrojů pro rozšíření *OpenGL API*. Tyto jsou však zaměřeny zejména na jednoduché vstupně/výstupní operace s operačním systémem, které by byly bez použití této knihovny komplikované. Původně byl vytvořen pro *X Window* systémy. Později byl přenesen i na platformy *Windows* a *Mac*. Umožňuje jednoduchou tvorbu a ovládání oken, zpětná volání při vstupech z klávesnice, myši i jiných zařízení a také tvorbu devíti trojrozměrných primitiv. Přináší i jednoduchou možnost tvorby *GUI*[3], které je však omezeno jen na vyskakovací menu.

5.3.4 GLUI

Rozšiřující, i když ne zcela plnohodnotné řešení tvorby grafického uživatelského rozhraní, nabízí knihovna GLUI (OpenGL User Interface Library). Jde o nadstavbu knihovny *GLUT*. Tato víceplatformní knihovna nabízí několik základních prvků pro tvorbu jednoduchého *GUI*. Do prvků, jako jsou rozvinovací či klasické panely, lze umístit obyčejná nebo zaškrtávací tlačítka, tlačítka předvolby, textová pole nebo třeba obyčejné oddělovače. Složitější prvky pro tvorbu pokročilých grafických rozhraní bohužel chybí. Dostatečným plusem je ovšem jednoduchost a z toho i částečně vyplývající rychlost. Škála ovládacích prvků je pro aplikaci *FloorPlanner* postačující.

1. OpenGL Architecture Review Board byla skupina společností s cílem vytvořit jednotné společné API.
2. Posloupnost vykonávaných operací při zpracování grafických dat v OpenGL.
3. Graphical User Interface neboli grafické uživatelské rozhraní.

5.3.5 Texture Loader

Texture Loader v současné verzi 1.4 je knihovna vytvořená *Chrisem Leathley* pro zjednodušené použití textur ve *Win32 OpenGL* aplikacích. Jedná se o velmi jednoduchý, ale přesto silný nástroj pro nahrání, uložení i použití textur v *OpenGL* aplikacích. Podporuje textury v *BMP*, *GIF*, *JPG* i *TGA* formátu, které může nahrát z disku, RAM nebo Internetu. Následně se dají zobrazit za použití bilineárního, nebo trilineárního filtrování. Podporuje generování MIP map a textur s polovičním rozlišením pro systémy se staršími grafickými kartami.

5.4 Vhodná rozšíření aplikace

Aplikace se nyní nachází ve funkční stádiu. Je plně použitelná v praxi. Možností pro rozšíření aplikace je však stále hodně. Vývoj aplikace bude tedy i nadále pokračovat. Následuje výčet několika vlastností, které by se mohly v budoucnu stát součástí aplikace *FloorPlanner*.

- Import nových vzorů pro aktualizaci seznamu podlah včetně jejich textur a velikosti.
- Pokročilé GUI se zobrazením hierarchické struktury objektů a náhledy podlah.
- Rozšíření o algoritmicky jednodušeji implementovatelné druhy podlah jako jsou koberce, linolea, podlahové čtverce nebo různé druhy podkladových materiálů.
- Možnost uložení nebo nahrání projektu, stejně jako import do některého z běžně používaných formátů.
- Import GDL objektů.
- Některé pokročilejší světelné efekty jako jsou odrazy nebo stíny.
- Vizualizace mozaikovitého vyskládávání podlah.
- Kompletní výpis všech materiálů formátovaný pro tisk.

Kapitola 6

Implementace algoritmů v aplikaci FloorPlanner

6.1 Řazení zdí v místnosti

Tvorba místností je v aplikaci *FloorPlanner* realizována postupným kreslením a editací zdí v dvojrozměrném módu. Aby mohlo být v aplikaci jednoduše implementováno mizení zdí, které by bránily pohledu uživatele do místnosti, musí být všechny zdi orientované stejným směrem. Mizení lze pak jednoduše zajistit odstřelem předních či zadních stran polygonů. Pro pozdější zpracování, jako je teselace nebo vyplnění podlahovými dílci, je také nutné, aby na sebe jednotlivé zdi navazovaly a tvořily polygon bez vzájemného křížení hran.

Spojování i otáčení zdí je implementováno ve třídě `Room`. Ta má jako svůj atribut kontejner `walls` obsahující všechny zdi třídy `Wall` a dvojrozměrné pole `vertices`. Toto pole obsahuje setříděné souřadnice shodně orientovaných zdí kontejneru `walls`.

```
class Room
{
private:
   GLfloat** vertices;    /*setridene pole s krajnimi body zdi vectoru
                            walls*/
   vector<Wall> walls;    /*kontejner setridenych zdi*/
   ...
   ...
public:
   void addWall(Wall &wall);        /*pridani nove zdi zadane parametrem*/
   void removeWall(GLuint id);      /*odstraneni zdi*/
   void rotateWall(GLuint id);      /*otaceni zdi*/
   void editWallPosition(GLuint id, /*uprava souradnic krajnich bodu zdi*/
      GLuint index, GLfloat value);

   GLfloat *nearPoint(GLfloat *point); /*pokud najde ve vertices bod
                                         blizky point, vrati jej*/
   ...
   ...
};
```

Samotná třída `Wall` obsahuje jako své atributy dvě souřadnice dvou krajních bodů zdi. Dalším atributem důležitým pro třídění je pole `vertices2d`, obsahující souřadnice dvou bodů. Ty jsou použity ve dvojrozměrném módu, kdy je zeď vykreslena jako čtyřúhelník. Souřadnice jsou při každé změně u vrcholu přepočítány.

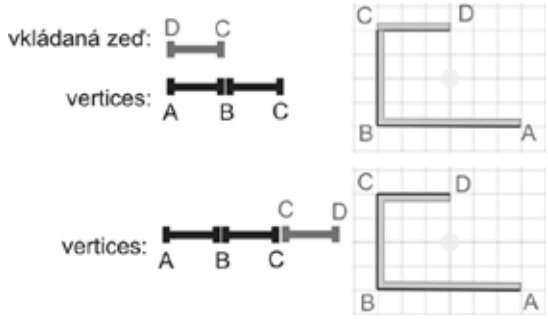
```
class Wall
{
private:
   GLfloat position[4];    /*krajni body zdi*/
   GLfloat vertices2d[4];  /*dva body pro 2D zobrazeni zdi jako
                             ctyruhelniku*/
   ...
   ...
public:
   void switchPoints();    /*otoci zed o 180° kolem sveho stredu*/
   ...
   ...
};
```

6.1.1 Vkládání nových zdí

Vkládání a následné řazení je prováděno v několika krocích. Nejprve třída `Viewport2D`, starající se o projekci a uživatelské vstupy dvojrozměrného módu, zpracuje vstup uživatele, jenž kreslí novou zeď. Pomocí metody `nearPoint()` je zjištěno, zda v blízkosti uživatelem zadaných bodů již neleží jiný bod. V případě nálezu je uživatelem zadaný bod posunut do nalezeného bodu a zdi jsou napojeny.

