VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

APLIKACE PRO SIMULACI AKUSTIKY MÍSTNOSTI

ROOM ACOUSTICS SIMULATION APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MARTIN KRBILA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. LADISLAV MOŠNER**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Krbila Martin**
Program: Informační technologie
Název: **Aplikace pro simulaci akustiky místnosti**
Room Acoustics Simulation Application
Kategorie: Zpracování signálů

Zadání:

1. Seznamte se s metrikami pro popis akustiky místnosti, pojmem impulsní odezva místnosti
2. Seznamte se s technikami měření impulsní odezvy místnosti, přístupy k simulaci akustiky (image method, ray-tracing)
3. Implementujte zvolenou metodu pro simulaci akustiky
4. Vytvořte aplikaci umožňující práci s modelem místnosti
5. Vytvořte plakát a/nebo video dokumentující práci

Literatura:

- KUTTRUFF, H.: *Room acoustics*. 5th ed. London & New York: Spon Press/Taylor & Francis, 2009. ISBN 9780415480215.
- Habets, E. A.: *Room Impulse Response Generator*

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2, rozpracovaný bod 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Mošner Ladislav, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Tato práce se zabývá simulací akustiky místností. V práci jsou nejprve teoreticky popsány existující přístupy k simulaci akustiky a srovnány jejich přednosti a nevýhody. K výpočtu odezvy místnosti bylo implementováno několik geometrických metod jako ray tracing a obrazová metoda, ale i kombinace těchto metod. Byla vytvořena aplikace s grafickým i textovým uživatelským rozhraním, která umožňuje provést simulaci v místnosti libovolného tvaru. Aplikace také umožňuje získat odezvu ve formě zvukového souboru, znázornit uživateli výsledky a postup simulace a provést auralizaci. Výstupy simulace byly porovnány s naměřenými odezvami skutečných místností. Při porovnání se ukázalo, že nejvyšší přesnosti z implementovaných metod dosahuje hybridní metoda ve středních nebo větších prázdných místnostech.

Abstract

This thesis deals with simulation of room acoustics. The first part of this thesis contains theoretical description of existing aproaches to simulation of acoustics and compares their strenghts and weaknesses. For the purpose of impulse response calculation, several geometrical methods were implemented, such as ray tracing, image method and a combination of those two methods. Application with graphical and text user interface was created, to allow simulation of rooms with arbitrary geometry. The application also allows user to obtain impulse response in a form of a sound file, to visualize the results and the process of sound simulation and to perform auralization. The results of the simulation were compared with measured impulse responses of real rooms. The comparison showed, that the hybrid method is the most accurate of methods implemented in this thesis, and that the best results are achieved by simulation of empty medium–sized or large rooms.

Klíčová slova

Akustika místností, Impulzní odezva, Simulace, Ray tracing, Obrazová metoda, Auralizace, Qt, OpenGL

Keywords

Room acoustics, Impulse response, Simulation, Ray tracing, Image method, Auralization, Qt, OpenGL

Citace

KRBILA, Martin. *Aplikace pro simulaci akustiky místnosti*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ladislav Mošner

Aplikace pro simulaci akustiky místnosti

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Ladislava Mošnera. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Krbila
15. května 2019

Poděkování

Chtěl bych poděkovat svému vedoucímu práce Ing. Ladislavu Mošnerovi za jeho pomoc s touto prací a také panu doktoru Igoru Szökemu za zapůjčení vybavení pro měření akustiky místností.

Obsah

Kapitola 1

Úvod

Cílem této práce je vytvoření aplikace pro simulaci akustiky místnosti. Aplikace musí být schopná načíst geometrický popis místnosti a následně provést simulaci šíření zvuku. Smyslem simulace zvuku místností je možnost studovat akustické parametry reálných i vymyšlených místností bez nutnosti měřit jejich vlastnosti ve skutečnosti. V simulaci bude zkoumána odezva přenosového kanálu mezi posluchačem a zdrojem zvuku na jednotkový impulz. Impulzní odezvu místnosti je možné použít například pro auralizaci[1], zjištění doby dozvuku místnosti nebo zkoumání útlumu hlasitosti zvuků vlivem prostředí. Práce se dále zabývá vytvořením grafické aplikace pro spouštění simulací a vizualizaci výsledků za pomoci *OpenGL*. Výsledky simulace budou porovnány s odezvami skutečných místností a vyhodnoceny silné a slabé stránky jednotlivých přístupů pro simulaci akustiky.

V první části práce budou vysvětleny pojmy impulzní odezva místnosti a auralizace a budou uvedeny některé metriky pro popis akustiky místností. Dále budou teoreticky shrnuty existující používané metody pro simulaci akustiky. Jsou zde popsány geometrické metody, ale i metody založené na vlnové simulaci. Dále jsou tyto metody srovnány z hlediska přesnosti a výpočetní náročnosti.

V kapitole 3 je popsána implementace tří geometrických metod použitých ve výsledné aplikaci. Z geometrických metod byla implementována simulace pomocí ray tracingu, obrazové metody a hybridního přístupu, který využívá jak ray tracing, tak obrazy zdroje zvuku.

V kapitole 4 je popsáno textové a grafické rozhraní aplikace. Textové rozhraní umožňuje spuštění aplikace z příkazové řádky, zatímco grafické rozhraní, poskytuje interaktivní zobrazení místnosti a možnosti nastavení všech parametrů simulace.

V kapitole 5 jsou srovnány výsledky simulace s naměřenými odezvami skutečných místností.

[1] Auralizace je proces, při kterém simulujeme zvukový vjem posluchače, jako by se nacházel v určité místnosti.

Kapitola 2

Existující přístupy k simulaci akustiky

V této kapitole budou shrnuty metody používané k simulaci akustiky místnosti. Zaměřil jsem se především na geometrické metody, konkrétně ray tracing a obrazovou metodu. Tyto metody sice zanedbávají vlnovou podstatu zvuku, ale jejich výhodou je jejich nízká výpočetní náročnost. Další uvedenou třídou metod jsou metody založené na simulaci šíření vln, které poskytují nejpřesnější výsledky za cenu vysoké výpočetní náročnosti [6]. Existují také statistické metody [4], které se ovšem používají spíše pro odhad šíření hluku v prostoru, a nejsou vhodné pro simulaci zvukového vjemu posluchače v místnosti (auralizaci) [4, 6].

2.1 Impulzní odezva místnosti

Cílem simulace akustiky místnosti je vytvoření impulzní odezvy místnosti. Impulzní odezva je signál získaný jako reakce systému na Diracův jednotkový impulz [9]. Impulzní odezva popisuje všechny změny, které signál podstoupí, když cestuje z jednoho bodu místnosti do druhého [9].

Protože Diracův impulz lze vytvořit jen teoreticky, je nutné použít pro měření jiný přístup. Nejjednodušší metodou je aproximace impulzu pomocí nějakého krátkého hlasitého zvuku jako tlesknutí, prasknutí balónku, apod. [1]. Problém tohoto přístupu je obtížná reprodukce naměřených výsledků a specifikace direktivity zdroje zvuku [1]. Další možností je přehrát v místnosti nějaký signál, a následně získat odezvu pomocí dekonvoluce naměřeného signálu s původním signálem [1]. Lze použít například takzvaný *exponential sine sweep*, nebo náhodný signál jako bílý šum [1].

U impulzních odezev můžeme zkoumat parametry jako například D (Definition), C (Clarity) nebo $RASTI$ (RApid Speech Transmission Index) [9].

Parametr D lze vypočítat podle následujícího vzorce [9]:

$$D_{50} = \frac{\int_0^{50\text{ms}} [g(t)]^2 \, dt}{\int_0^{\text{inf}} [g(t)]^2 \, dt} 100\%, \tag{2.1}$$

kde D_{50} je výsledný parametr D pro čas 50 ms a $g(t)$ je hodnota impulzní odezvy v čase t.

Parametr C můžeme vypočítat pomocí vzorce [9]:

$$C_{80} = 10 \log_{10} \left(\frac{\int_0^{80\text{ms}} [g(t)]^2 \, dt}{\int_{80\text{ms}}^{\text{inf}} [g(t)]^2 \, dt} \right), \tag{2.2}$$

kde C_{80} je výsledný parametr C pro čas 80 ms a $g(t)$ je hodnota impulzní odezvy v čase t.

2.1.1 Auralizace

Cílem auralizace je simulovat zvukový vjem posluchače, jako by se nacházel v určité místnosti. Pro základní auralizaci lze použít jednoduchou diskrétní konvoluci impulzní odezvy s libovolným jiným signálem podle vzorce:

$$(a * b)[n] = \sum_{i=0}^{\text{len}(a)} a[i] \cdot b[n-i], \tag{2.3}$$

kde a a b jsou vstupní signály a $\text{len}(a)$ označuje počet vzorků signálu a. Aby byl zvukový vjem realistický, je možné provést tzv. binaurální zpracování [9]. Při binuarálním zpracování se zohledňuje natočení hlavy posluchače v místnosti. Úkolem binaurálního zpracování je získat zvláštní signál pro levé a pravé ucho posluchače. Odezvu pro binaurální zpracování můžeme získat například pomocí mikrofonů umístěných v umělém modelu lidské hlavy [14]. V případě simulované impulzní odezvy je nutné znát směr odkud přicházejí jednotlivé impulzy a přenosovou funkci pro každé ucho [9].

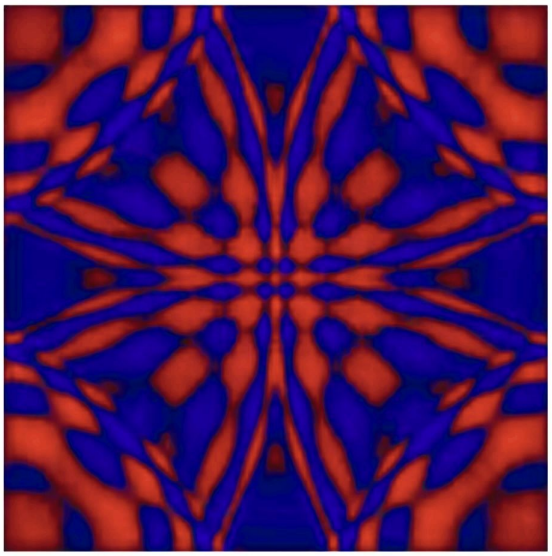
2.2 Metody založené na vlnové simulaci

Metody založené na vlnové simulaci vycházejí z rovnice šíření vln [17]. Do této kategorie patří například *FEM* (Finite element method) a *BEM* (Boundary element method). *FEM* rozděluje prostor na velké množství buněk, které jsou použity pro řešení vlnové rovnice. Doporučené množství těchto buněk je alespoň šest na jednu vlnovou délku [4]. Na Obrázku 2.1 je vidět výstup jednoduché 2D simulace pomocí metody *FEM*. Metody typu *BEM* nerozdělují celý prostor, ale pouze plochy místnosti. Buňky na stěnách místnosti se potom ovlivňují podle pravidel šíření vln [6]. Hlavní výhodou těchto metod je jejich přesnost. Protože jde o simulaci vlny jako takové, je možné modelovat všechny jevy, které se vyskytují při šíření vln ve vzduchu, jako je interference a difrakce. Nevýhoda spočívá především v jejich výpočetní náročnosti. U *FEM* potřebujeme modelovat místnosti ve třech rozměrech. K tomu potřebujeme třírozměrnou matici simulačních buněk, kde celkový počet simulačních buněk roste kubicky s frekvencí. To znamená, že metoda *FEM* je vhodná pro nízké frekvence nebo malé místnosti.

2.3 Ray tracing

Ray tracing je geometrická metoda, ve které je energie zdroje zvuku přenášena pomocí paprsků. Ze zdroje zvuku se vysílá velké množství paprsků (statisíce až miliony), které se odrážejí od stěn místnosti. Posluchač je reprezentován koulí, která detekuje průlet paprsků a započítává jejich energii.

Na začátku cesty každého paprsku se nastaví jeho energie. Paprsek je následně vyslán náhodným směrem do prostoru místnosti. Po vyslání paprsku může dojít k několika situacím. Paprsek se může odrazit od některého povrchu místnosti, vyletět ven z místnosti (ztratit se) a nebo narazit do posluchače. V případě, že paprsek narazí do nějakého povrchu, je třeba započítat absorpci povrchu a snížit energii paprsku. Další možností je náhodně zahazovat paprsky podle koeficientu absorpce povrchu [9]. Při zachování paprsku je nutné vypočítat



Obrázek 2.1: Experiment 2D simulace šíření vln ve čtvercové místnosti, impulz začíná ve středu místnosti a původní vlna se již odrazila od stěn a nyní se nachází těsně před bodem původu. Červené oblasti značí místa se zvýšeným akustickým tlakem, modrá místa značí oblasti se sníženým akustickým tlakem.

nový směr odraženého paprsku. Pokud uvažujeme jen spekulární odrazy[1], můžeme nový směr zjistit pomocí zákona odrazu.

Mezi hlavní výhody ray tracingu metody patří možnost výpočtu difuzních odrazů[2] a poměrně dobrá rychlost výpočtu ve srovnání s obrazovou metodou. V místnostech se složitou geometrií roste složitost lineárně s počtem odrazových ploch oproti exponenciálnímu růstu u obrazové metody. Hlavní nevýhodou této metody je, že zanedbává vlnovou podstatu zvuku a jevy jako interference a difrakce. Další nevýhodou je hledání optimální velikosti posluchače, která závisí na velikosti místnosti, vzdálenosti posluchače od zdroje, výpočetním výkonu a tvaru místnosti. Příliš malý poloměr znamená, že jen velmi málo paprsků zasáhne posluchače a výpočet bude neefektivní. Příliš velký poloměr naopak znamená, že posluchač bude zaznamenávat i paprsky, které by ho správně neměly zasáhnout a výsledek bude nepřesný.

2.3.1 Difuzní odrazy

Ray tracing také oproti ostatním metodám podporuje difuzní odrazy [11]. Difuzní odrazy paprsku umožňují modelovat nerovné a hrubé povrchy, aniž by bylo nutné je fyzicky dělit na více menších trojúhelníků. Existuje několik metod simulace difuzních odrazů.

První možností je při každém odrazu paprsku generovat jak jeho spekulární, tak jeho difuzní složku. To znamená, že po každém odrazu vzniká jeden spekulární paprsek odražený přesně podle úhlu dopadu a libovolné množství druhotných paprsků rovnoměrně rozložených v poloprostoru daném rovinou trojúhelníku, do kterého paprsek narazil, a směrem paprsku [12]. Tento proces poskytuje poměrně přesné výsledky, ale zásadně zvyšuje výpo-

[1] Spekulární odraz je takový odraz, při kterém se úhel dopadu rovná úhlu odrazu.
[2] Difuzní odraz je takový odraz, při kterém se uvažuje hrubost povrchu.

četní složitost, protože množství paprsků roste exponenciálně s každým dalším odrazem [12].

Druhou možností je zachovat při každém odrazu jediný paprsek. O paprsku se při každém odrazu rozhodne jestli bude odražen spekulárně nebo difuzně. Rozhodnutí probíhá tak, že vygenerujeme náhodné číslo s rovnoměrným rozložením v intervalu $\langle 0, 1\rangle$ a tuto hodnotu porovnáme s „hrubostí“ povrchu, která je také reprezentována hodnotou z intervalu $\langle 0, 1\rangle$. To je výpočetně mnohem méně náročné, protože při každém odrazu se zachovává jen jeden paprsek, ale nedosahuje se takové přesnosti [12].

Třetí možností implementace difuzních odrazů je lineární kombinace vektorů. Nejprve je vygenerován vektor, který odpovídá rozložení pravděpodobnosti podle Lambertova modelu [15]. Druhý vektor je přesným odrazem původního paprsku podle zasažené plochy. Výsledný vektor, a tedy směr odraženého paprsku, je lineární kombinací obou vektorů [12]:

$$\overrightarrow{dir} = (p) \cdot \overrightarrow{diff} + (1 - p) \cdot \overrightarrow{spec} \tag{2.4}$$

V rovnici (2.4) je uveden výpočet nového směru paprsku $\overrightarrow{dir}$. Koeficient p značí „hrubost“ materiálu a vektory $\overrightarrow{diff}$ a $\overrightarrow{spec}$ označují postupně náhodný difuzní paprsek a přesně odražený spekulární paprsek.

Další možností je využití techniky „diffuse rain“ [12]. Při každém odrazu paprsku ověříme, jestli je z místa dopadu viditelný příjemce. Pokud ano, započítáme energii paprsku, přičemž spekulární paprsek pokračuje dál. Energii difuzně odraženého paprsku můžeme vypočítat pomocí Lambertova modelu [15], který se používá i v počítačové grafice pro osvětlování „matných“ povrchů [15]:

$$I_D = \overrightarrow{L} \cdot \overrightarrow{N} I_L, \tag{2.5}$$

kde $\overrightarrow{L}$ je směr zdroje energie, $\overrightarrow{N}$ je normála plochy a I_L je energie zdroje. Výhodou tohoto přístupu je nízká výpočetní náročnost a také fakt, že získáme informaci i z paprsků, které se spekulárními odrazy k posluchači nikdy nedostanou. Nevýhodou této metody je, že vzniká poměrně velká aproximace. Ze všech možných difuzních paprsků, které by se od povrchu mohly odrazit, započítáváme vždy jen ten, který míří přímo na posluchače. Případné difuzní paprsky, které by se dalším odrazem dostaly k posluchači, jsou tedy zanedbány.