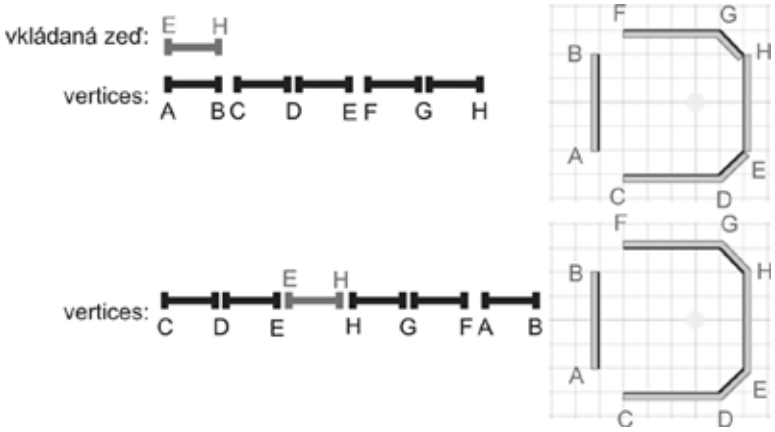
Při každé takto vložené zdi je pole `vertices` i odpovídající zdi v kontejneru `walls` setříděno. To probíhá ve třech krocích pomocí tří metod třídy `Room`. Přidání zdi se realizuje voláním metody `addWall()` s parametrem vkládané zdi. Prohledáním pole `vertices` je následně zjištěno, zda již existuje zeď, která by se dala s nově vkládanou zdí propojit. Pokud není propojení nalezeno, je nová zeď vložena na konec pole `vertices` i kontejneru `walls`. V opačném případě je volána metoda `findSecondConnection()`. Zatímco metoda `addWall()` zjistí pouze jedno propojení, metoda `findSecondConnection()` zkoumá, zda nelze propojit i druhý konec nově vkládané zdi. Pokud tento konec není nijak propojen, dojde nejprve v případě rozdílné orientace původní a nové zdi k otočení nové zdi metodou `rotateWall()`. Následně je pak nová zeď vložena jak do pole `vertices`, tak i do kontejneru `walls` před, resp. za zeď nalezenou v metodě `addWall()`. Je tak vytvořen *díl propojených zdí*. Příklad je zobrazen na obrázku 6.1.

Metoda `findSecondConnection()` také ošetřuje uzavření polygonu místnosti poslední zdí. Testování uzavřenosti kontroluje, zda je celé pole tvořeno jedním *dílem propojených zdí* a zda se poslední bod rovná prvnímu. Tímto testováním je zamezeno provádění teselace a kalkulace v místnostech s otvorem nebo sloupem uvnitř. Poslední metodou třídy `Room` pro řazení hran je `sortWalls()`. Ta je volána v metodě `findSecondConnection()` při nalezení shody nepropojeného bodu vkládané zdi s některým bodem z pole `vertices`. Metoda `sortWalls()` zabezpečí případné otočení zdí a setřídění. Otáčení zdí je zde o něco složitější. Pokud mají zdi, jež mají být s vkládanou zdí propojeny, opačnou orientaci, je třeba jednu z nich otočit. Výslednou orientaci v tomto případě určuje nově vkládaná zeď. Otáčí se však nejen zeď přímo spojená s vkládanou zdí, ale celý *díl propojených zdí*. To znamená, že všechny zdi v tomto dílu musí být otočeny a přeskládány v opačném pořadí. Pole



Obrázek 6.1: Aplikace seřazení pole `vertices` v metodě `findSecondConnection()`.

`vertices` je pak setříděno následovně: Nejdříve jsou vloženy všechny zdi nově vzniklého *dílu propojených zdí* ve správném pořadí i se správnou orientací. Pak následují všechny ostatní zdi ve stejném pořadí, jako byly uloženy v poli `vertices` před setříděním. Příklad třídění v metodě `sortWalls()` je znázorněn na obrázku 6.2.



Obrázek 6.2: Aplikace seřazení pole `vertices` v metodě `sortWalls()`.

6.1.2 Ostatní operace se zdmi

Kromě vkládání zdí jsou v aplikaci FloorPlanner implementovány i jiné operaci se zdmi. Seřadění navazujích zdí musí být zachováno. V následujících řádcích jsou popsány a vysvětleny záludnosti, na které musel být při implementaci těchto jednotlivých operací brán zřetel.

- **Odstraňování zdi** představuje operaci zmenšení kontejneru walls a pole vertices o právě označenou zeď. Důležité je ošetřit případ, kdy je zeď uvnitř uzavřené smyčky. V tomto případě je nutné celou tuto smyčku přeuspořádat. Při každém odstranění zdi je nutné provést kontrolu uzavřenosti polygonu.

- Při **převrácení zdi** je nutné otočit i všechny navazující zdi. Ty jsou následně uspořádány v opačném pořadí.
- **Editace krajních bodů zdi** upráví vždy jednu ze souřadnic zdi. Spolu s ní je třeba upravit i souřadnici zdi, která v tomto bodě navazuje.

6.2 Výběr objektu

OpenGL nabízí kromě módu pro vykreslení objektů na obrazovku i jiné módy. Díky nim lze provádět výběr objektů nebo používat zpětnou vazbu. V aplikaci *FloorPlanner* se objekty nejen vykreslují, ale je s nimi i manipulováno. Pro tento účel nabízí *OpenGL* sadu funkcí a procedur. Ty jsou však navrženy především pro použití v trojrozměrné scéně. Vzhledem k tomu, že výběr probíhá v aplikaci *FloorPlanner* ve dvojrozměrné ortografické projekci, bylo by použití tohoto aparátu zbytečně složité. Namísto něj lze použít speciální vykreslení do nepoužívaného *color-bufferu*. Objekty jsou jednotlivě vykresleny unikátní barvou. Následně jsou z *color-bufferu* čtena pixelová data, jež jednoznačně identifikují vybíraný objekt. Tento postup lze využít pouze pro výběr objektů, které jsou nejblíže pozorovateli. Zároveň je tento algoritmus nevhodný pro scény s příliš velkým počtem objektů. Pro použití v aplikaci *FloorPlanner* je ovšem ideální.