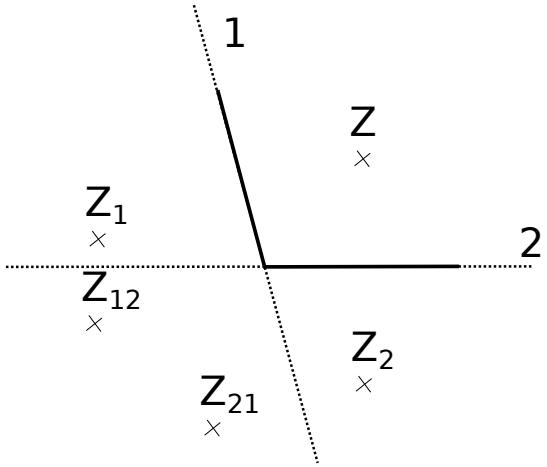
2.4 Obrazová metoda

Obrazová metoda je stejně jako ray tracing geometrická metoda, která také zanedbává vlnovou podstatu zvuku. Na rozdíl od ray tracingu se obrazová metoda snaží nalézt všechny možné cesty mezi příjemcem a zdrojem zvuku „hrubou silou“.

V první fázi obrazová metoda vygeneruje obrazy z původního zdroje zvuku. Pro zdroj zvuku je možné vygenerovat jeho přímé obrazy převrácením podle všech ploch v místnosti, jak je znázorněno na Obrázku 2.2. Takto můžeme postupovat pro každý další zdroj zvuku, dokud nedosáhneme požadovaného řádu odrazu. Jedná se tedy o prohledávání do šířky. Počet obrazů lze vyčíslit jako [12]

$$\sum_{k=1}^{K} N(N-1)^{k-1}, \tag{2.6}$$

kde K je maximální řád odrazu a N je počet ploch místnosti. Ze vztahu (2.6) vidíme, že paměťová i časová složitost obrazové metody roste exponenciálně s řádem odrazu.



Obrázek 2.2: Příklad zdrojů zvuku vygenerovaných obrazovou metodou. Místnost obsahuje dvě stěny označené 1,2 a původní zdroj zvuku Z. Z_1 a Z_2 jsou odražené zdroje zvuku prvního řádu.

V druhé fázi výpočtu je nutné projít všechny získané zdroje zvuku a zjistit, jestli jsou viditelné. Viditelnost zdroje je možné testovat vysláním paprsku od posluchače k vybranému zdroji. Paprsek musí narazit do plochy, podle které byl testovaný zdroj zvuku odražen, jinak není viditelný. Dále je nutné pokračovat z bodu nárazu ke zdroji zvuku, který je předkem aktuálně testovaného zdroje a takto postupovat až k původnímu zdroji. Při ověřování viditelnosti může nastat situace, kdy zdroj nižšího řádu není viditelný, ale zdroje z něj vygenerované už viditelné jsou [12]. Proto není možné ušetřit výpočetní výkon rozgenerováním jen viditelných zdrojů zvuku.

Příspěvek jednoho zdroje do odezvy místnosti je možné vyjádřit pomocí následujícího výrazu [9]:

$$\frac{E}{4\pi r^2_{m_1 m_2 \ldots m_i}} \rho_{m_1} \rho_{m_2} \ldots \rho_{m_i} \delta\left(t - \frac{r_{m_1 m_2 \ldots m_i}}{c}\right), \tag{2.7}$$

kde E značí energii zdroje zvuku, c je rychlost zvuku, $r_{m_1 m_2 \ldots m_i}$ je celková vzdálenost zdroje od posluchače, m_x jsou stěny podle kterých byl zdroj zrcadlen, ρ_{m_x} značí koeficient odrazivosti povrchu a $\delta(t)$ je jednotkový impulz v čase t. Energie zdroje se řídí zákonem sférické disperze [9] a klesá s druhou mocninou vzdálenosti. Čas příchodu impulzu je závislý na vzdálenosti a rychlosti zvuku.

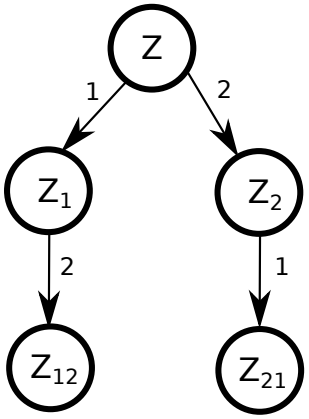
Hlavní nevýhoda obrazové metody plyne ze způsobu generování zdrojů zvuku, jejichž počet roste exponenciálně s řádem odrazu. Obrazová metoda je tedy nevhodná pro výpočet vyšších řádů odrazu nebo simulaci v místnostech se složitou geometrií. Další nevýhodou obrazové metody je skutečnost, že většina obrazů vyšších řádu je pro posluchače neviditelná. Obrazová metoda také neumožňuje simulovat difuzní odrazy. Výhodou obrazové metody je, že nalezne vždy všechny možné cesty mezi posluchačem a zdrojem zvuku do určitého řádu a je tedy velmi přesná.

2.5 Kombinované metody

Všechny zmíněné metody mají určité nedostatky, které je možné zredukovat jejich vhodnou kombinací. Jednou možností je generování obrazů zdroje zvuku v obrazové metodě pomocí ray tracingu [9]. Základní obrazová metoda má velké problémy s generováním obrazů vyšších

řádů. Jednak počet zdrojů zvuku roste exponenciálně s řádem odrazu, ale navíc je většina zdrojů vyšších řádů neplatná, takže jejich hledáním se plýtvá jak paměť tak čas.

Řešením tohoto problému je provést vyhledávání zdrojů pomocí ray tracingu [9]. S takovým přístupem je možné zdroje do druhého nebo třetího řádu vygenerovat hrubou silou, což nebude tak náročné na výpočet a následně použít ray tracing k vyhledávání zdrojů vyšších řádů. Vzhledem k tomu, že ray tracing vyžaduje, aby byl posluchač reprezentován nějakým objemem, mohou být některé takto vzniklé zdroje neplatné. Lze usuzovat, že tato metoda bude mít výrazně vyšší úspěšnost v poměru nalezených platných zdrojů k neplatným zdrojům.



Obrázek 2.3: Strom obrazů odpovídající situaci na Obrázku 2.2. Kořenem stromu je původní zdroj. U hran stromu je zaznačeno podle jakých ploch byly vytvořeny synovské uzly.

Tato kombinace ovšem přináší i některé nové problémy. Ray tracing totiž nezaručuje nalezení všech cest, proto je nutné vysílat paprsky vhodně dlouhou dobu. Dalším problémem je, že ray tracing nezaručuje, že posluchač bude danou cestou zasažen pouze jednou. Z toho důvodu je nutné najít způsob, kterým by bylo možné vyřadit paprsky, které prošly stejnou cestou odrazů jako některý předchozí paprsek. Zdroje zvuku můžeme uchovávat ve stromové struktuře (viz Obrázek 2.3), kde každý prvek bude znát své následníky včetně toho, podle jakých ploch místnosti byly zdroje vytvořeny. Pokud paprsek zasáhne posluchače vložíme zdroj který by vznikl odpovídající posloupností odrazů do stromu. Tímto způsobem je možné určit, jestli byl daný zdroj již nalezen.

Další možnou kombinací je spojení metod *FEM* nebo *BEM* s některou geometrickou metodou [4]. Takto můžeme pro nízké frekvence provést simulaci některou z metod založených na vlnové simulaci. Pro vyšší frekvence (kde je vlnová délka mnohem kratší v poměru k velikosti překážek v místnosti) můžeme použít méně přesnou geometrickou metodu.

Další možností je opačné spojení ray tracingu a obrazové metody [4], kde první odrazy jsou přesně vypočteny pomocí obrazové metody a na pozdější odrazy je použit ray tracing, který už může obsahovat difuzní paprsky.

Kapitola 3

Implementace simulátoru

V této kapitole jsou popsány postupy použité pro implementaci ray tracingu, obrazové metody a také hybridní metody. Jako programovací jazyk bylo zvoleno *C++* z důvodů, že jde o jazyk překládaný do strojového kódu, a lze tak očekávat vysoký výkon. *C++* má vestavěné možnosti paralelního programování, objektové orientace. Také pro něj existují knihovny pro tvorbu grafického uživatelského rozhraní, jako je například *Qt*. Definice vyjmenovaných metod se nachází v souboru `room.h`, který obsahuje třídu `Room` pro reprezentaci místnosti. Pro výpočty ve 3D prostoru byla využita knihovna *GLM* (OpenGL mathematics), která obsahuje typy kompatibilní s knihovnou *OpenGL*, ale také funkce pro základní transformace ve 3D prostoru.

3.1 Zpracování místnosti

Protože zvolené metody (ray tacing, obrazová a hybridní metoda) jsou geometrické metody, vyskytují se u obou podobné implementační problémy. V této kapitole budou podrobněji popsány.

3.1.1 Načítání místnosti

Prvním krokem celé simulace je načtení místnosti ze souboru. Pro tento účel byl zvolen formát *Wavefront OBJ* [3]. Souborový formát *OBJ* slouží k popisu geometrie a je podporován mnoha grafickými 3D editory jako jsou *Blender, Maya* a *3DsMax*. V těchto editorech je možné vymodelovat libovolnou místnost a následně ji exportovat do *OBJ*.

Formát *OBJ* obsahuje mnoho druhů konstrukcí, proto bylo řešeno zpracování pouze těch konstrukcí, které jsou relevantní pro simulaci akustiky. Příklad souboru *OBJ* obsahující jeden trojúhelník:

```
[frame=single]
                                # Komentář
mtllib tri.mtl                  # Import knihovny materiálů
o Trojuhelnik                   # Nový objekt

v -1.000000 0.000000 -1.000000 # Vertex 1
v 1.000000 0.000000 1.000000   # Vertex 2
v -1.000000 0.000000 1.000000  # Vertex 3
```

```
vn 0.000000 -1.000000 0.000000 # Normála vertexu 1

usemtl Material                # Použití materiálu
f 2//1 3//1 1//1               # Nový polygon
                               # [vertex]/[UV]/[normála]
```

Komentář je uvozen znakem „`#`“ a platí do konce řádku. Další důležité konstrukce jsou „`o <název>`“, která uvozuje začátek nového objektu, „`v <X> <Y> <Z>`“, která popisuje nový vertex a „`f`“, která značí polygon. Polygony mohou být v tomto formátu zapsány libovolným počtem vrcholů, přičemž každý vrchol je dán třemi indexy oddělenými znakem „`/`“. Při načítání souboru jsou zpracovány jen indexy do pole vertexů, protože normály vertexů nemusí odpovídat normále plochy z důvodů hladkého stínování objektu a UV koordináty jsou potřebné jen pro vykreslování textur.

3.1.2 Převod polygonů na trojúhelníky

Vzhledem k tomu, že načtený soubor může obsahovat obecný polygon, u kterého navíc není zaručeno, že se jeho vrcholy nacházejí v jedné rovině, je nutné polygon převést na jednotlivé trojúhelníky. To je dobré udělat zvláště kvůli možnosti použít jednodušší algoritmus pro detekci kolize paprsku s povrchem místnosti. Detekce kolize s obecným polygonem je mnohem náročnější, než detekce kolize s trojúhelníkem. Dalším důvodem převodu polygonů na trojúhelníky je jejich následné vykreslení pomocí *OpenGL*, které se značně zjednoduší pokud budou všechny prvky scény (trojúhelníky) mít stejný počet vertexů. Dělení obecného polygonu na trojúhelníky je ovšem složitá operace. Moje implementace předpokládá pouze konvexní n-úhelníky, které jsou ve 3D modelech nejběžnější. Pro triangulaci polygonu byl zvolen nejjednodušší možný algoritmus, který předpokládá konvexní polygon zadaný n-ticí vektorů $P = (V_1, V_2, V_3, ..., V_n), n \in N \wedge n \geq 3$, kde platí, že pouze $\{V_i, V_{mod(i,n)+1}\}, i \in \{1, 2, ..., n\}$ tvoří hranu. Výsledný algoritmus triangulace vypadá následovně:

Algoritmus 1: Jednoduchý algoritmus triangulace

Vstup : Uspořádaná n-tice P obsahující n vrcholů, kde $n \geq 3$
Výstup: Množina trojúhelníků T

```
1 while N > 3 do
2     Ulož do Y první prvek množiny P;
3     Ulož do Z třetí prvek množiny P;
4     Vyber z množiny P druhý prvek a ulož ho do X;
5     Vlož do T trojúhelník {X, Y, Z};
6 Vlož do T trojúhelník daný třemi prvky P;
```

Výhodou tohoto algoritmu je jeho jednoduchost, časová složitost $O(n)$, a snadná implementace. Nevýhodou je, že ho lze použít jen pro konvexní polygony. Nově vzniklé hrany směřují vždy k jednomu vrcholu, což není vhodné zejména pro vykreslování osvětlení na takovém povrchu. Pro účely simulace zvuku je ovšem taková topologie dostačující. Uvedený algoritmus by bylo možné dále vylepšit například náhodnou volbou vrcholu, který bude odebrán, čímž by se zabránilo vzniku „vějířovité“ topologie. V případě, že uživatel potřebuje mít v místnosti nějaký nekonvexní polygon, může při exportu modelu využít triangulaci, kterou poskytuje jím zvolený editor, aby se ujistil, že převod proběhl korektně.

3.1.3 Zdroj zvuku a posluchač

Zdroje zvuku jsou popsány strukturou `Source`. Tato struktura obsahuje pozici zdroje a jeho energii. U původního zdroje se předpokládá energie 1. Posluchač je reprezentován strukturou `Listener`. Struktura `Listener` obsahuje pozici příjemce a jeho poloměr, který je využit při ray tracingu. Obě tyto struktury slouží jako parametry pro metody realizující simulaci.

3.1.4 Uložení geometrie

Geometrie místnosti je uložena v poli struktur `Triangle`. Struktura obsahuje tři indexy, které ukazují do pole vertexů, dále vektor značící normálu trojúhelníku, identifikátor materiálu trojúhelníku a identifikátor plochy ve které se trojúhelník nachází.

Jednotlivé vertexy jsou reprezentovány pomocí struktury `glm::dvec3`, která obsahuje tři koordináty typu *double*.

Materiály povrchů jsou popsány strukturou `Material`. Struktura obsahuje název materiálu, údaj o odrazivosti povrchu, kde 0 znamená plná absorpce a 1 plná odrazivost. Dále obsahuje informaci o hrubosti povrchu, která ovlivňuje procento difuzních paprsků.

Plochy jsou uloženy pomocí pole struktur `Plane`. Tato struktura obsahuje pouze normálu a jeden bod, který leží na ploše.

3.2 Navzorkování odezvy

Obě použité metody generují jako výsledek simulace objekt typu `Response`, popsaný v souboru `response.h`. Objekt `Response` obsahuje kolekci dvojic (čas→energie), seřazenou podle času. Pokud potřebují jednotlivé metody zapsat nový impulz, použijí metodu `add_impulse()`.

Z tohoto objektu je následně možné získat navzorkovaný výsledek simulace pomocí metod `sample()` a `sample_lpf()`, které vracejí pole hodnot typu *double*. Obě tyto metody mají jako parametr požadovanou vzorkovací frekvenci. První z těchto funkcí každému impulzu přidělí vzorek podle vztahu $i_s = floor(t_s f_v + 0.5), t_s >= 0$, kde t_s je čas příchodu impulzu, f_v je vzorkovací frekvence a $floor()$, je funkce která zaokrouhlí výsledek dolů na celé číslo.

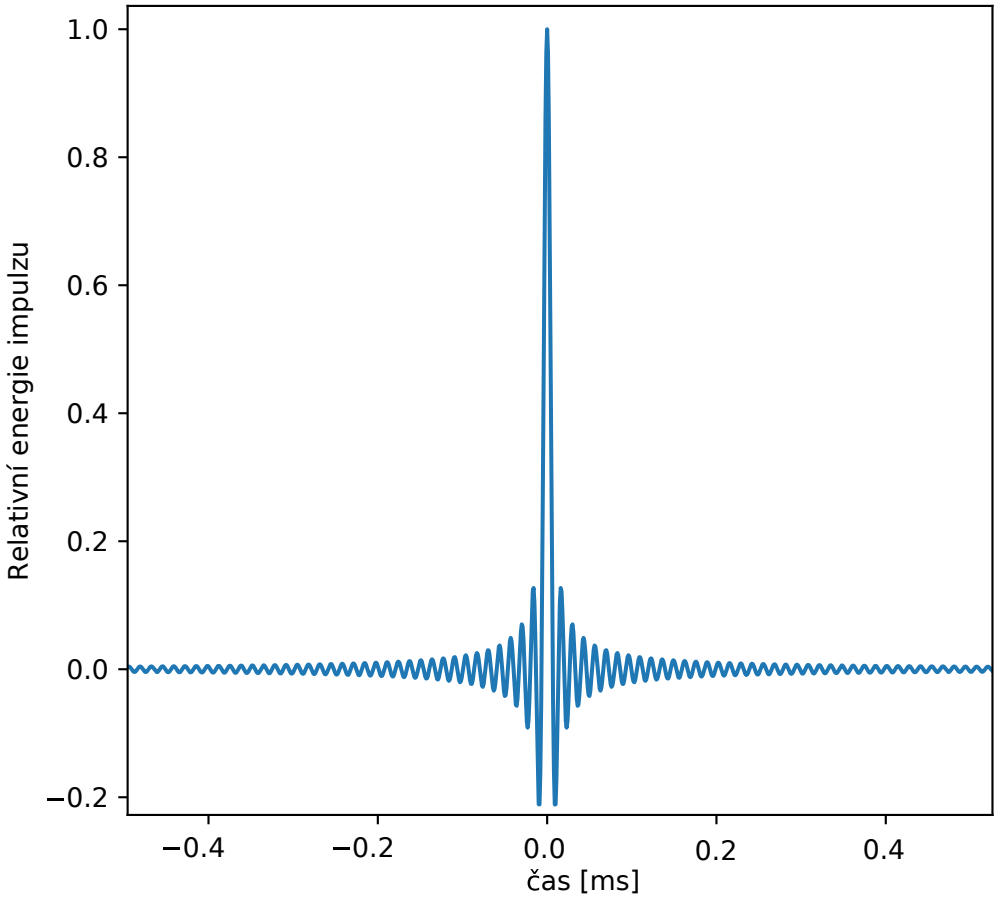
Funkce `sample_lpf()` prokládá každý impuls impulzní odezvou filtru dolní propusť. Každý vzorek je proložen kardinálním sinem (Obrázek 3.1) podle funkce uvedené v [6]:

$$\delta(t) = \begin{cases} \frac{1}{2}\left(1 + \cos\left(\frac{2\pi t}{T_w}\right)\right) \operatorname{sinc}(2\pi f_c t) & \text{pro } -\frac{T_w}{2} < t < \frac{T_w}{2} \\ 0 & \text{jinak} \end{cases}, \tag{3.1}$$

kde T_w je šířka okna, ve kterém je funkce nenulová (byla nastavena na 4 ms [6]) a f_c je maximální frekvence, kterou filtr propustí. f_c byla nastavena jako polovina vzorkovací frekvence [6].

3.3 Ray tracing

Ray tracing je implementován v metodě `calculate_energy_response()`. Tato metoda požaduje jako parametry zdroj zvuku, příjemce, počet paprsků a maximální počet odrazů. Každý paprsek nese informaci o bodu, ze kterého vyšel, směr kterým letí, celkovou uraženou vzdálenost a svou relativní energii.



Obrázek 3.1: Tvar impulzu při použití filtru (3.1).

3.3.1 Generování paprsků

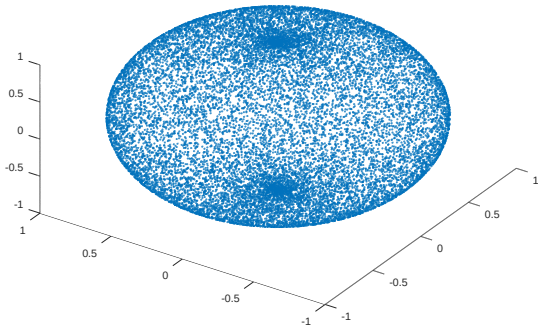
Na začátku cesty každého paprsku je nutné vygenerovat jeho směr ze zdroje zvuku. K tomu je potřeba metoda, která bude generovat náhodný bod v jednotkové kouli s uniformním rozložením. Lze využít Archimédův teorém [10]:

Teorém 1 *Plocha koule se rovná ploše každého kolmého válce opsaného kolem této koule bez podstav.*

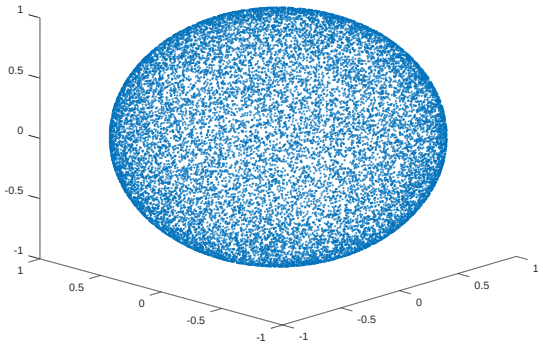
Platí, že jakýkoliv region na ploše jednotkové koule má po projekci na opsaný válec bez podstav stejnou plochu [10]. Pokud tedy vygenerujeme náhodný bod s uniformním rozložením na plášti válce, můžeme ho pomocí projekce převést na bod koule a uniformní rozložení bude zachováno. Pro generování náhodného bodu na plášti válce stačí vygenerovat souřadnici $z \in \langle -1, 1 \rangle$ a azimut $\varphi \in \langle 0, 2\pi \rangle$. Souřadnice z přesně odpovídá souřadnici bodu na kouli. Souřadnice x a y získáme pomocí projekce [2]:

$$(x, y, z) = \left(\sqrt{1 - z^2} \cos \varphi, \sqrt{1 - z^2} \sin \varphi, z \right) \tag{3.2}$$

Pokud chceme generovat náhodný vektor v polokouli stačí vygenerovat souřadnici $z \in \langle 0, 1 \rangle$. Obrázky 3.2 a 3.3 porovnávají jednotlivé přístupy pro generování bodů na jednotkové kouli.



Obrázek 3.2: Body generované pomocí azimutu a elevace. Lze vidět zvýšený výskyt bodů na pólech jednotkové koule, který neodpovídá uniformnímu rozložení.



Obrázek 3.3: Body generované pomocí projekce na plášť válce s uniformním rozložením.

3.3.2 Detekce kolize paprsku s trojúhelníkem

Pro detekci kolize paprsku s trojúhelníkem byl použit Möller–Trumborův test [13]. Tento test nejprve určí, zda došlo k průniku přímky s plochou a následně vypočte barycentrické koordináty (u, v) bodu průniku. Pokud $u \in \langle 0, 1 \rangle \wedge v \in \langle 0, 1 \rangle \wedge u + v \in \langle 0, 1 \rangle$, nachází se bod uvnitř trojúhelníku. Následně je vypočten parametr t z rovnice přímky $\overleftrightarrow{p} = \overrightarrow{origin} + \overrightarrow{dir} \cdot t$, pomocí kterého lze přesně určit, kde ke kolizi došlo.

Při detekci kolize paprsku a trojúhelníku je nutné určit takový trojúhelník, pro který platí, že parametr t získaný při detekci kolize je nejnižší ze všech trojúhelníků, ale zároveň je nezáporný. Tedy uvažujeme jen trojúhelníky, které se nacházejí před výchozím bodem paprsku ve smyslu směru paprsku.

Dalším problémem je detekce kolize paprsku a koule. To je nutné k určení nárazu paprsku do posluchače. Následující test byl odvozen ze středové rovnice koule a parametrické rovnice přímky.

$$(x - a)^2 + (y - b)^2 + (z - c)^2 - r^2 = 0 \tag{3.3}$$

$$x = p_x + t \cdot d_x \tag{3.4}$$

$$y = p_y + t \cdot d_y \tag{3.5}$$

$$z = p_z + t \cdot d_z \tag{3.6}$$

Dosazením rovnic (3.4), (3.5), (3.6) do rovnice (3.3) získáme kvadratickou rovnici, pomocí které můžeme určit, zda došlo ke kolizi. Z kořenů vzniklé rovnice lze odvodit dva body průniku dosazením do rovnice přímky. Z těchto bodů je vybrán ten, který je bližší počátku paprsku a má kladný parametr t. Následně je rozhodnuto, jestli se výsledný bod nachází před nejbližším nalezeným trojúhelníkem a pokud ano, došlo ke kolizi s koulí.

3.3.3 Odraz paprsku

Pokud je detekován náraz paprsku na některý trojúhelník, je snížena jeho energie E podle koeficientu odrazu m_{refl} příslušného materiálu, kde $m_{refl} \in \langle 0, 1\rangle$ a $m_{refl} = 1$ značí odraz 100% energie :

$$E_n = E \cdot m_{refl}$$

Následně je vypočítán nový směr paprsku. Zde jsou dvě možnosti odrazu. Může se jednat o spekulární nebo difuzní odraz. V případě spekulárních odrazů získáme nový směr pomocí zákona odrazu. Směrový vektor získáme pomocí rovnice:

$$\overrightarrow{dir_{n+1}} = \overrightarrow{dir_n} - 2 \cdot \mathrm{dot}(\overrightarrow{norm}, \overrightarrow{dir_n}) \cdot \overrightarrow{norm} \tag{3.7}$$

kde $\overrightarrow{norm}$ je normála plochy, $\overrightarrow{dir_n}$ je aktuální směr paprsku a $\overrightarrow{dir_{n+1}}$ je nový směr paprsku. Pokud je normála plochy vektor délky jedna, skalární součin normály a směru paprsku určuje vzdálenost bodu $\overrightarrow{dir_i}$ od plochy s normálou $\overrightarrow{norm}$ procházející počátkem souřadného systému. Vynásobíme-li tuto vzdálenost mínus dvěma, zjistíme o kolik musíme vektor posunout ve směru normály plochy, abychom ho převrátili podle plochy. Tuto hodnotu tedy vynásobíme normálou a přičteme k původnímu vektoru. Rovnice (3.7) je použita ve funkci knihovny *GLM* zvané `reflect()`.

Difuzní model byl implementován pomocí druhého postupu uvedeného v kapitole 2.3.1. Paprsek se tedy může odrazit spekulárně nebo difuzně s určitou pravděpodobností řízenou hrubostí materiálu. Směr difuzních paprsků je náhodně vygenerovaný v jednotkové polokouli, jejíž pól je orientován stejným směrem jako normála povrchu. V kapitole 3.3.1 bylo ukázáno generování náhodného vektoru v jednotkové kouli a polokouli. Polokoule je ovšem orientována podle vektoru $\overrightarrow{v} = (0, 0, 1)$, zatímco polokoule kterou potřebujeme je orientována podle normály plochy, která může směřovat libovolným směrem. Aby bylo možné provést rotaci, je vypočteno jaký úhel svírají normála plochy $\overrightarrow{n}$ a vektor $\overrightarrow{v} = (0, 0, 1)$.

$$\varphi = \arccos\left(\frac{\overrightarrow{n} \cdot \overrightarrow{v}}{|\overrightarrow{n}| \cdot |\overrightarrow{v}|}\right) \tag{3.8}$$

Pokud $\varphi \approx 0$, není potřeba vektor vůbec rotovat. Pokud je $\varphi \approx \pi$, stačí vektor obrátit (vynásobit -1). V ostatních případech je nutné provést rotaci. Prvním krokem této rotace je nalezení rotační osy. Vektor je nutné rotovat v rovině dané vektory $\overrightarrow{n}$ a $\overrightarrow{v}$. Rotační osu lze získat jednoduše pomocí vektorového součinu, jehož výsledkem je vektor kolmý na oba původní vektory. Posledním krokem je samotné provedení rotace. K tomu je možné využít rotační matici [16]:

$$R = \begin{bmatrix} \cos\varphi + u_x^2(1-\cos\varphi) & u_x u_y(1-\cos\varphi) - u_z\sin\varphi & u_x u_z(1-\cos\varphi) + u_y\sin\varphi \\ u_y u_x(1-\cos\varphi) + u_z\sin\varphi & \cos\varphi + u_y^2(1-\cos\varphi) & u_x u_z(1-\cos\varphi) - u_x\sin\varphi \\ u_z u_x(1-\cos\varphi) - u_y\sin\varphi & u_z u_y(1-\cos\varphi) + u_x\sin\varphi & \cos\varphi + u_z^2(1-\cos\varphi) \end{bmatrix} \tag{3.9}$$

Dosazením rotační osy $\overrightarrow{u}$ a úhlu φ, o který se polokoule vychyluje od normály plochy, získáme výslednou rotační matici, kterou můžeme vynásobit náhodně vygenerovaný vektor v polokouli. Tím získáme náhodný vektor v polokouli orientované podle plochy. Rotace je implementována ve funkci knihovny *GLM* nazvané `rotate()`.

3.4 Obrazová metoda

Obrazová metoda je rozdělená do dvou metod třídy `Room`, konkrétně `generate_img_srcs()` a `calculate_energy_response_img()`. První ze zmíněných metod se stará o vytvoření zdrojů zvuku a požaduje jako parametr výchozí zdroj zvuku. Druhá metoda slouží k ověření validity zdrojů z pozice posluchače a výpočet odezvy.

3.4.1 Slučování polygonů

Pro efektivní fungování obrazové metody je nutné, aby místnost měla co nejméně povrchů, podle kterých se zrcadlí zdroj zvuku. To je zjevné ze vztahu popisujícího růst počtu obrazů zvuku (2.6). Čím vyšší základ exponenciální funkce, tím menší řád odrazu bude možné vypočítat. Naivní řešení zrcadlení podle každého trojúhelníku umožní generovat zdroje jen velmi nízkých řádů už i v relativně velmi jednoduchých geometriích místností. Řešením tohoto problému je sdružování trojúhelníků, které leží v jedné rovině. Algoritmus slučování polygonů vypadá následovně:

Algoritmus 2: Algoritmus slučování trojúhelníků

Vstup : Množina trojúhelníků T
Výstup: Množina ploch P
1 **foreach** $i \in T$ **do**
2 **if** $p \notin P, i \in p$ **then**
3 Přidej do P plochu a obsahující trojúhelník i;
4 **foreach** $j \in T, j \neq i$ **do**
5 **if** $(p \notin P, j \in p) \wedge (j$ *leží v ploše* $a)$ **then**
6 Přidej j do plochy p;

Algoritmus 2 má v nejhorším případě složitost $\mathrm{O}(n^2)$, a to pokud jsou všechny trojúhelníky ve vzájemně rozdílných plochách. Pokud jsou všechny trojúhelníky v jedné ploše bude složitost pouze $\mathrm{O}(n)$. V reálném modelu místnosti se dá předpokládat existence množství ploch rozdělených na menší trojúhelníky, proto se reálná složitost pohybuje někde mezi těmito časovými složitostmi.

Posledním problémem, který zbývá vyřešit je detekce případu kdy dva trojúhelníky leží ve stejné ploše. Jelikož známe normály trojúhelníků, můžeme je porovnat a zjistit, jestli směřují stejným směrem. Pomocí vzorce

$$\varphi = \arccos\left(\frac{|\overrightarrow{u} \cdot \overrightarrow{v}|}{|\overrightarrow{u}| \cdot |\overrightarrow{v}|}\right) \tag{3.10}$$

zjistíme odchylku přímek daných těmito dvěma vektory. Pokud $\varphi \cong 0$, můžeme pokračovat, jinak trojúhelníky nejsou v jedné ploše. Nyní víme, že trojúhelníky se nacházejí ve dvou rovnoběžných plochách a musíme určit, jestli jsou plochy totožné. Vypočteme vzdálenost bodů z každého trojúhelníku od jedné z rovnoběžných ploch procházejících středem souřadného systému:

$$\left(\overrightarrow{n} \cdot \overrightarrow{ta_x}\right) \approx \left(\overrightarrow{n} \cdot \overrightarrow{tb_y}\right) \tag{3.11}$$

Pokud je tento výraz pravdivý, trojúhelníky se nacházejí v jedné ploše. $\overrightarrow{n}$ označuje společnou normálu obou trojúhelníků, $\overrightarrow{ta_x}$ a $\overrightarrow{tb_y}$ jsou libovolné body prvního a druhého trojúhelníku.

3.4.2 Generování zdrojů

Generování zdrojů probíhá v metodě `generate_img_srcs()`. Metoda implementuje prohledávání do šířky, kde každý zdroj zvuku je postupně zrcadlen podle všech ploch v místnosti. V případě, že jde o již zrcadlený zdroj zvuku, nebude generován obraz podle stejné plochy dvakrát. Tato metoda vyžaduje znalost umístění zdroje zvuku, geometrii místnosti a limit počtu zrcadlení. Kompletní algoritmus generování zdrojů vypadá následovně:

Algoritmus 3: Algoritmus generování zdrojů zvuku

Vstup : Množina ploch $P \neq \emptyset$, Výchozí zdroj zvuku s_0, Limit počtu zrcadlení lim
Výstup: Uspořádaná množina zdrojů zvuku S

```
1 Vlož do S zdroj s_0;
2 Nastav index i na první prvek S;
3 while Zdroj na indexu i má nižší řád než lim do
4     foreach p ∈ P do
5         if Zdroj na indexu i není odražen podle p then
6             Vlož na konec S zdroj na indexu i zrcadlený podle p;
7     Posuň index i na další prvek;
```

3.4.3 Uložení obrazů

Obrazy zdroje zvuku jsou v mém programu reprezentovány strukturou `Img_source`. Struktura obsahuje pozici zdroje, index plochy podle které byl tento zdroj zrcadlen, údaj o tom, jestli je viditelný pro posluchače, energii zdroje, řád odrazu a index předchozího zdroje. Pro kterýkoliv zdroj můžeme pomocí tohoto indexu najít cestu zpět k původnímu zdroji. Pomyslný jednosměrně provázaný strom zdrojů je uložen ve standardním vektoru.