6.2.1 Datové struktury

V této části jsou popsány datové struktury potřebné pro výběr objektů. Celý proces výběru je realizován instancí třídy `Viewport2D`.

```
Viewport2D vp2d;  /*instance pro dvojrozměrný mód*/
Room room;        /*instance representující místnost*/
```

Každému objektu scény přidělí `vp2d` pomocí třídy `Identifier` unikátní identifikátor.

```
class Identifier
{
private:
   GLuint current;
public:
   Identifier(void):current(1){};
   GLuint generate() { return current++; };
};
```

Každý objekt si uchovává svůj identifikátor. Zároveň jsou všechny identifikátory uloženy v asociativním kontejneru spolu s údajem o typu objektu.

```
map<unsigned int, unsigned short> identifiers_of_objects;
```

6.2.2 Implementace výběru objektu

Kliknutím myši do prostoru rýsovacího plátna bez vybrané akce je v třídě `Viewport2D` aktivován mód výběru. Souřadnice kliknutí jsou uloženy. Následně je celá scéna vykreslena do zadního bufferu.

Nejprve je nutné vyčistit *color-buffer* černou barvou a vypnout výpočet osvětlení.

```
glClearColor(0.0, 0.0, 0.0, 0.0);  /*nastaveni barvy color-bufferu na
                                     cernou*/
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glDisable(GL_LIGHTING);            /*vypnuti osvetleni*/
```

Projekční matice je nastavena shodně jako v případě klasického vykreslení scény ve dvojrozměrném režimu.

```
glMatrixMode(GL_PROJECTION); /*mod projekcni matice*/
vp2d.projection();           /*nastaveni projekce*/
glMatrixMode(GL_MODELVIEW);  /*mod modelovaci matice*/
glLoadIdentity();            /*nahrani jednotkove matice*/
```

Následuje vykreslení celé scény v módu výběru. Identifikátor každého objektu je symetricky převeden na barvu tak, aby každá barva jednoznačně určovala jeden objekt scény. Jednotlivé objekty jsou pak touto barvou vykresleny.

```
room.displaySelection();     /*zobrazeni objektu specialne pro vyber*/
```

Před čtením pixelových dat je zapotřebí dokončit všechny operace prováděné *OpenGL*. To lze vynutit funkcí `glFlush()`.

```
glFlush();                   /*vynucene dokonceni vsech OpenGL operaci*/
```

Metoda `selectObject()` se postará o výběr objektu v místnosti zadané parametrem. Vzhledem k většímu množství operací je tato metoda rozepsána níže.

```
vp2d.selectObject(&room);    /*cteni pixelovych dat a vyber objektu*/
```

Díky absenci volání funkce `glutSwapBuffers()` se celá procedura nezobrazí na obrazovce.

V metodě `selectObject()` jsou použity dříve uložené souřadnice kliknutí. V kódu jsou reprezentovány polem `mouse_click[]`. Pro uložení vlastností pixelu, na který bylo kliknuto, poslouží ukazatel `*pixel`.

```
GLubyte *pixel = new GLubyte[3];
```

Funkce `glReadBuffer()` nastaví zadní buffer pomocí parametru `GL_BACK` pro čtení.

```
glReadBuffer(GL_BACK);
```

Pro čtení pixelových dat z frame bufferu slouží funkce `glReadPixels()`. V tomto případě je čten pouze jeden pixel, a to ten, na nějž bylo kliknuto. Jeho barva je uložena do pole daného ukazatelem `*pixel`. Použitím konstant `GL_RGB` a `GL_UNSIGNED_BYTE` je červená, zelená i modrá složka barvy uložena jako osmibitové číslo bez znaménka.

```
glReadPixels(
    mouse_click[0],window[1]-mouse_click[1], /*souradnice ctene oblasti*/
    1,1,                        /*velikost ctene oblasti*/
    GL_RGB, GL_UNSIGNED_BYTE, /*parametry ulozeni oblasti*/
    pixel);               /*ukazatel na pole pro ulozeni pixelovych dat*/
```

Barvu lze nyní jednoduše pomocí bitových posunů a bitového součtu zpětně převést na identifikátor.

```
GLuint r,g,id;
r = pixel[0] << 16;
g = pixel[1] << 8;
id = pixel[2];
id = r | g | id;
```

Nakonec je z asociativního pole `identifiers_of_objects` získána informace o typu objektu s tímto identifikátorem. Objekt je vyhledán v instanci třídy `Room`, označen a připraven pro editaci a jiné operace.

6.3 Antialiasing

Vykreslování scén bez použití nějaké formy vyhlazení vytváří nepříjemné artefakty a příliš ostré a zubaté přechody (*jaggies*). *OpenGL* nabízí relativně kvalitní nástroje pro vyhlazování bodů, úseček, polygonů i celé scény.

6.3.1 Vyhlazování úseček

Dvojrozměrný režim aplikace *FloorPlanner* vykresluje pouze úsečky a polygony bez použití textur či osvětlení. Okraje polygonů jsou zde vždy obtaženy úsečkami, tudíž „zubatost“ polygonů není potřeba řešit. Úsečky však často trpí dosti viditelným *aliasem*, a proto je na ně použit jednoduchý antialiasing úseček v *OpenGL*.

Nejprve je nutné pomocí funkce `glEnable()` povolit vyhlazování úseček a míchání barev.

```
glEnable(GL_LINE_SMOOTH);
glEnable(GL_BLEND);
```

Poté jsou nastaveny faktory míchání barev.

```
glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
```

Posledním nastavením je funkce `glHint()`. Ta nastaví *OpenGL* tak, že pokud to bude implementací povoleno, použije pro vyhlazování nejkvalitnější metodu.

6.3.2 Vyhlazování scény

OpenGL využívá pro vyhlazování celé scény metodu *multisamplingu*[3]. Celá scéna je několikrát vykreslena s minimálním posunem do akumulačního bufferu, kde je smíchána. Pro pohyb scény (tzv.*jittering*) jsou použity fukce `accFrustum()` a `accPerspective()`.