3.4.4 Validita zdrojů

Před vytvořením impulzní odezvy je nutné určit, které zdroje jsou pro příjemce viditelné a které nikoliv. To řeší metoda `is_visible()`, která rozhodne pomocí ukazatele na zdroj zvuku a pozice posluchače, jestli je zdroj viditelný.

Hlavní smyčka metody prochází postupně od zkoumaného zdroje až ke kořenovému zdroji. Na začátku je určen výchozí bod a směr testovacího paprsku. Paprsek tedy směřuje z pozice příjemce ke zkoumanému zdroji. Pomocí paprsku je proveden test na průnik s geometrií místnosti. Z trojúhelníku, do kterého paprsek narazil, získáme index plochy ve které trojúhelník leží. Index se musí shodovat s indexem ve zdroji zvuku, který popisuje podle které plochy byl zrcadlen. Pokud se indexy neshodují, znamená to, že zdroj není z pozice posluchače viditelný. V opačném případě se nastaví nový výchozí bod paprsku na bod dopadu předešlého paprsku a směr nového paprsku odpovídá spekulárnímu odrazu paprsku jako při ray tracingu. Smyčka pokračuje s předchozím zdrojem v hierarchii zdrojů.

Jakmile se hlavní smyčka dostane ke kořenovému zdroji, je ukončena a provede se poslední test na kolizi s geometrií místnosti. Paprsek začíná na posledním bodu odrazu a směřuje do kořenového zdroje. Celý test je limitován vzdáleností výchozího bodu paprsku od zdroje zvuku a zjišťuje, jestli se mezi zdrojem zvuku a bodem posledního dopadu nachází žádná překážka. Pokud je tento test negativní, původní zdroj je viditelný.

3.5 Hybridní metoda

Kvůli vysoké časové a paměťové složitosti obrazové metody byla implementována jedna z hybridních metod zmíněných v 2.5. Konkrétně použití ray tracingu pro nalezení obrazů zdroje zvuku. Generování zdrojů pomocí ray tracingu je implementováno v metodě `hybrid_method_generate()`. Podobně jako u ray tracingu jsou i v hybridní metodě vysílány paprsky ze zdroje, s tím rozdílem, že uvažujeme pouze spekulární odrazy. Na cestě každého paprsku se navíc ukládají indexy ploch, od kterých se paprsek odrazil.

Aby se zaručilo, že každá cesta (spekulárního paprsku) mezi zdrojem zvuku a posluchačem bude nalezena nejvýše jednou, je nutné uložit každý zdroj do stromu obrazů zvuku (viz 2.5). Pro získání odezvy z vygenerovaných zdrojů lze použít stejný postup jako u obrazové metody.

3.5.1 Strom zdrojů zvuku

Strom obrazů zdroje zvuku je implementován v modulu `image_tree.h`. Tento modul obsahuje třídy `Image_tree`, která reprezentuje strom jako celek, a `Tree_node`, která popisuje chování uzlu stromu. Ve stromu nejsou uloženy zdroje zvuku přímo, ale pouze jako předpis pro jejich vytvoření z původního zdroje. To je z důvodu kompatibility s metodou `calculate_energy_response_img()` pro generování impulzní odezvy. U každého uzlu je dále uloženo, jestli jde o nalezený zdroj zvuku, nebo pouze o pomocný zdroj, podle kterého je zrcadlen nějaký jiný viditelný zdroj.

Třída `Image_tree` obsahuje kořenový uzel stromu a tři základní metody. První z nich je vložení nového zdroje `add_source()`. Zdroj je definován vektorem indexů ploch podle kterých byl zdroj zrcadlen, kde první index značí zrcadlení prvního řádu. Samotný algoritmus vkládání vypadá následovně:

Algoritmus 4: Vkládání zdroje do stromu

Vstup : Uspořádaná množina ploch O, Ukazatel u na kořenový uzel stromu

```
foreach o ∈ O do
    if u má potomka zrcadleného podle o then
        Posuň u na synovský uzel u zrcadlený podle o;
    else
        Vytvoř v u synovský uzel zrcadlený podle o;
        Posuň u na synovský uzel u zrcadlený podle o;
        Nastav zdroj na u jako neviditelný;
Nastav zdroj na u jako viditelný;
```

Další důležitou metodou pro práci se stromem je `merge_trees()`. Tato metoda slouží ke sloučení dvou stromů do jednoho. Toho lze využít například při paralelním běhu hybridní

metody, kdy každé vlákno může pracovat ve svém stromu a následně se výsledky všech vláken mohou sloučit do jednoho stromu, který bude použit k dalšímu výpočtu.

Algoritmus 5: Sloučení stromů

Vstup : Ukazatel A na cílový uzel, Ukazatel B na uzel druhého stromu

1 **if** *B je viditelný* **then**
2 Nastav A jako viditelný;
3 **foreach** *s, který je synovským uzlem B* **do**
4 Nastav t jako ukazatel na synovský uzel A odražený podle stejné plochy jako s, pokud takový není, vytvoř nový neviditelný synovský uzel A odražený podle této plochy;
5 Volej rekurzivně algoritmus slučování na uzly t a s;

K převodu stromu do pole obrazů zvuku slouží metoda `linearize()`, která má jako parametry místnost obsahující odrazové plochy, které byly použity k vytvoření stromu a pozici původního zdroje. Metoda prochází strom obrazů do šířky s tím, že kromě fronty uzlů stromu vzniká současně fronta zdrojů zvuku, kam se generují obrazy zdrojů odražené podle ploch zadané místnosti.

3.6 Paralelizace simulace

Všechny tři použité metody jsou vhodné k paralelizaci, protože nejnáročnější části těchto metod lze provádět nezávisle na sobě. V případě ray tracingu je paralelizace nejsnazší. Ray tracing stačí spustit paralelně na libovolném počtu dostupných vláken a získané impulzní odezvy ukládat do odděleného místa v paměti pro každé vlákno. Na konci výpočtu lze částečné odezvy jednoduše sečíst a získáme finální výsledek.

Nepatrně složitější situace nastává u obrazové metody. Generování obrazů zdroje zvuku je dostatečně efektivní i na jednom vlákně. Problémem je následné ověření platnosti zdrojů zvuku, které vyžaduje vysílání paprsků. Pole zdrojů zvuku je proto dobré rozdělit mezi několik vláken. Protože zdroje zvuku jsou generovány algoritmem průchodu do šířky, znamená to, že na začátku pole jsou zdroje nižších řádů než na konci pole. Ověření platnosti zdroje nižšího řádu trvá kratší dobu než ověření zdroje vysokého řádu kvůli tomu, že je nutné vypočítat více odrazů paprsku pro ověření zdroje. Aby se práce rozdělila rovnoměrně mezi všechna vlákna, začíná každé vlákno na indexu, který je shodný s indexem vlákna (např. vlákno 0 začíná na indexu 0, vlákno 2 na indexu 2, atd.) a posunuje se po počtu vláken. Pokud je například použito 8 vláken, tak vlákno 3 zpracuje zdroje $3, 11, 19, 27, \ldots$. Tímto způsobem se obrazy různých řádů rozdělí rovnoměrně mezi všechna vlákna.

U hybridní metody lze paralelizovat hledání obrazů zdroje zvuku pomocí ray tracingu. Podobně jako u samotného ray tracingu lze vyhledávání zdrojů spustit paralelně nezávisle na sobě a generovat stromy obrazů zvuku pro každé vlákno zvlášť. Nakonec lze stromy obrazů sloučit pomocí metody popsané v 3.5.1. Dodatečné ověření viditelnosti vygenerovaných zdrojů lze provádět opět na jednom vlákně, protože ray tracing generuje výrazně méně zdrojů zvuku oproti původní obrazové metodě.

3.7 Auralizace

Auralizace lze dosáhnout pomocí konvoluce impulzní odezvy získané ze simulace s libovolným jiným zvukem. Pro načtení zvuku ke konvoluci byla použita knihovna *sndfile*[1]. Knihovna *sndfile* podporuje množství zvukových formátů, jako například *WAV* a *FLAC*, a je volně dostupná pro linuxové distribuce. Knihovna sndfile také poskytuje přímý přístup ke vzorkům načteného zvuku ve formě hodnot *double*.

Samotná konvoluce je implementována v modulu `conv.h` podle vzorce (2.3). Při implementaci bylo využito faktu, že generované impulzní odezvy obsahují množství tichých úseků, u kterých není nutné provádět násobení, protože se nijak nepodílí na výsledku. Přesto má základní algoritmus problémy s výkonem, protože má kvadratickou složitost. Proto byla vytvořena také efektivnější implementace algoritmu.

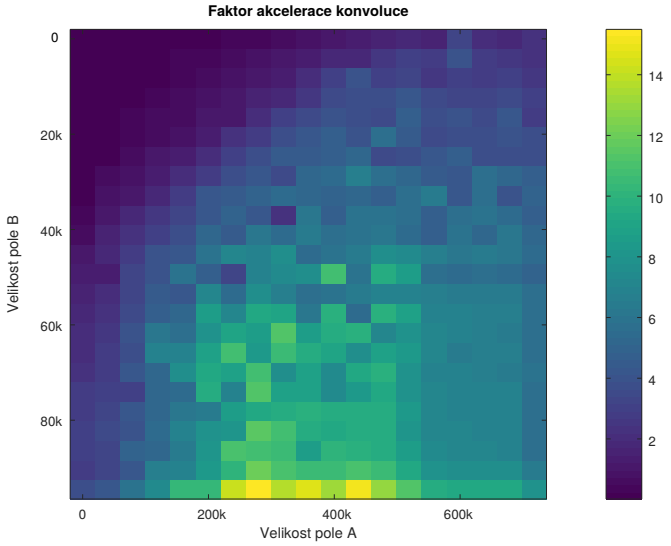
Prvním vylepšením funkce pro konvoluci je paralelní zpracování konvoluce. První signál je rozdělen mezi jednotlivá vlákna a každé vlákno potom vypočte jednu částečnou konvoluci s druhým signálem. Při sloučení částečných konvolucí jsou jednotlivé části posunuty o odpovídající indexy, kde jednotlivá vlákna začínají.

Dalším zásadním vylepšením v této funkci je použití rozšíření *AVX* (Advanced Vector Extensions) procesorů architektury *x86*. Rozšíření *AVX* umožňuje zavedení datového paralelizmu podle modelu *SIMD* (Single Instruction Multiple Data). Pomocí instrukcí *AVX* je možné zpracovávat 4 hodnoty *double* nebo 8 hodnot typu *float* v rámci jedné instrukce. Takové zpracování může několikanásobně urychlit běh programu. *AVX* poskytuje 8 registrů (*XMM0–XMM7*) v 32-bitové architektuře a 16 registrů (*YMM0–YMM15*) v případě 64-bitové architektury. Konvoluce pomocí instrukcí *AVX* se nachází v souboru `code.asm`. Při spuštění konvoluce se zjistí, jestli procesor podporuje rozšíření *AVX*. Pokud *AVX* není podporováno, je využita běžná implementace konvoluce na *FPU* (Floating point unit). Není tak nutné překládat program zvlášť pro procesory bez *AVX*, které tímto způsobem akorát nedosáhnou takového výkonu. Pokud je *AVX* podporováno, je spuštěna konvoluce pomocí *AVX*.

Konvoluce pomocí *AVX* obsahuje vnější a vnitřní smyčku. Vnější smyčka prochází každý vzorek prvního signálu a ukládá ho do registru *YMM0* pomocí instrukce *VBROADCASTSD*, která vloží do všech čtyř hodnot double v registru hodnotu vzorku. Pomocí instrukce *VCMPPD* je tento vzorek porovnán s hondotou 0 a pokud je nulový, pokračuje se s dalším vzorkem. Ve vnitřním cyklu funkce je provedeno násobení s druhým signálem. Pro načtení části druhého signálu do registru *YMM1* je použita instrukce *VMOVAPD*, která načte 4 hodnoty double ze zarovnané paměti. Tato instrukce vyžaduje, aby byl druhý signál zarovnaný v paměti na 32 bajtů. To je možné, protože vnitřní cyklus se pohybuje v poli po čtyřech hodnotách double, které mají 8 bajtů. Dále je potřeba načíst aktuální obsah výsledného pole pro přičtení mezivýsledku. To lze udělat pomocí instrukce *VMOVUPD*, která umí načítat i z nezarovnané paměti, za cenu časového zpoždění. V tomto případě se nelze vyhnout nezarovnanému čtení z paměti, protože index do výsledného pole je ovlivněn indexem do prvního pole, který se posouvá o jedničku. Dále je možné provést samotné násobení pomocí instrukce *VMULPD* a přičtení k výsledku *VADDPD*. Výsledek je opět uložen pomocí instrukce *VMOVUPD* a vnitřní smyčka se může posunout o 4 hodnoty double dál.

Jak je vidět na Obrázku 3.4, pro velmi malé pole způsobuje akcelerovaná verze algoritmu i několikanásobné zpomalení výpočtu. To je způsobeno režií spuštění vláken a přesunu do zarovnané paměti. Čím delší jsou vstupní pole, tím větší zrychlení můžeme pozorovat. Pro pole v řádech 10 tisíc až 100 tisíc prvků se konvoluce zrychlí 6 až 12 krát. Z toho vyplývá,

[1] Knihovna *sndfile*: `http://www.mega-nerd.com/libsndfile/`



Obrázek 3.4: Urychlení konvoluce v závislosti na délce vstupních polí. Na barevné škále je vidět faktor zrychlení, kde faktor 1 znamená žádné zrychlení, faktor 2 dvojnásobné zrychlení, atd.

že pro menší pole je vhodnější použít jednoduchý algoritmus konvoluce, který není zatížen takovou režií, a pro delší pole má smysl použít akcelerovaný algoritmus. Vzhledem k tomu, že ve výsledné aplikaci lze očekávat zvukové soubory, které mají i statisíce vzorků, je použita pouze akcelerovaná verze algoritmu.

Kapitola 4

Uživatelské rozhraní

V této sekci bude popsána implementace a použití uživatelského rozhraní aplikace. K aplikaci byly vytvořeny dvě uživatelské rozhraní. První uživatelské rozhraní umožňuje spouštění simulace z příkazové řádky a je čistě textové. Druhé rozhraní umožňuje spouštět simulaci z okenní aplikace.

4.1 Textové uživatelské rozhraní

Program textového rozhraní se nachází v souboru `main.cpp` v adresáři `implementace`. Textové rozhraní bylo vytvořeno aby umožnilo dávkové zpracování místností pomocí skriptů příkazové řádky. Protože backend aplikace nemá jiné závislosti než *GLM*, což je knihovna tvořená pouze hlavičkovými soubory, lze textovou verzi přeložit na libovolné platformě. Příklad spuštění textového rozhraní je na Obrázku 4.1.

```
[implementace]: ./simulator objects/mistnost2.obj out.wav -S 0.1 0.3 0.4 -L -0.9 -0.8 -0.1

STARTING SIMULATION:

room:            objects/mistnost2.obj
threads:         8

freq:            48000 Hz
low pass filter: yes

rays:            1000000
bounces:         8
sound speed      340 m/s
method:          Hybrid
diffuse model:   none

source:          0.1 0.3 0.4
listener:        -0.9 -0.8 -0.1

Materials:
        None: Refl=0.8 Rough=0.8

100 %
Result saved to: out.wav
[implementace]:
```

Obrázek 4.1: Příklad spuštění simulace z příkazové řádky v místnosti `mistnost2.obj`. Výsledná odezva je uložena do `out.wav`. Dále jsou zadány pozice posluchače (*-L*) a zdroje zvuku (*-S*). Program doplňuje nezadané parametry na výchozí hodnoty a zobrazí nastavení simulátoru před provedením simulace.

4.2 Grafické uživatelské rozhraní

Pro grafické uživatelské rozhraní byl použit framework *Qt*[1]. Framework *Qt* je multiplatformní, má podporu vykreslování pomocí *OpenGL*, což lze využít pro vykreslení místností a je dostupný volně pod licencí *(L)GPL v3*. Navíc poskytuje vývojové prostředí, které obsahuje grafický editor rozhraní.

Cílem vytvoření grafického uživatelského rozhraní bylo poskytnout uživateli lepší přehled o tvaru a vlastnostech místnosti a pozici posluchače a zdroje zvuku vzhledem k ní. To je užitečné zejména pokud grafický editor exportuje objekty převrácené podle některé osy (např. $(y, -x, z)$). Další motivací vzniku GUI bylo dát uživateli zpětnou vazbu o průběhu simulace a nějakým způsobem znázornit cesty šíření zvuku v místnosti a výsledky simulace.

4.2.1 Rozložení GUI

Uživatelské rozhraní je rozděleno na tři pohyblivé ovládací panely (V Obrázku 4.2 označeny čísly 1,2,3) a centrální prvek, který obsahuje dvě záložky: zobrazení místnosti a grafy impulzních odezev. Přeskládané ovládací panely jsou vidět na Obrázku 4.3.

Panel *Control* umožňuje nastavení vlastností místnosti jako je rychlost zvuku v místnosti, pozice reproduktoru a mikrofonu, a také vlastnosti materiálů v místnosti (odrazivost a hrubost povrchu). Seznam materiálů je zobrazen pomocí editovatelné tabulky, kam lze zapisovat hodnoty odrazivosti a hrubosti povrchů. Každý materiál je reprezentován jinou barvou, která odpovídá barvě povrchu v pohledu do místnosti. Tento panel se také stará o načítání místností a nastavení parametrů výstupní odezvy (vzorkovací frekvence a použití filtru).

Panel *Simulation* umožňuje nastavit parametry simulace. Tento panel umožňuje zvolit metodu simulace a relevantní parametry k dané metodě, jako je počet odrazů zvuku, limit paprsků, velikost příjemce a difuzní model.

Panel *Auralization* umožňuje po dokončení simulace načíst zvuk a provést konvoluci s impulzní odezvou místnosti. Výsledek je možné opět uložit do souboru.

Centrální prvek obsahuje dvě záložky. První (*3d viewport*) zobrazuje geometrii místnosti a pozice zdroje a posluchače, druhá záložka (*Impulse response*) vykresluje grafy impulzních odezev vypočtených během simulace.

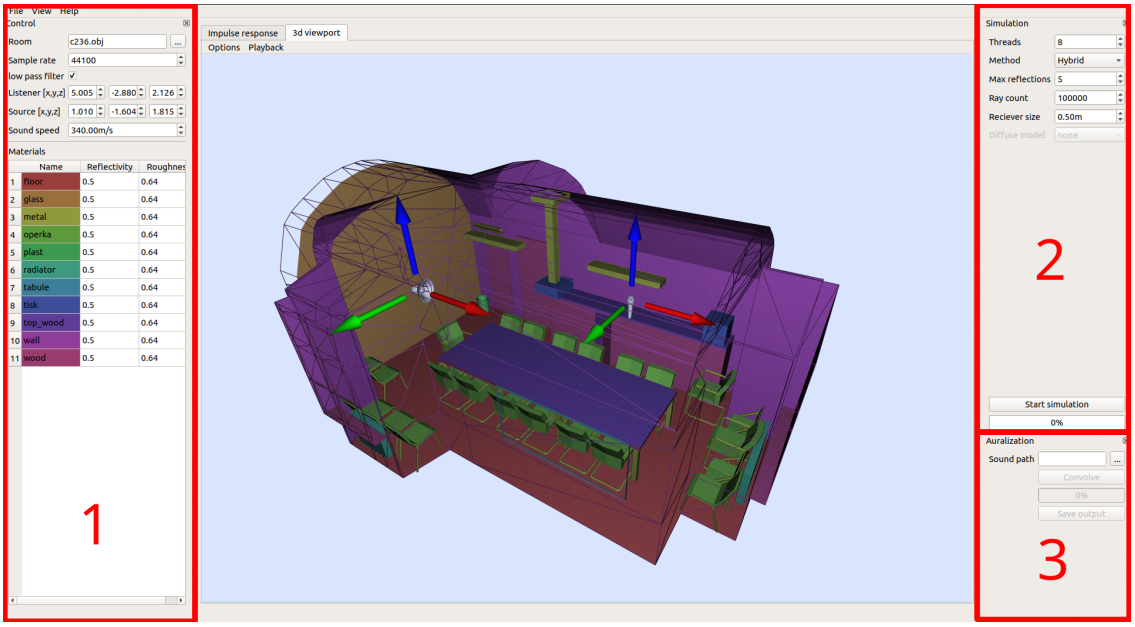
4.2.2 Vykreslení impulzní odezvy

Impulzní odezvy získané během simulace jsou zobrazovány pomocí externí knihovny *QCustomPlot*. Knihovna poskytuje třídu *QCustomPlot*, která dědí z třídy *QWidget*, a umožňuje tak použít tento element přímo v rámci *Qt*. Knihovna využívá pro vykreslování grafů *QPainter*, což je knihovna *Qt* umožňující vykreslovat objekty přímo do prostoru prvku rozhraní. Knihovna *QCustomPlot* také umožňuje manipulovat s grafem místnosti pomocí myši. Impulzní odezva vykreslená pomocí knihovny *QCustomPlot* je vidět na Obrázku 4.4.

4.2.3 Zobrazení místnosti

Pro zobrazení místnosti byl použit *QOpenGLWidget*, což je element *Qt*, který poskytuje plátno pro kreslení pomocí funkcí *OpenGL*. Pro samotné vykreslení bylo použito *OpenGL*

[1] Framework *Qt*: https://www.qt.io/



Obrázek 4.2: Finální GUI simulátoru. Na levém panelu (1) se nachází nastavení podmínek pro simulaci jako rychlost zvuku, parametry materiálů a pozice příjemce a zdroje. Pravý horní panel (2) ovládá parametry simulace a umožňuje spustit simulaci. Panel vpravo dole (3) dokáže provést auralizaci. Centrálním prvkem gui je interaktivní pohled do místnosti.

3.3 a to kvůli podpoře instancovaného vykreslování[2]. Vykreslování je implementováno ve třídě *viewport* v souboru `viewport.h`.

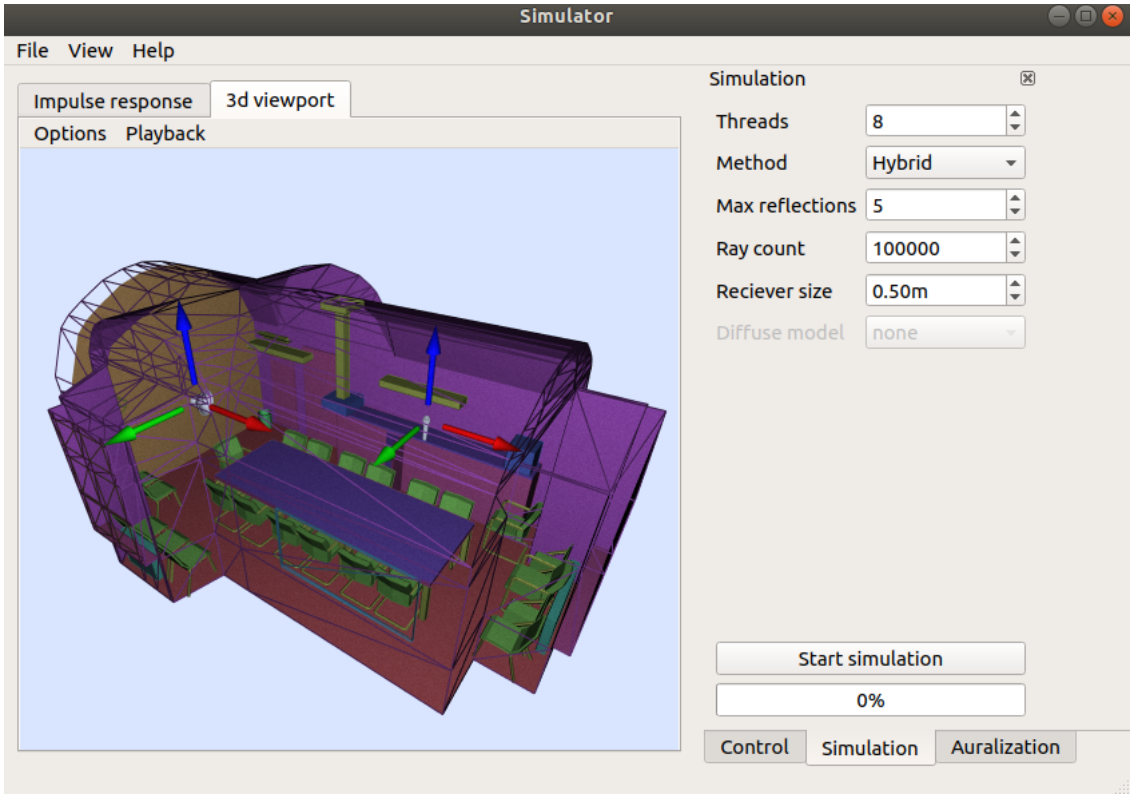
Třída *viewport* uchovává veškeré informace nutné pro vykreslení scény jako je pozice a směr pohledu kamery (azimut a elevace), pozice zdroje zvuku a posluchače a geometrie místnosti. Tato třída se také stará o veškerou interakci s uživatelem. Zachytává události stisku klávesy, tlačítek myši, pohyb myši a hodinový signál pro překreslení scény. Třída *viewport* umožňuje vykreslit tři druhy objektů: objekty složené z trojúhelníků, cesty tvořené segmenty přímek a částicové systémy.

Pohled do místnosti se překresluje každých 24 ms. Před každým vykreslením je vypočtena perspektivní a pohledová matice. Perspektivní matice je použita pro vykreslení objektů s perspektivní projekcí, kde vzdálenější objekty se jeví menší než bližší objekty. Pohledová matice slouží k posunu a rotaci vykreslovaných objektů tak, aby se vykreslily korektně vzhledem k rotaci a pozici kamery. Každý z vykreslovaných objektů může dále použít translační, rotační a zvětšovací matici, které umožní měnit jejich pozici, rotaci a velikost bez nutnosti měnit vlastnosti jejich geometrie. Matice jsou aplikovány v následujícím pořadí:

$$M = P \cdot V \cdot O_l \cdot O_r \cdot O_s, \tag{4.1}$$

kde M je výsledná transformační matice (pro transformaci bodů objektu), P je perspektivní matice, V je pohledová matice, O_l je translační matice objektu, O_r je rotační matice objektu a O_s je zvětšovací matice objektu. Matice jsou násobeny v opačném pořadí aplikace operací, nejprve je tedy aplikováno zvětšení, potom rotace, následně posun a nakonec pohledová matice (která převede scénu do kooridnátů relativních ke kameře) a perspektivní matice (která převede scénu do koordinátů relativních k vykreslovacímu plátnu).

[2]Vykreslení více objektů, které sdílejí vlastnosti jako např. geometrii, ale liší se např. v barvě a pozici.



Obrázek 4.3: Příklad zmenšeného rozložení GUI, jednotlivé panely mohou být přeskládány aby zabíraly co nejméně prostoru.

Zobrazení objektů

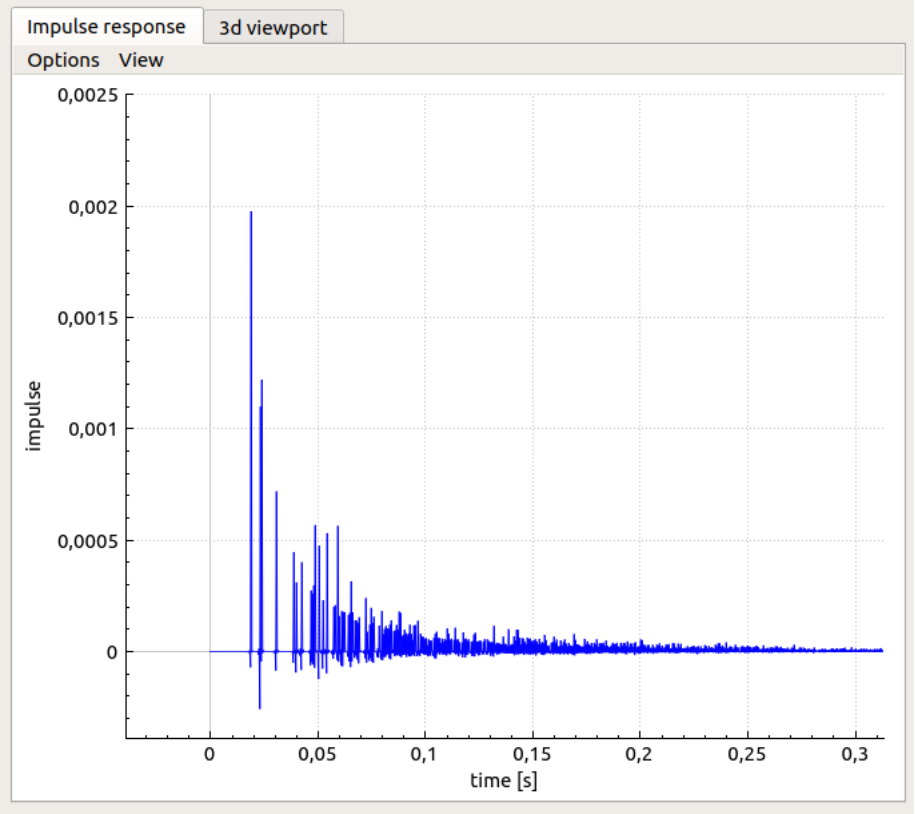
První typ objektů, které je třeba vykreslovat, jsou objekty složené z trojúhelníků. Tyto objekty jsou reprezentovány ve třídě `Mesh`. Objekty jsou zde popsány pomocí pole souřadnic bodů a normál, které po trojicích tvoří trojúhelníky. Normály, přestože je lze dopočítat ze souřadnic bodů jsou zde explicitně uvedeny pro případy, kdy je nutné hladké vykreslení povrchu objektu. V takovém případě je nutné při stínování interpolovat mezi normálami tří bodů které tvoří trojúhelník. Pokud se vykreslují plochy s ostrým stínováním, normály jsou stejné pro celý trojúhelník. Každý objekty třídy `Mesh` může použít vlastní stínovací program, který obsahuje volitelný *Vertex shader*[3] a *Fragment shader*[4].

Samotné vykreslování probíhá v jednoduchém cyklu, kde se postupně vykreslí všechny objekty ve scéně v oddělených voláních vykreslovací funkce *glDrawArrays()*. Výjimku tvoří pomocné objekty šipek pro posun posluchače a zdroje zvuku. Šipky jsou vykresleny na konec přes všechny ostatní objekty, aby bylo možné posouvat zdroj a posluchače i pokud se nacházejí za nějakou překážkou.

Zobrazení cest zvuku

Aplikace dokáže vykreslit cesty, kterými se zvuk šíří při simulaci, jak lze vidět na Obrázku 4.5. Cesty jsou reprezentovány třídou `Line`. Jednotlivé body přímky jsou uloženy v poli

[3]Program, který provádí projekci objektů a perspektivní transformaci do prostoru zobrazení.
[4]Program, který se stará o vykreslení odpovídající barvy pixelů náležící jednotlivým objektům.



Obrázek 4.4: Vykreslený graf impulzní odezvy v aplikaci.

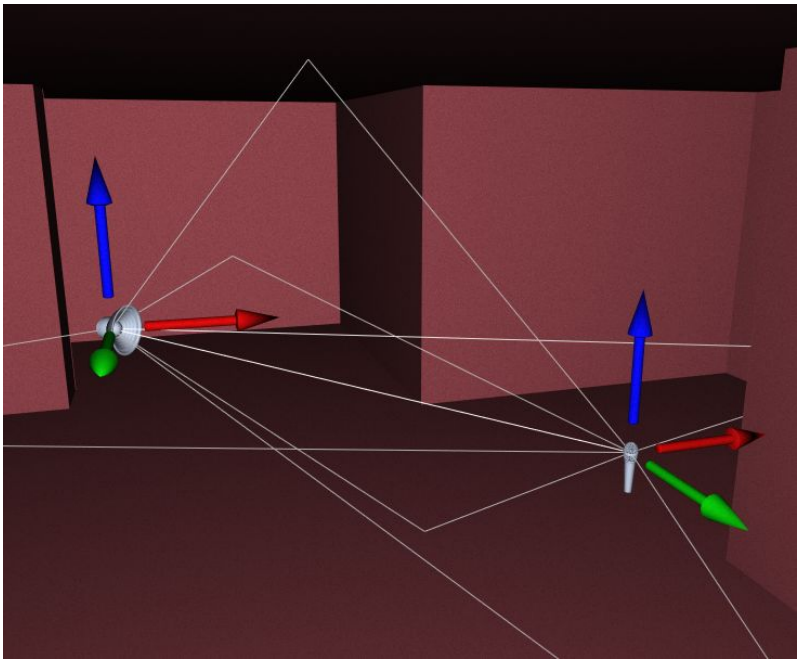
souřadnic a v poli barev. Takže každý bod má určenou pozici v prostoru a barvu. Barva cesty je použita ke znázornění řádu odrazu paprsku. Stejně jako u třídy `Mesh` lze použít libovolný stínovací program.

Vykreslování cest probíhá podobně jako u objektů typu `Mesh` s tím rozdílem, že v poli cest k vykreslení se obvykle nachází maximálně jeden objekt tohoto typu. Zde se využívá toho, že všechny cesty začínají ve zdroji a končí v příjemci. Lze proto všechny cesty sloučit do jediného objektu, kde každá lichá cesta je vykreslena směrem od zdroje k příjemci a každá sudá cesta od příjemce ke zdroji. Tímto způsobem lze ušetřit práci procesoru v případě, že scéna obsahuje velké množství cest, které mohou být vykresleny v jednom volání vykreslovací funkce.

Částicový systém

Pro účely vizualizace šíření zvukové vlny v místnosti byl vytvořen částicový systém, který je vidět na Obrázku 4.6. Částicové systémy jsou reprezentovány třídou `Particle_system`. Instance této třídy obsahuje tvar částice zapsaný jako pole koordinátů. Částicový systém dále obsahuje množinu částic třídy `Particle`. Třída `Particle` ovládá chování jednotlivých částic v systému.