Následující pseudokód popisuje akumulaci do akumulačního bufferu:

```
vyčisti akumulační buffer;
for( i = 0; i < n; i++)
{
vyčisti color-buffer a buffer hloubky;
nastav perspektivu za použití jitter[i];
vykresli scénu;
přidej color-buffer do akumulačního bufferu;
}
zkopíruj akumulační buffer do color-bufferu;
```

Před samotnou akumulací je nutné vyčistit akumulační buffer pomocí funkce `glClear()`.

Proměnná `n` udává počet akumulací. Její hodnota razantně ovlivňuje dobu vykreslování, ale i výslednou kvalitu. V programu *FloorPlanner* lze její hodnotu nastavit optimálně dle rychlosti právě používaného hardwaru.

Akumulace probíhá vykreslováním do *color-bufferu*, jehož obsah je následně zkopírován do akumulačního bufferu. Proto je nutné při každém cyklu vyčistit jak *color-buffer*, tak *buffer hloubky*.

Pro nastavení perspektivy se používá funkce `accPerspective()`, která je upravenou verzí funkce `gluPerspective()`. Oproti původní funkci má dva nové parametry pro posunutí scény. Hodnoty posunutí jsou uloženy v poli `jitter[n]`. Optimální hodnoty pro co nejlepší výsledky jsou použity z [3].

Následně jsou běžným způsobem vykresleny všechny objekty scény do *collor-bufferu*. V aplikaci *FloorPlanner* to znamená, že na instance třídy `Room` a `MaterialCalculation` je volána metoda `displayGL()`.

Aniž by byl *color-buffer* zobrazen, je jeho obsah pomocí funkce `glAccum()` s parametrem `GL_ACCUM` zkopírován do akumulačního bufferu.

Po skončení akumulačního cyklu jsou příkazem `glAccum()` s parametrem `GL_RETURN` akumulovaná data jednoduše zkopírována zpět do *color-bufferu* a zobrazena.

6.4 Teselace

Samotné *OpenGL* dokáže zobrazit pouze základní primitiva, jako jsou body, úsečky, trojúhelníky a konvexní polygony. Kvůli optimalizaci světelného modelu tak nezvládá korektně zobrazit konkávní polygony, tedy polygony mající nejméně jeden vnitřní úhel větší než π. Takovýto případ však může v aplikaci *FloorPlanner* u podlahy i stropu běžně nastat, a proto je nutné provést teselaci a získat rozdělením jednoho konkávního polygonu několik, již konvexních, trojúhelníků. Nadstavbová knihovna *GLU* pro tento účel nabízí sadu procedur a funkcí.

Grafická data podlahy i stropu jsou uložena v zobrazovacích seznamech uvnitř instance třídy `Floor`. Její metoda `tessellate()` získá vrcholy polygonu z pole `vertices` instance třídy `Room`. Pro teselaci je nutné provést několik kroků. Prvním z nich je definovat sadu funkcí zpětného volání. Jednotlivě jsou pak volány při vzniku určitých událostí během vykonávání algoritmu. Kromě jednoduchých volání pro začátek a konec vykreslování, je zde také volání pro definici chování vrcholů s nastavením normál. Posledním a nejdůležitějším voláním je ošetření chyb při teselaci. Toto volání speciálně reaguje na chybu křížení dvou stěn. Tímto je ošetřen tento chybný uživatelský vstup.

Teselace je podmíněna vytvořením teselačního objektu. Na něj jsou navázána výše zmiňovaná zpětná volání. *GLU* umožňuje vykreslování polygonu s několika samostatnými částmi. Tato vlastnost je však vyloučena algoritmem řazení zdí. Algoritmu jsou tedy předány body získané z pole `vertices` a ten je zpracuje jako jeden uzavřený celek. Celý proces je ukládán do zobrazovacích seznamu, který je následně vykreslen poprvé jako podlaha a podruhé posunutě a převráceně podle roviny *xy* jako strop místnosti.

6.5 Vyplnění polygonu

Stěžejním úkolem práce bylo implementovat korektní algoritmus pro kalkulaci a vizualizaci plovoucích podlah v interiéru. Pro tento účel bylo jako nejideálnější vyhodnoceno *řádkové vyplňování se seznamem aktivních hran*. Aspekty vedoucí k výběru a podrobnější popis tohoto algoritmu je obsahem kapitoly 4.3.

Algoritmus byl ovšem podstatně upraven. Níže je vypsán výčet vlastností, které jsou pro použití v aplikaci *FloorPlanner* nedostatečné nebo nevhodné a musely být upraveny.

- **Pravidelný rastr** — algoritmus je vytvořen pro vyplňování pravidelného rastru. Řádek je tedy vždy určen celým číslem. V aplikaci *FloorPlanner* se však vyplňuje podlahovými dílci o neceločíselné šířce, resp. výšce. Posun na nový řádek tedy neprobíhá snížením souřadnice *y* o jedničku, ale o šířku dílce.
- **Nepřesné vyplnění polygonu** — při vyplnění polygonu původním algoritmem nemusí být vyplněna celá plocha polygonu, protože hraniční body jsou pouze aproximací původních geometricky zadaných hran. To však je nežádoucí, neboť podlahové dílce musí zakrývat celou plochu místnosti.
- **Kompromisy při zpracování vrcholů** — jakmile se algoritmus dostane ke konci hrany, pro zachování sudého počtu hran ji vyřadí z dalšího zpracování a počítá jen s hranou na ní navazující. Někdy jsou hrany v předzpracování dokonce zkracovány o jedničku, aby na sebe nenavazovaly. Aby však byl vyplněn celý prostor místnosti, je zapotřebí chování na vrcholech ošetřit pokročilejším mechanismem.
- **Tabulka hran** — v této tabulce odpovídá jeden řádek jednomu řádku v rastru. Hrany jsou uloženy do řádků, ve kterých začínají. Při neceločíselném zpracování by tabulka byla zbytečná, a tak je místo ní použit obyčejný seznam se sestupným řazením podle souřadnice *y* počátečního bodu.

Pozměněný algoritmus se v aplikaci *FloorPlanner* sestává ze tří oddělených částí. Po definici potřebných datových struktur jsou tyto části obsahem následujících tří kapitol. V první části je naplněn *seznam hran*. Další částí je cyklus zpracovávající jednotlivé řádky vyplňování. Zde je třeba ošetřit chování na vrcholech a ve správný okamžik přidávat či odebírat hrany ze *seznamu aktivních hran*. Třetí částí je zpracování úseků na aktuálním řádku, jež se vyplňují jednotlivými podlahovými dílci.