Při spuštění částicového systému se vygeneruje tolik částic, kolik bylo částicovému systému předáno cest ze simulátoru. Každé částici je potom přidělena jedna cesta, po které se bude pohybovat. Protože částic v systému může být velké množství (desítky až stovky tisíc), je nutné je vykreslovat efektivně. Z toho vyplývá, že nelze použít stejný způsob vy-



Obrázek 4.5: Zobrazení cest maximálně prvního řádu odrazu, které byly nalezeny při simulaci akustiky.

kreslování jako u objektů třídy `Mesh`, kde by se musela vykreslovací funkce volat pro každou částici zvlášť [8].

Řešením tohoto problému je využití instancování částic na grafické kartě. Protože všechny částice jsou v podstatě stejné a liší se pouze v pozici a barvě, můžeme sdílet geometrii částice mezi všemi jejími instancemi. Toho můžeme docílit použitím funkce *glDrawArraysInstanced()*. Před použitím této funkce jsou vytvořeny pole s geometrií částice, pole s pozicemi částic a pole s barvami částic. Funkce *glVertexAttribDivisor()* umožňuje nastavit o kolik položek se má po vykreslení každé instance posunout v jednotlivých polích. V případě pole s geometrií částice se po vykreslení neposouvá vůbec, protože další částice vypadají stejně. V případě polí s pozicemi a barvou částic se po každém vykreslení posouvá o jeden prvek, takže při vykreslení se mění pouze barva a umístění částic a jejich geometrie je sdílená.

Dalším problémem při vykreslování částic je jejich orientace. Vzhledem k tomu, že částice jsou typicky reprezentovány jedním čtvercem (nebo jiným 2D útvarem), je nutné, aby byly při vykreslování otočeny vždy na pozorovatele. Rotace částic musí být efektivní a vzhledem k jejich počtu nelze provádět tuto transformaci na procesoru. Protože známe orientaci kamery a vektory $\overrightarrow{c_r}, \overrightarrow{c_t}$ (pravý a horní vektor směřující z kamery), je možné předat informace do stínovacího programu a tam provést transformaci podle rovnice [8]:

$$\overrightarrow{v_{pos}} = \overrightarrow{p_{pos}} + \overrightarrow{c_r} \cdot x + \overrightarrow{c_t} \cdot y, \tag{4.2}$$

kde $\overrightarrow{v_{pos}}$ je výsledná pozice bodu, $\overrightarrow{p_{pos}}$ je pozice vykreslované částice, x a y jsou souřadnice právě zpracovávaného bodu geometrie částice a $\overrightarrow{c_r}, \overrightarrow{c_t}$ jsou vektory pohledu kamery. Jakýkoliv 2D objekt zadaný pomocí souřadnic x, y se tímto způsobem zobrazí vždy otočený směrem ke kameře.

Částice také využívají texturu s průhledností. K mapování vykreslených fragmentů na pixely textury jsou opět využity *uv* koordináty, které jsou generovány automaticky z geo-

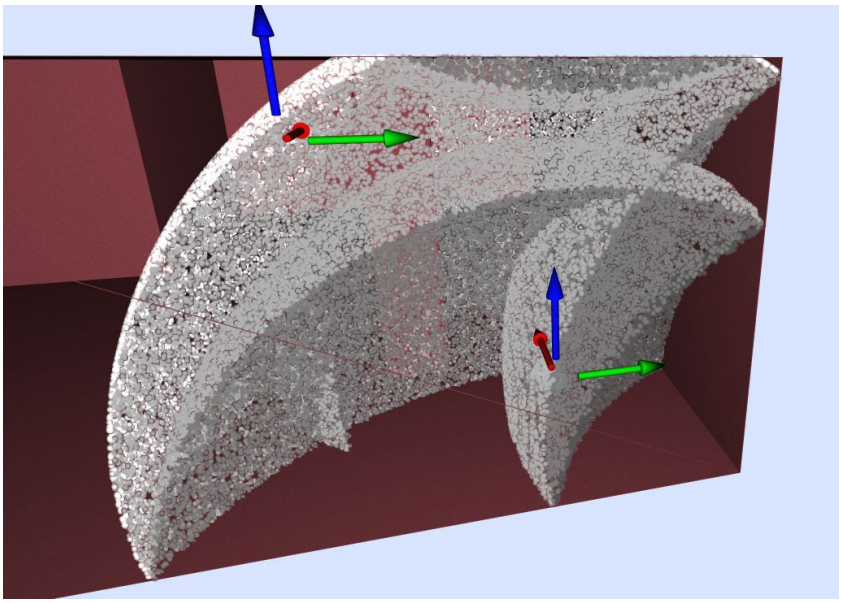
metrie částice. Protože všechny body částice se nacházejí v jedné ploše, lze normalizovat souřadnice x a y do rozsahu $\langle 0, 1\rangle$ a tyto hodnoty použít jako uv koordináty do textury.

Trajektorie jednotlivých částic je dána následující interpolací:

$$k = \frac{v_p \cdot t}{d}, \tag{4.3}$$

$$\overrightarrow{pos} = k \cdot \overrightarrow{p_i} + (1 - k)\overrightarrow{p_{i+1}}, \tag{4.4}$$

kde v_p je rychlost částice, t je čas částice strávený na segmentu cesty, d je délka segmentu cesty, p_i je i-tý bod cesty a $\overrightarrow{pos}$ je výsledná pozice částice. V případě, že $k > 1$, posune se částice na další segment cesty. Pokud je částice již na posledním segmentu, je možné ji odstranit. Při každém přesunu na další segment cesty se ztmaví barva částice, aby šlo vizuálně odlišit odrazy různých řádů.



Obrázek 4.6: Vizualizace výsledků simulace pomocí částic. Scéna obsahuje 100 000 částic. Barva částic odpovídá řádu odrazu, kde světlejší částice značí nižší řád odrazu.

Stínování objektů

O stínování vykreslených objektů se stará takzvaný *Fragment shader*. Stínovací program je proveden nad každým pixelem na obrazovce, který patří právě vykreslovanému objektu. Při zobrazení místnosti a dalších pomocných objektů není vhodné, aby všechny povrchy byly vykresleny jednotnou barvou, protože uživatel ztrácí přehled o tvaru vykreslovaného objektu.

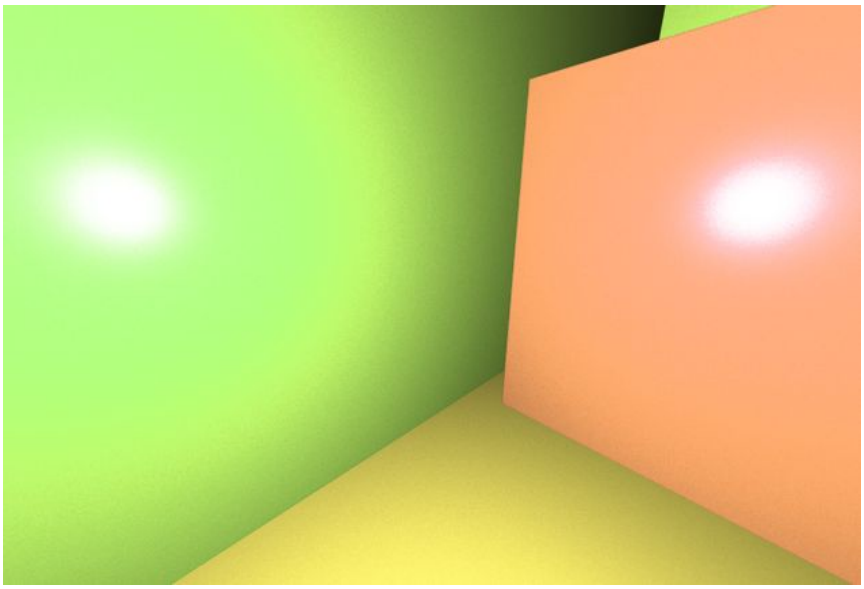
Pro vykreslení stínovaného objektu je možné použít například *Phongův* osvětlovací model [7]. *Phongův* osvětlovací model uvažuje ambientní, difuzní a spekulární osvětlení. Ambientní světlo představuje světlo prostředí a vypočte se vynásobením základové barvy povrchu barvou světla. Difuzní složka světla se řídí Lambertovým difuzním modelem [15] (viz rovnice (2.5)). V případě bodového světla které bylo použito pro osvětlení navíc intenzita klesá se čtvercem vzdálenosti světla od povrchu. Poslední složkou Phongova modelu je spekulární

světlo. Spekulární složka představuje odlesk světla a lze ji vypočítat jako [7]:

$$I_s = \frac{\left(\text{reflect}(\overrightarrow{L_d}, \overrightarrow{n}) \cdot \overrightarrow{C_{dir}}\right)^s}{d^2}, \tag{4.5}$$

kde I_s je intenzita spekulární složky, d je vzdálenost světla od povrchu, $\overrightarrow{n}$ značí normálu povrchu, $\overrightarrow{L_d}$ je směr světla a $\overrightarrow{C_{dir}}$ je směr pohledu kamery. Funkce reflect() počítá odraz paprsku podle normály plochy.

Kompletní osvětlení povrchu můžeme získat sečtením těchto tří složek. Výsledek osvětlení lze vidět na Obrázku 4.7.



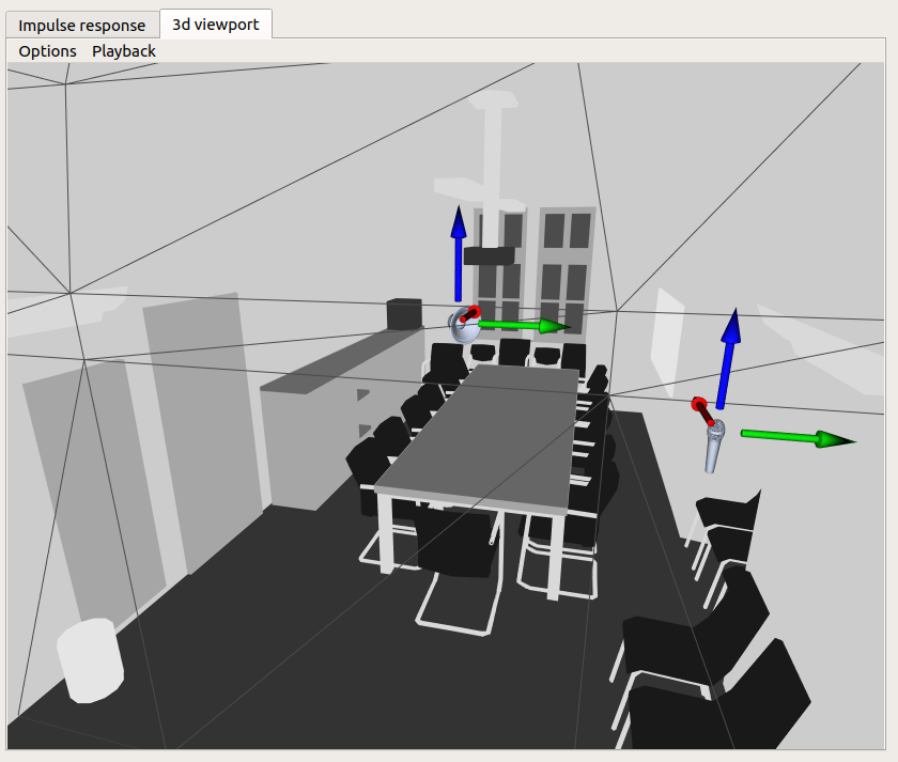
Obrázek 4.7: Osvětlení povrchu pomocí Phongova modelu. Bílé odlesky světla na pozici kamery vznikají přičtením spekulární složky.

Ovládání pohledu do místnosti

Zobrazení místnosti umožňuje pohyb po místnosti pomocí „létající“ kamery a rozhlížení v místnosti pomocí rotace kamery z první osoby (kamera rotuje kolem své vlastní pozice). Základní ovládání kamery je realizováno zachytáváním stisků kláves a modifikací pozice kamery a její rotace. Pohyb kamery je realizován pomocí kláves W,A,S,D. Klávesy mezerník a levý Ctrl slouží k pohybu nahoru a dolů. Rotaci pohledu lze provést buď pomocí směrových šipek na klávesnici, nebo přidržením středního tlačítka myši a následným pohybem myší. Zobrazení místnosti také umožňuje přesun objektů posluchače a zdroje zvuku v místnosti, a to kliknutím na příslušnou směrovou šipku a tažením myší.

Pohled do místnosti také poskytuje náhledy na vlastnosti materiálů, kde jsou místo barev materiálů vykresleny jejich vlastnosti jako je hrubost a reflektivita (lze vidět na Obrázku 4.8). Uživatel také může prohodit normály ploch místnosti při vykreslení v případě, že stěny místnosti blokují pohled do místnosti z prostoru mimo místnost, nebo vykreslit pouze drátový model.

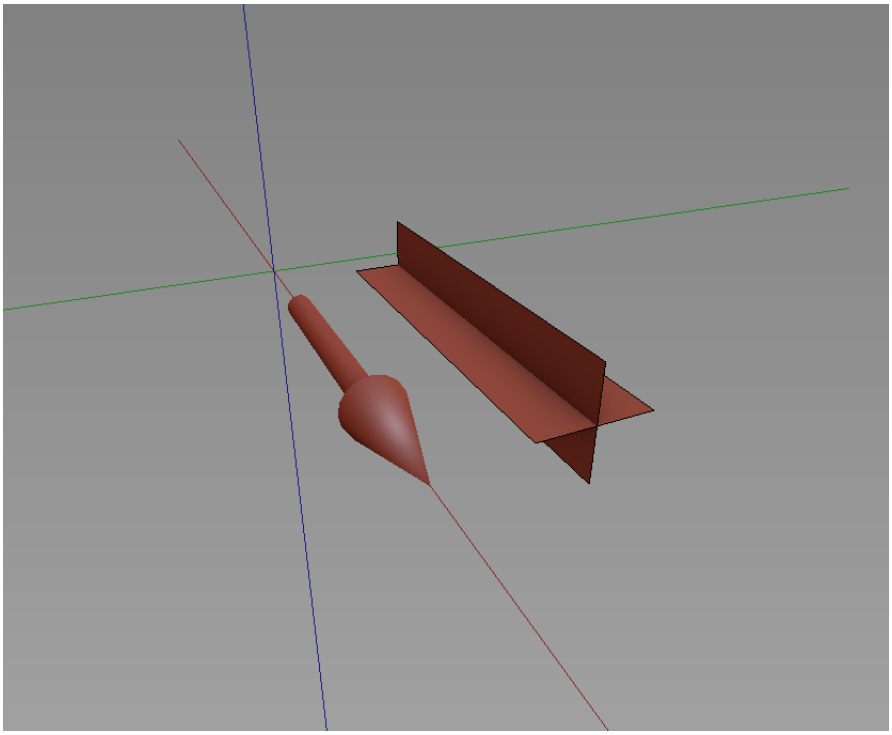
Částicový systém lze ovládat pomocí záložky *Playback*. Zde je možné přehrát částicovou simulaci, nastavit limit počtu částic, velikost částic a také jejich rychlost.



Obrázek 4.8: Ukázka náhledu na vlastnosti materiálů. Vlastnosti jsou reprezentovány pomocí stupňů šedi, kde bílá značí hodnotu 1 dané vlastnosti a černá hodnotu 0. Aplikace umožňuje zobrazit náhled na odrazivost a hrubost povrchu.

Posun objektů příjemce a zdroje zvuku přímo v pohledu do místnosti je realizován pomocí vysílání paprsků do scény a detekce nárazu paprsku do pohyblivého objektu ve scéně, v tomto případě šipky umožňující pohyb objektu po jednotlivých osách (jak je dobře vidět např. v Obrázku 4.5). Protože k zobrazení byla použita perspektivní projekce, je nutné najít způsob, kterým lze převést číslo pixelu, na který uživatel kliknul v rámci zobrazení, na směr paprsku v koordinátech světa (tedy neovlivněné perspektivní a pohledovou projekcí). To je možné udělat nalezením a postupnou aplikací inverzní pohledové a perspektivní matice [5]. K nalezení inverzní matice byla použita knihovna *GLM*. Vektor ve světových koordinátech lze potom získat aplikací inverzní projekční matice, kterou se vektor transformuje do souřadného systému relativně ke kameře, a inverzní pohledové matice, kterou se vektor transformuje do souřadného systému světa, čímž získáme výsledný směrový vektor, který je vhodné normalizovat.

Druhým problémem, který je nutné řešit, je detekce kolize paprsku s objektem ve scéně. Jediné objekty, které je nutné detekovat jsou šipky zarovnané podle osy x, y nebo z. Bod průniku můžeme najít na plochách xy, xz nebo yz posunutých na pozici objektu posluchače a příjemce a tím zjednodušit detekci kolize s šipkou, která obsahuje množství polygonů. Šipku můžeme reprezentovat jako obdélník na ploše, jak lze vidět na Obrázku 4.9. U každé šipky můžeme provést detekci průniku se dvěma plochami. Například u šipky pro posun po ose x připadají v úvahu plochy xz a xy, ze dvou ploch můžeme vybrat tu, kde přímka se směrem normály plochy a vektor pohledu kamery svírají menší úhel.