6.5.1 Datové struktury

Následuje popis datových struktur použitých při vyplňování. Základními strukturami jsou hrana a podlahový dílec.

```
struct Edge /* hrana */
{
    GLfloat y_begin;      /* souradnice y pocatecniho bodu */
    GLfloat x_end, y_end; /* souradnice koncoveho bodu */
    GLfloat x_current;    /* souradnice x pruseciku s atualnim radkem */
    GLfloat x_increment;  /* zmena x_current pri prechodu na novy radek*/
};
struct Tile /* podlahový dílec */
{
    GLfloat x,y;  /* rozmery podlahoveho dilce */
};
```

Hrany jsou získány z pole `vertices` v instanci třídy `Room`, a to v pouze v případě, že hrany tvoří uzavřený polygon. Velikost podlahových dílců je závislá na aktuálně vybraném typu podlahy. Pro uložení aktuální pozice algoritmu v polygonu slouží proměnné `x_alg` a `y_alg`.

```
GLfloat x_alg; /*souradnice x v aktualnim radku*/ a
GLfloat y_alg; /*souradnice y aktualního radku*/.
```

Dalšími strukturami jsou kontejnery pro uchovávání hran, ořezaných podlahových dílců a údajů o vzájemném posunutí dvou řad vůči sobě. Podrobněji jsou vysvětleny níže v popisu algoritmu.

6.5.2 Inicializace

Pole `vertices` je předáno na vstup algoritmu. Pokud je vyplňování provedeno pod úhlem α , jsou v tu chvíli všechny body pole `vertices` otočeny o $-\alpha$. Hrany jsou poté uloženy s orientací shora dolů do *seznamu hran*.

```
list<Edge> edgelist; /*seznam hran*/
```

Vodorovné hrany jsou zanedbatelné. Nejsou tedy do seznamu hran uloženy. U každé hrany je vypočítán její přírůstek *x* při přechodu na nový řádek. Seznam je následně setříděn

sestupně podle atributu `y_begin`. Proměnné `y_alg` je přiřazena hodnota `y_begin` první hrany *seznamu hran* `edgelist`.

Kalkulaci spotřebovaného materiálu zajišťuje celočíselné počítadlo `tile_counter`. Před spuštěním algoritmu je počítadlo vynulováno.

Celý následující proces se z důvodu optimalizace ukládá do zobrazovacího seznamu. Pokud bylo prvně pole `vertices` otočeno, musí být nyní modelová matice vynásobena rotační maticí s úhlem α.

6.5.3 Zpracování hran na jednotlivých řádcích

Již z názvu algoritmu je patrné, že postupuje po řádcích, a to od maximální souřadnice y vyplňovaného polygonu po minimální. V jednotlivých řádcích se vyplňují úseky mezi hranami v *seznamu aktivních hran*.

```
list<Edge> activeedgelist; /*seznam aktivnich hran*/
```

Úkolem této části algoritmu je udržovat právě *seznam aktivních hran* aktuální. Při zpracování jednotlivých řádků je třeba nalézt vrcholy polygonu, které se v tomto řádku nacházejí. Hrany, které v těchto vrcholech v daném řádku začínají či končí, jsou uloženy do seznamu `vertexedgelist`.

```
list<Edge> vertexedgelist; /*seznam koncicich a zacinajicich hran*/
```

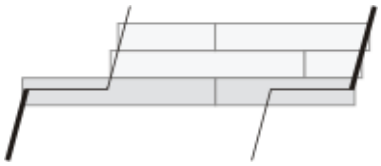
Začínající hrany jsou zde přesunuty ze *seznamu hran* a končící ze *seznamu aktivních hran*. Po seřazení seznamu vzestupně podle atributu `x_current` odpovídají vždy dvě po sobě jdoucí hrany jednomu vrcholu, popřípadě dvěma vrcholům spojených vodorovnou hranou. Nyní je potřeba vyhodnotit situaci na vrcholu a podle toho umístit, nebo naopak neumístit hrany do dvou následujících seznamů.

```
list<Edge> preedgelist; /*seznam aktivnich hran pro aktualni radek*/
list<Edge> postedgelist;  /*seznam aktivnich hran pro pristi radek*/
```

Při vyhodnocování je brán ohled na to, zda se v budoucím seznamu `activeedgelist` bude jednat o lichou, nebo sudou hranu. Tedy zda vyplňování aktuálního úseku bude probíhat od této hrany, nebo po tuto hranu. Příklad je na obrázku 6.3, kde je vidět, že i při stejném tvaru hran se vlevo bude v aktuálním řádku zpracovávat začínající hrana, zatímco vpravo je zapotřebí ještě použít končící hranu. Dalšími ovlivňujícími faktory jsou znaménko atributu `x_inkrement`, velikost `x_inkrement` nebo zda obě hrany ve vrcholu končí či začínají. Před vykreslením aktuálního řádku jsou *seznamu aktivních hran* přiřazeny hrany seznamu `preedgelist` a po vykreslení pak hrany seznamu `postedgelist`.

K dodržení minimálního posunu mezi dílci ve dvou sousedících řadách slouží kontejner `lastshift`.

```
vector<GLfloat> lastshift; /*vektor posunu dilcu predchoziho radku*/
```



Obrázek 6.3: Příklad zpracování končících a začínajících hran.

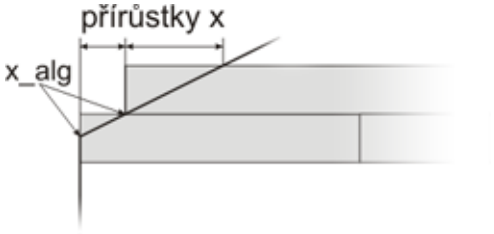
Nastavení hodnot jeho vnitřních prvků se vztahuje až k následující kapitole, avšak velikost tohoto kontejneru se musí řešit právě zde. Posun dílců vůči sobě se počítá pro každý úsek řádku odděleně, tudíž každý prvek odpovídá jednomu úseku, tedy dvěma hranám v *seznamu aktivních hran*. Při zvětšení či zmenšení *seznamu aktivních hran* je třeba přizpůsobit i velikost kontejneru `lastshift`.