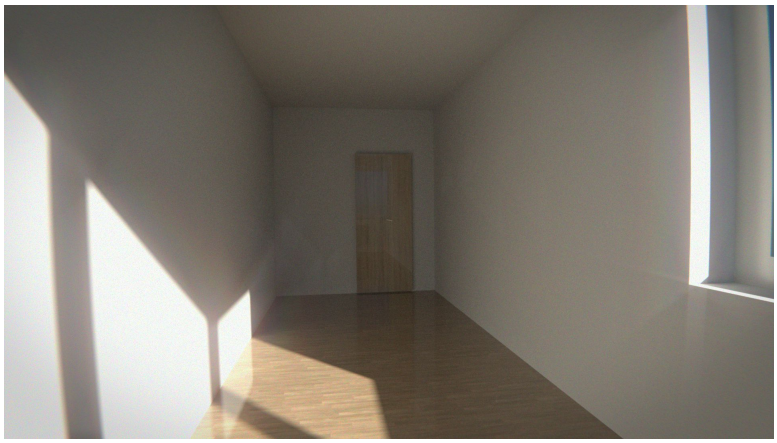
Obrázek 4.9: Model šipky a zástupný objekt pro detekci kolize (pro názornost jsou zobrazeny vedle sebe). Šipka slouží pro posun po ose x. Zástupný objekt je složen ze dvou obdélníků zarovnaných podle ploch xy a xz.

Výhoda tohoto přístupu je možnost určení nové pozice šipky po posunu myši. Při posunu myši se už objekt nemusí nacházet pod kurzorem, a je tak nutné nějakým způsobem určit, jaký posun objektu ve světových koordinátech odpovídá posunu myši na obrazovce. V případě detekce průniku s plochou, která je nekonečná získáme odpovídající posun přímo ze souřadnic průniku.

Kapitola 5

Testování

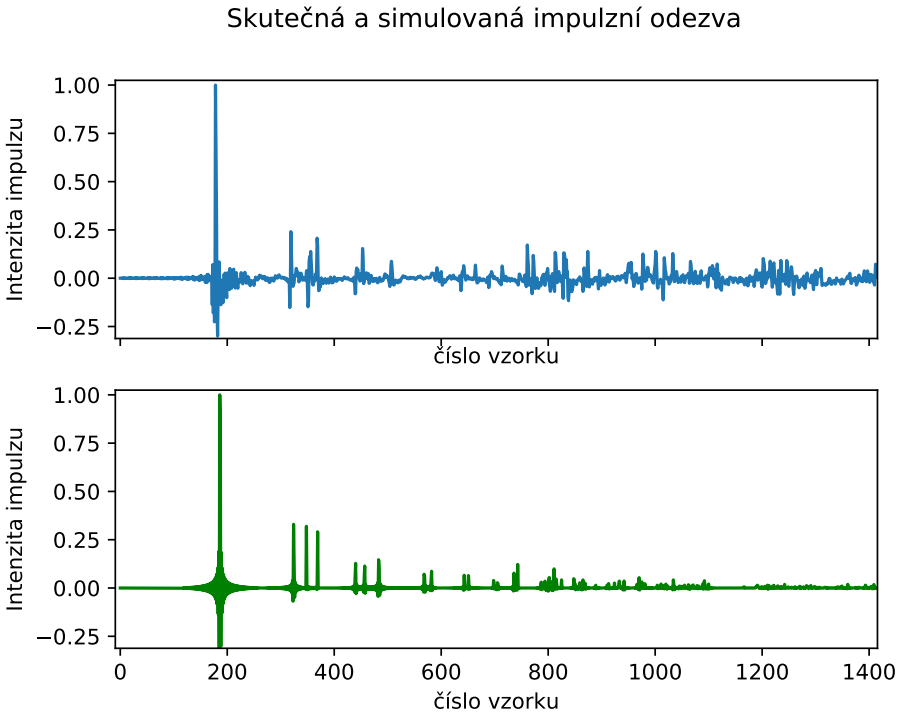
V této kapitole budou porovnány výsledky simulace s naměřenými odezvami skutečných místností. Pro testování byly naměřeny impulzní odezvy čtyř místností: místnosti *C236* (Obrázek 5.7), která se nachází na *FIT* a tří dalších místností (Obrázky 5.1,5.3 a 5.5). Měření probíhalo v místnostech malých a středních rozměrů. Modely místností byly vytvořeny a vykresleny pomocí 3D editoru *Blender*, který umí exportovat do formátu *obj*.



Obrázek 5.1: Počítačový model prázdné místnosti A, ve které probíhalo měření.

Tabulka 5.1: Porovnání metod na základě chyby při předpovídání času (e_t) a energie (e_I) impulzů v impulzní odezvě. Je uvedena střední hodnota E a směrodatná odchylka σ. U metody *Diffuse rain* není uvedena chyba z hlediska času kvůli vysokému výskytu neplatných impulzů v simulované odezvě.

	$E(e_t)$	σ	$E(e_I)$	σ
Hybridní metoda	0.104 ms	0.242 ms	0.11	0.082
Ray tracing	0.126 ms	0.133 ms	0.15	0.111
Diffuse rain	-	-	0.27	0.186

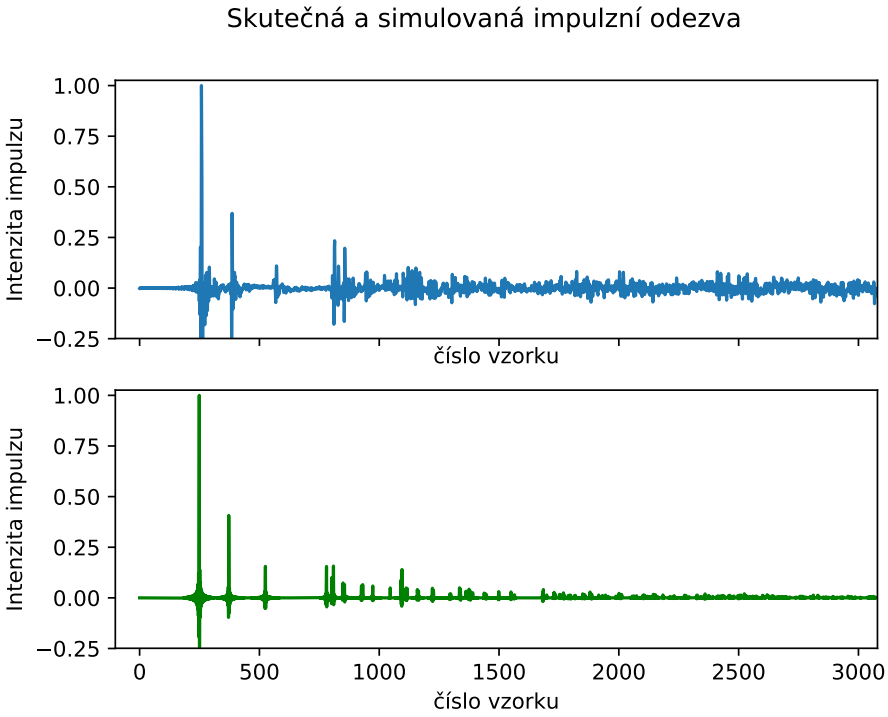


Obrázek 5.2: Graf nahoře zobrazuje naměřenou impulzní odezvu z místnosti A (viz Obrázek 5.1). Graf dole obsahuje simulovanou impulzní odezvu pomocí hybridní metody.

Obrázek 5.3: Počítačový model prázdné místnosti B, ve které probíhalo měření.

5.1 Zhodnocení použitých metod

Při testování se nejvíce osvědčila hybridní metoda, která zachovává přesnost základní obrazové metody ovšem s výrazně lepší výpočetní složitostí. Základní obrazová metoda se ukázala jako nevhodná pro hledání obrazů vyšších řádů v místnostech s poměrně jednoduchou geometrií (v řádu desítek odrazových ploch) ale i obrazů druhého nebo třetího řádu v místnostech s velmi složitou geometrií (tisíce odrazových ploch).

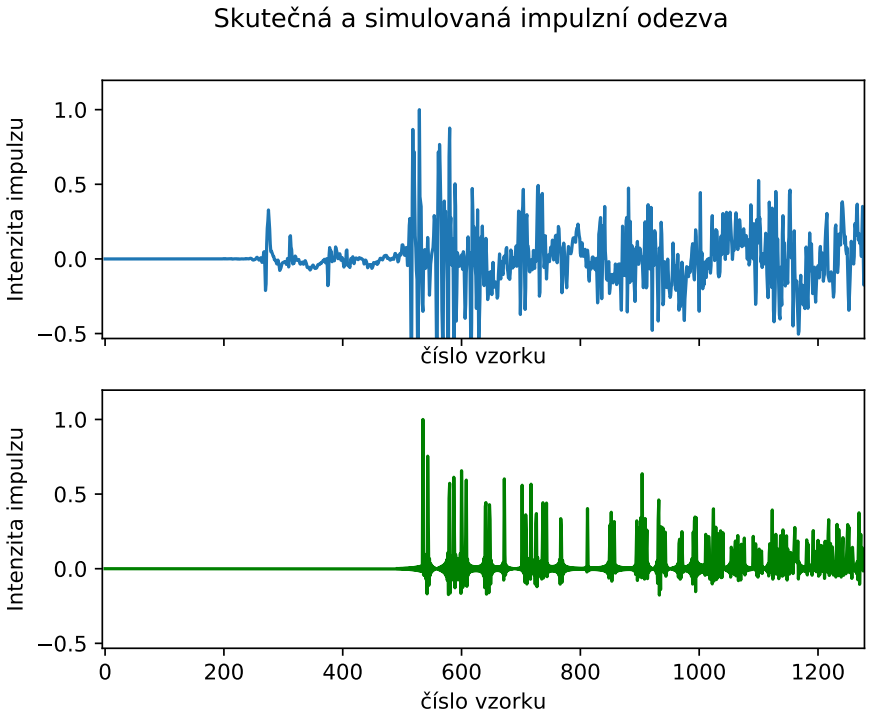


Obrázek 5.4: Graf nahoře zobrazuje naměřenou impulzní odezvu z místnosti B (viz Obrázek 5.3). Graf dole obsahuje simulovanou impulzní odezvu pomocí hybridní metody.

Obrázek 5.5: Počítačový model prázdné místnosti C, ve které probíhalo měření.

Z hlediska odhadu času příchodu impulzů (viz Tabulku 5.1) se osvědčil i ray tracing. Ray tracing ovšem vykazoval horší přesnost z hlediska energie impulzů. To může být způsobeno opakovanými zásahy zdroje zvuku ze stejných cest, které podléhají náhodnému generování paprsků.

Ray tracing s technikou *diffuse rain* se ukázal jako nejrychlejší ze všech implementovaných metod, protože k nalezení nějaké cesty mezi posluchačem a zdrojem zvuku stačí jen několik paprsků. Tato metoda se ale také ukázala jako nejméně přesná, protože generuje množství neplatných impulzů.

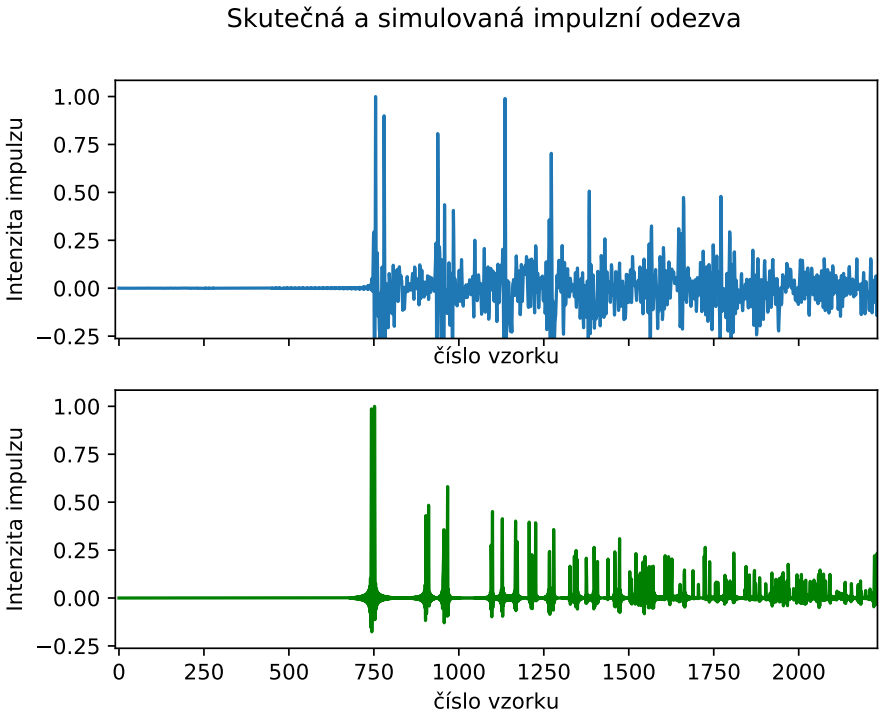


Obrázek 5.6: Graf nahoře zobrazuje naměřenou impulzní odezvu z místnosti C (viz Obrázek 5.5). Graf dole obsahuje simulovanou impulzní odezvu pomocí hybridní metody. Mikrofon je v tomto měření skryt za překážkou. Zvuk se dostane k mikrofonu dříve pomocí difrakce, než odrazem od stěny místnosti.

Obrázek 5.7: Počítačový model místnosti C236 na FIT.

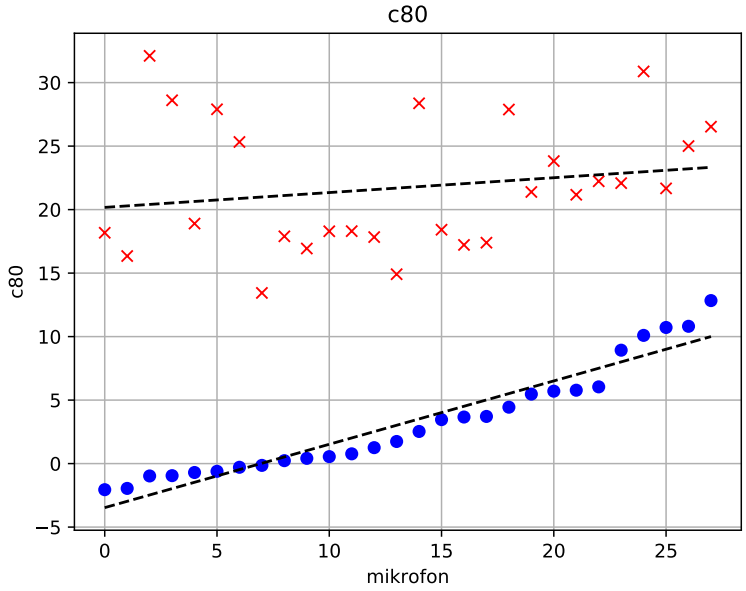
V případech kdy není zdroj viditelný z pozice posluchače, dochází k nepřesnosti u všech implementovaných metod. Zvuk vlivem difrakce doputuje k mikrofonu dříve než dojde k odrazu zvuku od stěny a do mikrofonu, jak je vidět např. na Obrázku 5.6.

Nejlepších výsledků bylo dosaženo za pomocí hybridní metody v největší prázdné místnosti B (Obrázek 5.3).

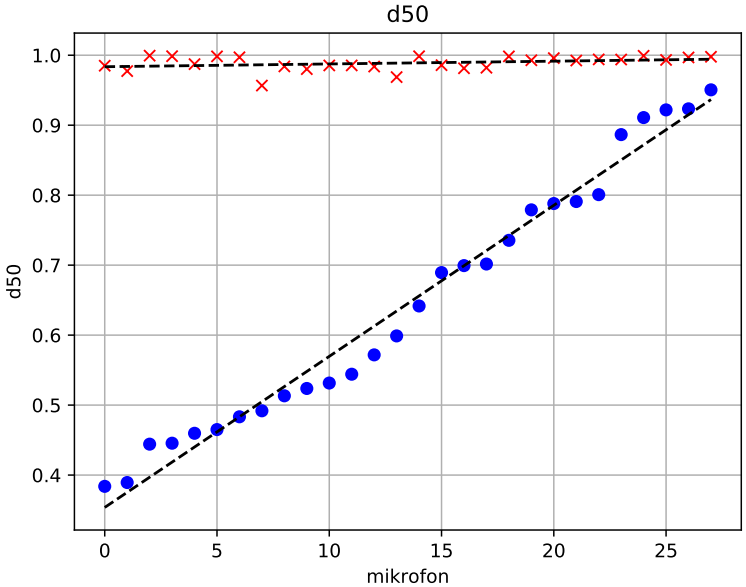


Obrázek 5.8: Graf nahoře zobrazuje naměřenou impulzní odezvu z místnosti *C236* na Obrázku 5.7.