Pro přechod na nový řádek je od aktuálního `y_alg` odečtena velikost *y* podlahového dílce. Hrany *seznamu aktivních hran* jsou nahrazeny hranami seznamu `postedgelist`, přičemž je pro každou hranu provedena operace přičtení `x_increment` k `x_current`. V případě začínajících hran ovšem musí dojít k přičtení pouze částečného přírůstku.

6.5.4 Vykreslení podlahových dílců do úseků na řádku

Vykreslování jednoho řádku je rozděleno na úseky. Vykreslované úseky se nachází vždy mezi lichou a sudou hranou *seznamu aktivních hran*. Ten však musí být na začátku každého řádku seřazen vzestupně podle `x_current`. Následující postup popisuje vykreslení dílců v rámci jednoho úseku mezi dvěmi hranami. U ostatních úseků na řádku je postup analogický.

Aktuální souřadnice `x_alg` je nastavena na hodnotu `x_current` první, tedy liché, hrany. Pokud má tato hrana záporný přírůstek `x_increment`, je k souřadnici `x_alg` přičten. Pokud se navíc jedná o končící hranu, je přičtený přírůstek pouze částečný. Analogicky je nastavena hodnota proměnné `x_stop`, obsahující souřadnici *x*, kde daný úsek končí.



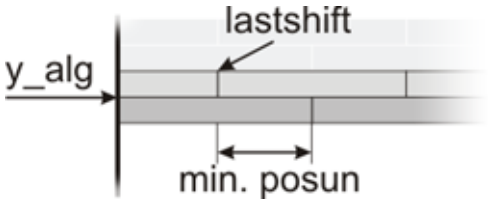
Obrázek 6.4: Nastavení proměnných na začátku úseku.

Úseky jsou vyplňovány podlahovými dílci obdélníkového tvaru. I na zkosených hranách mají podlahové dílce pro lepší přehlednost obdélníkový tvar. Na konci a někdy i na začátku každého úseku dochází ke zkrácení dílce. Zbytek však nemůže být považován za odpad, ale musí být uložen pro použití v některé z následujících řad. Tyto zkrácené dílce jsou ucho-

vány ve dvou, dle velikosti *x* seřazených, seznamech dílců.

```
list<Tile> leftcut_tiles;  /*seznam zleva zkracených dilcu*/
list<Tile> rightcut_tiles; /*seznam zprava zkracených dilcu*/
```

Nejdůležitější podmínkou při pokládce plovoucích podlah je dodržovat minimální posun mezi dvěmi sousedícími řadami. Proto je nutné správně zvolit první dílec každého úseku tak, aby byl posun vůči předchozí řadě dostatečný. K tomu je použit výše zmiňovaný kontejner `lastshift`, do kterého je vždy zaznamenána souřadnice *x* spojnice dvou dílců. Tato hodnota je pak použita na dalším řádku pro vložení správně dlouhého prvního dílce.



Obrázek 6.5: Minimální posun dvou řad vůči sobě.

První možností je použít jeden ze zkrácených dílců z předchozích řad. Ze seznamu zkrácených dílců `leftcut_tiles` je vybrán nejdelší dílec splňující podmínku dostatečného posunu. Výběr nejdelšího dílce je důležitý pro snížení počtu spotřebovaných dílců. Druhou možností je použít na začátek celý dílec. To však kvůli dodržení minimálního posunu také nemusí být možné. Pokud předchozí dva způsoby nevedou k úspěchu, je nutné zkrátit nový celý dílec. Zde je velmi důležité spolehlivě vyhodnotit situaci a zkrátit dílec co nejméně. Zbytek dílce je uložen do seznamu zprava zkrácených dílců `rightcut_tiles`.

Hodnota proměnné `x_alg` je nastavena za první dílec. Následuje vyplnění úseku celými dílci. Po každém dílci je hodnota `x_alg` zvětšena o délku podlahového dílce. Na konci každého úseku je většinou opět nutné použít zkráceného dílce. Na potřebnou délku je zkrácen buď dostatečně dlouhý dílec ze seznamu `rightcut_tiles`, nebo celý dílec. Zatímco zbytek z již ořezaného dílu již zde tvoří nepoužitelný odpad, zbytek z ořezaného celého dílce je uložen do seznamu `leftcut_tiles` pro případné použití na začátek některé z dalších řad.

Svou roli zde hraje i atribut `tile_counter`. S každým použitým novým dílcem se jeho hodnota inkrementuje. Ve finále tento atribut udává počet spotřebovaných podlahových dílců.

6.6 Mapování náhodných textur

Plovoucí podlahy vynikají pestrostí vzorů a kreseb, které dřevo vytváří. Použití jedné opakující se textury by výsledný zážitek z podlahy dosti degradovalo. Pro větší variabilitu a reálnější vzhled je v aplikaci *FloorPlanner* pro každý typ podlahy použito celkem čtyř různých textur. Textury jsou fotokopiemi původních plovoucích podlah v dostatečném rozlišení.

Neboť jsou lamely plovoucích podlah vždy nejméně čtyřnásobně dlouhé, než jsou široké a *OpenGL* ve starších verzích dovoluje použití pouze textur o velikosti mocniny dvou, je možné uložit tyto čtyři textury jednotlivých lamel do textury jedné. Během kalkulace materiálu je pak každému vykreslovanému dílu předána náhodně jedna ze čtyř sad texturových souřadnic. Tím je dosaženo vizuálně uspokojivé variabilnosti podlahy.

Obrázek 6.6: Příklad a aplikace textury se čtyřmi variantami lamel plovoucí podlahy.

Kapitola 7

Závěr

Aplikace pro návrh a kalkulaci plovoucích podlah jsou velkým přínosem při realizaci interiérů. Měly by poskytnout zákazníkům firem, zabývajících se prodejem a pokládkou podlah, nástroj pro zhmotnění a ucelení jejich představ. Zákazníci tak získávají nejen dobrý pocit z dobře vynaložených financí, ale díky přesnější kalkulaci také skutečně ušetří. Firmám se naopak sníží množství navráceného materiálu i konfliktů se zákazníky. Bohužel v této sféře zatím není dostupný žádný softwarový produkt. Proto v rámci této práce vznikla jednoduchá CAD aplikace *FloorPlanner* pro rychlý návrh plovoucích podlah.