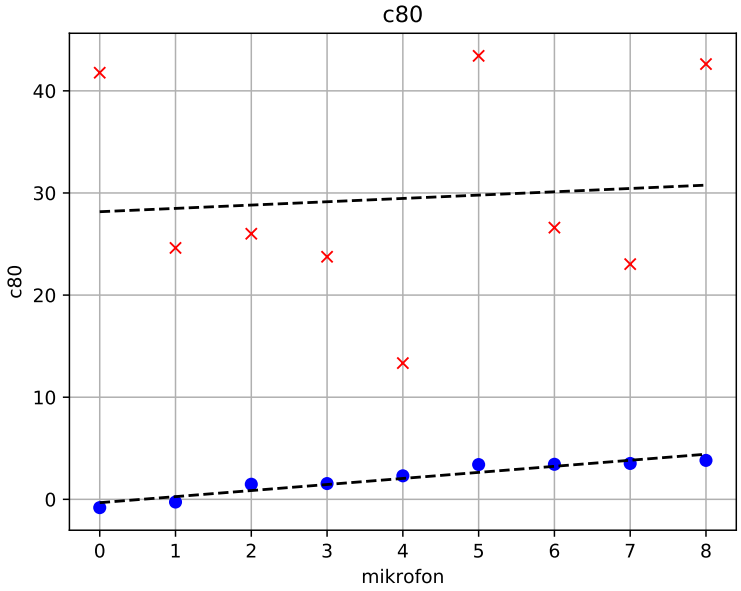
Implementované metody nejsou příliš vhodné pro odhad akustických parametrů jako C_{80} a D_{50}, které jsou založeny na porovnání energie prvních 80/50-ti ms se zbytkem odezvy (viz Obrázky 5.9, 5.10,5.11 a 5.12). To je do značné míry způsobeno tím, že energie simulované odezvy klesne velmi rychle na hodnotu nula, zatímco ve skutečné odezvě na konci zbývá šum, který stále nese nějakou energii.



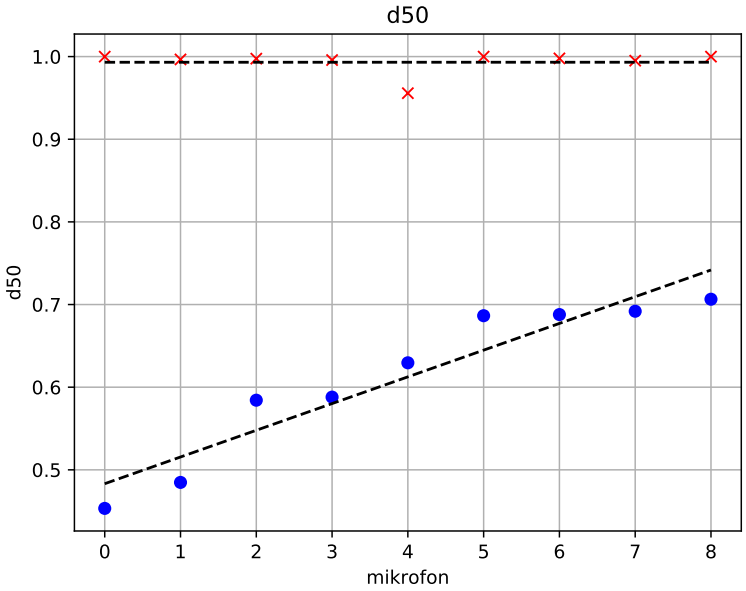
Obrázek 5.9: Porovnání naměřených (modře) a simulovaných (červeně) impulzních odezev pomocí parametru C_{80}. Hodnoty byly naměřeny v místnosti *C236* (Obrázek 5.7). Naměřené výsledky jsou seřazeny vzestupně podle parametru C_{80} a k nim jsou vyobrazeny odpovídající hodnoty ze simulace.



Obrázek 5.10: Porovnání naměřených (modře) a simulovaných (červeně) impulzních odezev pomocí parametru D_{50}. Hodnoty byly naměřeny v místnosti *C236* (Obrázek 5.7). Naměřené výsledky jsou seřazeny vzestupně podle parametru D_{50} a k nim jsou vyobrazeny odpovídající hodnoty ze simulace.



Obrázek 5.11: Porovnání naměřených (modře) a simulovaných (červeně) impulzních odezev pomocí parametru C_{80}. Výsledky ze všech místností kromě *C236*. Naměřené výsledky jsou seřazeny vzestupně podle parametru C_{80} a k nim jsou vyobrazeny odpovídající hodnoty ze simulace.



Obrázek 5.12: Porovnání naměřených (modře) a simulovaných (červeně) impulzních odezev pomocí parametru D_{50}. Výsledky ze všech místností kromě *C236* Naměřené výsledky jsou seřazeny vzestupně podle parametru D_{50} a k nim jsou vyobrazeny odpovídající hodnoty ze simulace.

Kapitola 6

Závěr

V této práci byly shrnuty metody používané pro simulaci akustiky místnosti. Byly důkladně popsány metody ray tracing a obrazová metoda, jejich výhody a nevýhody a také jejich možná kombinace. Následně byla popsána moje implementace ray tracingu a obrazové metody v jazyce *C++* s využitím knihovny *GLM* a také hybridní metody využívající jak ray tracing tak obrazovou metodu. Dále byla vytvořena aplikace s textovým i grafickým uživatelským rozhraním, která umožňuje provádět simulace. Vytvořená aplikace s grafickým uživatelským rozhraním umí navíc zobrazovat místnosti, graficky znázornit výsledky simulace, vykreslovat impulzní odezvy, provádět auralizaci a vizualizovat šíření zvukové vlny v prostoru. Grafické uživatelské rozhraní bylo vytvořeno pomocí knihovny *Qt5* a *OpenGL 3.3*.

Výsledky simulace byly porovnány s měřením impulzních odezev v několika skutečných místnostech a bylo zjištěno, že nejpřesnějších výsledků dosahuje hybridní metoda, která zachovává přesnost obrazové metody ovšem s lepší výpočetní náročností. Nejlepších výsledků bylo dosaženo ve velkých prázdných místnostech. Ukázaly se také některé problémy implementovaných metod jako například modelování ohybu (difrakce) vlny kolem překážky, nebo příliš velký útlum konce impulzní odezvy, který se projevil při měření parametrů C_{80} a D_{50}.

6.1 Možnosti pokračování práce

Při práci na aplikaci se objevilo několik oblastí, které by mohly být dále rozšířeny nebo vylepšeny. Výkon simulátoru by mohl být zlepšen zavedením nějaké akcelerační struktury jako je Oktalový, *BSP* nebo *BVH* strom. Toto zlepšení by se projevilo zvláště v místnostech se složitou geometrií.

Mohly by být implementovány a porovnány další metody simulace jako například *FEM* nebo *BEM*, které by mohly simulovat některé efekty šíření vlny v prostoru jako například difrakce. Zajímavé by mohlo být také využití ray tracingu s modelem „diffuse rain“ pro simulaci pohyblivého zdroje a příjemce v reálném čase například v rámci nějakého grafického/herního „enginu“.

Přesnost simulace by mohla být zvýšena měřením vlastností zdroje zvuku a mikrofonu jako například direktivita a následná aplikace těchto dat v modelu šíření zvuku.

Další možností vylepšení procesu auralizace by bylo takzvané binaurální zpracování, které by umožnilo simulovat zvukový vjem posluchače v závislosti na natočení hlavy posluchače v místnosti a lomu zvuku kolem hlavy posluchače.

Také by bylo možné zavést podporu pro více souborových formátů pro popis geometrie jako například formát *Autodesk FBX Technology*, což je další široce používaný formát.

Dalším možným rozšířením by bylo vytvoření nástroje pro analýzu získaných impulzních odezev přímo v aplikaci.

Literatura

[1] Measuring impusle response using Dirac, An Introduction. [Online; přístup 10.5.2019]. URL https://www.acoustics-engineering.com/files/TN001.pdf

[2] Belk, J.: *How to find a random axis or unit vector in 3D? 2011.* *URL https://math.stackexchange.com/q/44691*

[3] Bourke, P.: Object Files (.obj). [Online; přístup 10.5.2019]. *URL http://paulbourke.net/dataformats/obj/*

[4] Elorza, D. O.: Room acoustics modeling using the ray-tracing method: implementation and evaluation. 2005.

[5] Gerdelan, A.: Mouse Picking with Ray Casting. 2016, [Online; přístup 8.5.2019]. *URL http://antongerdelan.net/opengl/raycasting.html*

[6] Habets, E. A. P.: Room Impulse Response Generator. September 2010.

[7] Hocevar, S.: Basic shading. 2004, [Online; přístup 6.5.2019]. *URL http://www.opengl-tutorial.org/beginners-tutorials/tutorial-8-basic-shading/*

[8] Hocevar, S.: Particles / Instancing. 2004, [Online; přístup 1.5.2019]. *URL http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/*

[9] Kuttruf, H.: Room acoustics. London & New York: Spon Press/Taylor & Francis, páté vydání, 2009, ISBN 9780415480215.

[10] Min-Zhi Shao, N. I. B.: Spherical Sampling by Archimedes' Theorem. 1996.

[11] Rindel, J. H.: Computer Simulation Techniques for Acoustical Design of Rooms. September 1995.

[12] Savioja, L. . S.: Overview of geometrical room acoustic modeling techniques. The Journal of the Acoustical Society of America. 138. 708-730. 10.1121/1.4926438, *2015.*

[13] Trumbore, T. M. . B.: Fast, Minimum Storage Ray/Triangle Intersection.

[14] Wikipedia contributors: Dummy head recording — Wikipedia, The Free Encyclopedia. 2019, [Online; přístup 11.5.2019]. *URL https://en.wikipedia.org/w/index.php?title=Dummy_head_recording&oldid=895581350*

[15] *Wikipedia contributors: Lambertian reflectance — Wikipedia, The Free Encyclopedia. 2019, [Online; přístup 1.5.2019].*
URL https://en.wikipedia.org/w/index.php?title=Lambertian_reflectance&oldid=886405253

[16] *Wikipedia contributors: Rotation matrix — Wikipedia, The Free Encyclopedia. 2019, [Online; přístup 1.5.2019].*
URL https://en.wikipedia.org/w/index.php?title=Rotation_matrix&oldid=891554293

[17] *Wikipedia contributors: Wave equation — Wikipedia, The Free Encyclopedia. 2019, [Online; přístup 1.5.2019].*
URL https://en.wikipedia.org/w/index.php?title=Wave_equation&oldid=891912306

Příloha A

Obsah přiloženého paměťového média

```
DVD/
├── doc/
│   └── html/
│       └── Programová dokumentace v HTML
├── aplikace/
│   └── Zdrojové soubory aplikace v Qt
├── implementace/
│   ├── glm/
│   │   └── Hlavičkové soubory knihovny GLM (OpenGL mathematics)
│   ├── objects/
│   │   └── Ukázkové modely místností
│   ├── Zdrojové soubory simulátoru
│   └── Makefile
├── konvoluce/
│   └── Zdrojové soubory pro konvoluci
├── sablona/
│   └── Zdrojové soubory tohoto dokumentu
├── Makefile
├── Doxyfile
├── README.md
└── plakat.pdf
```

Příloha B

Manuál

B.1 Použití na příkazové řádce

Aplikaci pro použití na příkazové řádce je možné přeložit pomocí **make** ve složce **implementace**, který vytvoří spustitelný soubor **simulator**. Ukázkové spuštění lze provést voláním **make run**.

Tento program má jeden povinný argument a tím je cesta k souboru s místností **.obj*. Dále je možné zadat název výstupního souboru **.wav* a materiálovou knihovnu **.mtl*.

Další užitečné argumenty jsou *-r* pro nastavení počtu paprsků při simulaci, *-b* pro nastavení maximálního počtu odrazů, *-f* pro nastavení vzorkovací frekvence, *-s* pro nastavení rychlosti zvuku a *-S* a *-L* pro nastavení pozice příjemce a zdroje. Seznam zbylých argumentů a přepínačů, včetně výchozích hodnot nastavení simulace je uveden v nápovědě programu, kterou lze zobrazit přepínačem *-h* nebo *–help*.

B.2 Spuštění aplikace s grafickým rozhraním

Aplikaci s grafickým rozhraním lze přeložit a spustit pomocí **make** v kořenovém adresáři. Spustitelný soubor s názvem **bp** lze potom najít v adresáři **aplikace**. Spuštěním **make run** v kořenovém adresáři lze aplikaci přeložit a spustit přímo.

Tato aplikace má následující závislosti:

- Qt5
- Sndfile
- nasm
- OpenGL 3.3
- freeglut 3
- glu 1

Tyto závislosti lze nainstalovat cílem **make dependency**, na platformách se správcem balíků *apt*. Vyžaduje administrátorské oprávnění.

B.3 Ovládání aplikace s grafickým rozhraním

Spuštěná aplikace je zobrazena na obrázku B.1. Místnost lze otevřít pomocí nabídky *File* → *Open*, nebo kliknutím na tlačítko *[...]* v panelu *Control*. V panelu *Control* lze také nastavovat další parametry simulace a upravovat vlastnosti materiálů místnosti přímo psaním do tabulky hodnot odrazivosti a hrubosti povrchu.

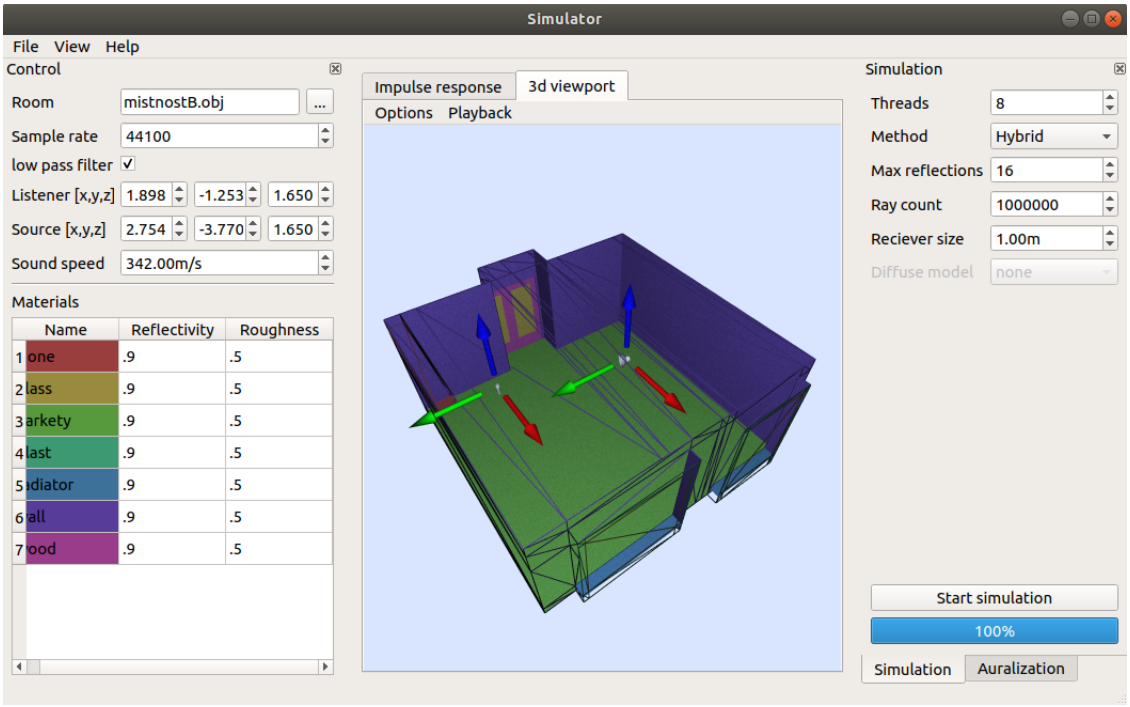
V panelu *Simulation* lze spustit simulaci. Panel umožňuje vybrat zvolenou metodu pro simulaci a nastavit dodatečné parametry relevantní pro danou metodu.

Po dokončení simulace lze zobrazit impulzní odezvu v záložce *Impulse response* v centrálním prvku rozhraní. Odezvu lze uložit ve formátu *WAV* kliknutím na možnost *Options* → *Save response* v této záložce.

Panel *3D viewport* obsahuje perspektivní pohled do místnosti. Kamerou lze pohybovat pomocí kláves W,A,S,D. Výšku pohledu lze měnit pomocí mezerníku a levé klávesy control. Pohyb lze urychlit držením klávesy levý shift při pohybu. Rotaci kamery lze provést pomocí šipek klávesnice, nebo přidržením středního tlačítka myši a následným pohybem myši. Kliknutím a tažením některé ze směrových šipek lze upravovat pozici posluchače a zdroje zvuku přímo v tomto pohledu.

Nabídka *Options* umožňuje zobrazit drátový model místnosti, zobrazit nebo skrýt cesty nalezené při simulaci, obrátit normály ploch místnosti a zobrazit náhledy na odrazivost a hrubost povrchu. Nabídka *Playback* umožňuje přehrát průběh simulace pomocí částicového efektu, nastavit počet částic, jejich rychlost a velikost.

Panel *Auralization* umožňuje provést konvoluci s jiným zvukem poté co je dokončena simulace. Kliknutím na tlačítko *[...]* v tomto panelu lze načíst zvukový soubor a tlačítkem *Convolve* provést konvoluci. Tlačítkem *Save output* lze uložit výsledný zvukový soubor. Před konvolucí není nutné upravovat vzorkovací frekvenci podle nahraného zvuku, výsledek simulace je automaticky převzorkován.



Obrázek B.1: Spuštěná aplikace s GUI.