V této bakalářské práci byla nejprve rozvinuta problematika pokládky plovoucích podlah. Byly popsány různé typy pokládky i faktory ovlivňující spotřebu materiálu. Tím byly stanoveny požadavky na aplikaci. Dále byl rozveden vývoj a současný stav CAD systémů. Zejména pak byly zdůrazněny výhody objektově orientovaných AEC/CAD aplikací, jež vedly k výběru tohoto přístupu pro aplikaci *FloorPlanner*.

Byly zde zhodnoceny i jiné aplikace pro vizualizaci plovoucích podlah. K dispozici jsou však jen jednoduché webové aplikace nebo příliš komplikovaná a drahá řešení, která navíc nejsou pro pokládku plovoucích podlah přímo určena. Proto jsou nedostačující.

Pro vytvoření vhodné aplikace bylo potřeba specifikovat požadavky a následně vybrat vhodné metody a algoritmy, které jsou nutné pro vyhovění všech požadavků správné pokládky podlah i následné vizualizace. Implementací těchto metod spolu s jednoduchým uživatelským prostředím vznikla aplikace *FloorPlanner*. Byla navrhnuta tak, aby vyhověla nárokům pro co nejpřesnější kalkulaci materiálu. Je toho docíleno simulací postupů při skutečné pokládce. V atypických prostorách lze takto zpřesnit odhad oproti klasickým postupům až o několik procent. Vzhledem k vysokým cenám těchto materiálů může vést k finanční úlevě jak pro firmy, tak i pro jejich zákazníky. Navíc je možné podlahy relativně reálně zobrazit v jejich cílovém prostoru a tak u zákazníka snížit strach z neznáma a zvýšit ochotu investovat.

Vzhledem k velkým úsporám díky této aplikaci bude i nadále vývoj aplikace pokračovat a rozšiřovat své zaměření i na jiné podlahové materiály. Tato práce také ukázala, že na trhu je v této problematice, i přes přínos těchto řešení, stále hluché místo.

Literatura

[1] Fischer, M. a Vaněk, P.: *CAD I-V*, e-Architekt.cz, 2004, Seriál dokumentů dostupný na <http://www.e-architekt.cz/> (leden 2008).

[2] Bozdoc, M.: *iMB The History of CAD*, MB Solutions, 2004, Dokument dostupný na <http://mbinfo.mbdesign.net/CAD-History.htm> (leden 2008) . 2.2.1

[3] Shreiner, D. a Woo, M. a Neider, J. a Davis, T.: *OpenGL Průvodce programátora*, Computer Press, a.s., 2006, 80-251-1275-6. 6.3.2

[4] McReynolds, T. a Blythe, D.: *Advanced Graphics Programming Techniques Using OpenGL*, Elsevier Inc., 2005, 1-55860-659-9, Dokument dostupný na adrese <http://web.informatik.uni-bonn.de/II/ag-klein/global/proseminar_ss2001/course12.pdf> (leden 2008) .

[5] Žára, J. a Beneš, B. a Sochor, J. a Felkel, P.: *Moderní počítačová grafika*, Computer Press, a.s., 2004. 4.3.1

[6] Turek, M.: *CZ NeHe OpenGL*, Michal Turek, 2004, Dokument dostupný na <http://nehe.ceske-hry.cz/download/download/cz_nehe_opengl.pdf> (leden 2008) .

[7] Wright, R. a Lipchak, B.: *OpenGL Super Bible*, Macmillan Computer Publishing, 1996, 1-57169-073-5.

[8] Snapp, R.: *Scanline Fill Algorithm*, Department of Computer Science, University of Vermont, 2003, Dokument dostupný na <http://www.cs.uvm.edu/~snapp/teaching/CS274/lectures/scanlinefill.pdf> (leden 2008) .

Příloha A

Návod k použití aplikace FloorPlanner

A.1 Úvod

FloorPlanner je aplikace pro vizualizaci a kalkulaci plovoucích podlah v interiéru. Je určena pro platformu *Win32*. Nabízí základní *AEC/CAD* funkce pro tvorbu místností, v nichž je plovoucí podlaha realizována. V příštích kapitolách tohoto návodu jsou podrobněji rozvedeny všechny funkce programu.

A.2 Uživatelské prostředí

Uživatelské prostředí aplikace FloorPlanner se dělí na dvě části. Vpravo jsou rozmístěny ovládací prvky pro tvorbu, ovládání, editaci i odstraňování objektů zobrazených v části vlevo. Práce na projektu se pak logicky dělí na dva způsoby. Prvním je kresba půdorysu místnosti na kreslicí plochu. Druhým je pak vizualizace a prezentace místnosti v trojrozměrném režimu. Přepínání mezi těmito režimy zajišťuje dvojice tlačítek v horní části pravého panelu.

A.3 Dvojrozměrný režim

Po spuštění aplikace FloorPlanner je zobrazen dvojrozměrný režim. Místnost je zobrazena formou půdorysu na pravidelné mřížce. Červeně zvýrazněné přímky označují osy soustavy. Veškeré operace, spojené s editací místnosti, lze provádět výhradně v tomto režimu.

Pokud není vybrána nebo aktivována žádná z funkcí, nachází se aplikace v režimu výběru. Kliknutím lze označit jednotlivé zdi, okna, nebo dveře. V pravém horním rohu jsou pak zobrazeny informace o objektu, jako jsou rozměry, nebo délka.

Manipulace s kreslicí plochou se provádí pomocí klávesy **Alt**. V kombinaci s levým tlačítkem myši s plochou posouvá a s pravým tlačítkem mění měřítko.

Následuje vysvětlení funkcí jednotlivých prvků rozvinovacích panelů, kterými se lze přepnout do různých režimů. Pro provedení některých akcí je nutné nejdřív označit objekt, na nějž bude funkce aplikována.

A.3.1 panel Wall editing

Veškeré prvky pro přidávání, odebírání či editaci zdí, tedy i tvaru celé místnosti, jsou obsaženy v tomto panelu.

- **Add walls** — stisknutím tohoto tlačítka se aplikace přepne do stavu pro vkládání zdí. Kliknutím a tahem levého tlačítka lze pak na ploše vlevo vytvářet nové zdi. Se stisknutou klávesou **Shift** lze navíc vytvářet pouze svislé či vodorovné zdi. Pro propojení dvou zdí je nutné vytvořit jejich krajní body dostatečně blízko sebe. Zeď je tvořena obdélníkem se zvýrazněnou jednou hranou. Ta označuje vnitřní okraj místnosti.
- **Rotate wall** — v případě vytvoření zdi se špatnou orientací je možné zeď označit a následně tímto tlačítkem otočit o 180°. Pokud je označená zeď spojená s jinými zdmi, je i jejich orientace přizpůsobena.
- **Remove wall** — toto tlačítko slouží k odstranění označené zdi. Spolu se zdí budou odstraněny i veškeré její okna a dveře.
- **Clear all** — jednoduchý příkaz pro odstranění všech objektů na kreslicí ploše.
- Editovatelná pole **1.point** a **2.point** — při označení zdi jsou v těchto polích uvedeny souřadnice prvního i druhého bodu zdi. Jejich editací lze se zdí manipulovat. Pokud zeď sousedí s jinou zdí, jsou souřadnice upraveny i u ní. Posunem dvou bodů blízko sebe však nelze dvě rozdělené zdi propojit.

A.3.2 panel Floor

Panel *Floor* obsahuje prvky pro práci s podlahou. Kalkulace i zobrazení podlahy je podmíněna uzavřením místnosti. Zdi tedy musí tvořit jeden uzavřený polygon bez vzájemného křížení zdí.

- **Display floor** — zásadním prvkem tohoto panelu je toto zaškrtávací tlačítko. Při splnění podmínky uzavřenosti místnosti je provedena kalkulace podlahy a plocha místnosti je vyplněna dílci plovoucí podlahy. V levém horním rohu jsou pak uvedeny informace o potřebném materiálu pro vykreslenou místnost.
- **Refresh** — v případě změny některého z parametrů podlahy je nutné toto tlačítko použít pro přepočítání a překreslení podlahy.
- **Floor angle** — hodnota tohoto pole udává pod jakým úhlem bude podlaha v místnosti vyskládávána. Původní hodnota 0° znamená směr shora dolů.
- **Get wall angle** — směr pokládky plovoucí podlahy se většinou řídí jednou ze zdí místnosti. Tímto tlačítkem lze nastavit úhel pokládky podle právě označené zdi. Hodnota je také zkopírována do pole *Floor angle*.

- **Shift limit** — tato hodnota je velice důležitá pro konečnou spotřebu materiálu. Změnou lze nastavit minimální posun v procentech mezi dvěma sousedními řadami. Snížením na povolené minimum tak lze částečně ušetřit materiál. Hodnota musí být menší než 50%.

- **Trash limit** — udává délku zbytku ze zkrácené lamely, který již bude považován za odpad. Zvětšením lze dosáhnout pravidelnějšího vzoru.

- **Type** — jednotlivé typy a vzory plovoucích podlah jsou obsahem právě tohoto seznamu. Výběrem požadované podlahy je místnost vyplněna podlahovými dílci s odpovídající velikostí i texturou.

A.3.3 panel Windows/Doors editing

Tento panel nabízí nástroje pro vložení dveří a oken. Vytváří tedy velikostně nastavitelné otvory ve zdech. Pro výběr mezi vytvářením oken, nebo dveří slouží dvě tlačítka předvolby.

- **Add** — tlačítko přepíná do módu přidávání oken a dveří. Je nutné, aby všechna textová pole v panelu obsahovala platná data. Okno je přidáno kliknutím na předem označenou zeď.

- **Remove** — tlačítko pro odstranění označeného okna, nebo dveří.

- **Width, Height** — textová pole pro nastavení výšky a šířky vkládaného okna, nebo dveří.

- **From floor to object** — toto pole je aktivní pouze u oken a nastavuje vzdálenost spodního okraje okna od podlahy.

A.3.4 panel Options

V tomto panelu je několik doplňujících nastavení ovlivňujících především zobrazení v trojrozměrném režimu. Ve dvojrozměrném módu je použitelná pouze jedna položka.

- **Set position** — toto tlačítko umožňuje nastavení pozice světla. Světlo je na ploše zobrazeno jako malý žlutý kroužek. Kliknutím na tlačítko a následně na kreslicí plochu je světlo přesunuto na pozici pod kurzorem.

A.4 Trojrozměrný režim

Tento režim aplikace *FloorPlanner* slouží především pro presentační účely. Zobrazuje místnost v trojrozměrném pohledu první osoby. Pohyb prostorem umožňuje kombinace kurzorových šipek a myši. Kurzorové šipky umožňují pohyb všemi směry ve vodorovné rovině. Pohyb myši při stisknutém levém tlačítku otáčí pohled pozorovatele. Pravé tlačítko pak slouží ke stoupání a klesání pozorovatele. Oproti dvojrozměrnému módu je zde jen několik možností voleb. Všechny se opět nacházejí v pravé části aplikace, nyní však pouze v jediném rozvinovacím poli.

A.4.1 panel Options

Panel *Options* skýtá několik nastavení pro změnu zobrazení trojrozměrného režimu. Některé volby mohou být velice náročné na výkon a razantně tak prodloužit výpočetní čas scény.

- **Light type** — ze seznamu lze vybrat jeden z druhů osvětlení. Na výběr je klasické bílé sluneční světlo, nažloutlé světlo žárovky a namodrale svítící zářivka.
- **MIP mapping** — při zaškrtnutí bude aplikace *FloorPlanner* generovat *MIP map* textury. Částečně tak lze zrychlit běh aplikace na úkor náročnosti na paměť. Tuto volbu lze zapnout pouze se zapnutým filtrováním (viz *Filtering*).
- **High quality textures** — odškrtnutím této položky bude velikost textur zmenšena na čtvrtinu. Lze tedy při minimální degradaci výsledku o něco zrychlit výpočet a zmenšit nároky na paměť.
- **Antialiasing** — zapnutím antialiasingu lze výrazně potlačit projevy moiré a „zubatých“ hran a docílit tak příjemného vyhlazení scény. Tento výpočet je velice časově náročný a tak je třeba volit hodnoty optimální pro právě používáný hardware.
- **Filtering** — zapnutím některého typu filtrování lze při relativně nízkém snížení výkonu odstranit některé nepříjemné efekty textur. Textury však ztrácejí detaily a mohou působit poněkud rozmazaně.

Příloha B

Přiložené CD

K této bakalářské práci je přiložen kompaktní disk s následujícím obsahem:

- Text této bakalářské práce ve formátech XML, PDF a PS.
- Veškeré použité obrázky ve formátu PNG.
- Zdrojový kód aplikace FloorPlanner včetně souborů projektu pro Visual Studio 2005 a použitých textur.
- Zkompilovaná aplikace FloorPlanner včetně použitých textur.