České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

Bakalárska práca

# Editor materiálov pre PBRT renderer

Stanislav Ľaš

Vedúci práce: Ing. Ivo Malý, Ph. D.

Študijný program: Otevřená informatika

Odbor: Software

2019

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ľaš** Jméno: **Stanislav** Osobní číslo: **466294**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Software**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Editor materiálů pro pbrt renderer**

Název bakalářské práce anglicky:

Pokyny pro vypracování:

Analyzujte strukturu textových souborů rendereru pbrt. Zaměřte se zejména na způsob definice materiálů a práci uživatelů s reálnými soubory. Na základě User-centred design metodiky navrhněte prototyp grafického editoru pro tvorbu a editaci materiálů. Prototyp otestujte s alespon 3 uživateli. Na základě prototypu vytvořte finální multiplatformní desktopovou aplikací s využitím platformy JavaFX. Aplikaci testujte pomocí SW i uživatelských testů.

Seznam doporučené literatury:

[1] K. Boogaart, You, I, and ReactiveUI, 2018.
[2] J. Vos, S. Chin, W. Gao, J. Weaver, D. Iverson Pro JavaFX 9: A Definitive Guide to Building Desktop, Mobile, and Embedded Java Clients, Apress, 2017.
[3] T. Lowdermilk, User-Centered Design, O'Reilly Media, 2013.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**Ing. Ivo Malý, Ph.D., katedra počítačové grafiky a interakce FEL**

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

Datum zadání bakalářské práce: **14.02.2019** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **20.09.2020**

| Ing. Ivo Malý, Ph.D.<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Ing. Pavel Ripka, CSc.<br>podpis děkana(ky) |
|---|---|---|

## III. PŘEVZETÍ ZADÁNÍ

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

21.5.2019
Datum převzetí zadání

Podpis studenta

## Poďakovanie

Chcel by som poďakovať vedúcemu tejto bakalárskej práce, ktorým je pán Ing. Ivo Malý, Ph. D. za odbornú podporu, trpezlivosť a ochotu pomôcť pri prekonávaní prekážok pri písaní tejto práce na tomto projekte. V ďalšom rade by som sa chcel poďakovať všetkým účastníkom testovania, ktorí sa tiež zaslúžili o úspešné ukončenie tejto práce. V neposlednom rade by som sa chcel poďakovať mojej rodine a priateľom, ktorí ma podporovali a povzbudzovali nielen pri tejto práci, ale počas celého môjho štúdia.

## Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne, a že som uviedol všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržiavaní etických princípov pri príprave vysokoškolských záverečných prác.

V Prahe dňa 24. mája 2019 .....

# Abstrakt

Táto bakalárska práca sa zaoberá analýzou, návrhom a implementáciou aplikácie, ktorá slúži ako editor a úložisko materiálov pre renderer PBRT-v3. Aplikácia umožňuje vytvárať, mazať, editovať a ukladať materiály podľa potreby užívateľa. Aplikácia je schopná načítavať a ukladať materiály do formátu určeného pre PBRT renderer. Na implementáciu aplikácie bola použitá platforma JavaFX, návrhový vzor MVVM a framework mvvm(FX). Ako databáza bol použitý súbor JSON. Výsledkom je aplikácia na editáciu a ukladanie materiálov pre PBRT-v3 renderer.



**Kľúčové slová**: editor, framework, java, javafx, json, materiály, mvvm, mvvm(fx), návrhové vzory, pbr, pbrt, pbrt-v3, renderer

**Vedúci práce**: Ing. Ivo Malý, Ph. D.



# Abstract

This bachelor thesis deals with analysis, design and implementation of an application which servers as editor and repository for PBRT-v3 renderer materials. The application provides creating, deleting, editing and storing materials as needed by the user. The application is able to load and store materials in the specified format for the PBRT renderer. JavaFX platform, MVVM design pattern and mvvm(FX) framework were used for implementation of the application. JSON file was used as the database. The result of the thesis is an application for editing and storing materials for the PBRT-v3 renderer.



**Key words**: design paterns, editor, framework, java, javafx, json, materials, mvvm, mvvm(fx), pbr, pbrt, pbrt-v3, renderer

**Title translation**: Material editor for PBRT renderer

# Obsah

# 1 Úvod

Úlohou tejto práce je vytvoriť grafický editor pre PBRT renderer [1], v ktorom bude užívateľ rýchlo a efektívne pracovať s rendererom. V súčasnosti existuje mnoho editorov pre rôzne rendereri, no pre PBRT renderer nebol vytvorený žiaden. Niektoré editory ponúkajú rozšírenia, vďaka ktorým je možné pracovať aj s PBRT rendererom. To je motivácia pre vytvorenie editoru práve pre PBRT renderer [1]. Renderer pracuje na základe PBR[1] čo je model, ktorý zobrazuje grafiku takým spôsobom, aby presnejšie modelovala tok svetla v reálnom svete. Tento model sa zrodil v 80. rokoch minulého storočia na Cornell University v programe počítačovej grafiky. Model sa dostal do povedomia hlavne vďaka M. Pharrovi, G. Humphreysovi a P. Hanrahanovi v knihe [2], kde je tento model popísaný. Kniha naviac obsahuje implementáciu rendereru PBRT.

Keďže neexistuje editor pre PBRT renderer [1] užívatelia musia s rendererom pracovať priamo cez súbory, ktoré sú vstupom rendereru. Navyše vykresľovanie zložitých scén zaberie veľa času a energie. Pri používaní editora nebude musieť užívateľ poznať zložitú syntax vstupného súboru PBRT a to umožní prácu s renderom aj pre ľudí, ktorí nemajú veľa skúseností s renderovaním. Úprava materiálu bude rýchlejšia, pretože sa nebude vykresľovať celá scéna, ale len jeden objekt s príslušným materiálom do jednoduchej scény. To zrýchli vykresľovanie a tým aj úpravu. V editore si užívateľ bude môcť ukladať materiály, ktoré bude môcť neskôr použiť pri editácií iných materiálov.

Práca sa skladá zo štyroch častí a to analýza, návrh, implementácia a záver. V každej časti sa posúvame krok po kroku k výslednej aplikácií, ktorá má za úlohou spríjemniť a zrýchliť užívateľovi prácu s PBRT rendererom [1].

V analýze budeme skúmať prácu s rendererom bez editoru. Ukážeme si jednoduchý súbor, prácu s ním, samotné vykresľovanie a aj finálny obrázok. Budeme analyzovať editory pre iné rendereri a porovnávať ich vlastnosti, čo nám pomôže pri návrhu editoru pre PBRT. Taktiež si definujeme funkčné a nefunkčné požiadavky na základe prianí užívateľov.

Návrh sa zaoberá vytváraním konceptu aplikácie. Navrhneme prototyp podľa zozbieraných požiadaviek a zadefinujeme funkcie editoru vzhľadom k prototypu. Určíme aké informácie o materiáloch budeme musieť ukladať a tým si pomôžeme k výberu úložiska pre materiály. Výsledkom tejto časti bude prototyp aplikácie, ktorý na záver návrhovej časti podrobíme testovaniu s troma potenciálnymi užívateľmi, zhodnotíme výsledky testovania a závažné nedostatky opravíme, čím dostaneme nový prototyp, ktorý neskôr využijeme ako základ aplikácie, ktorú budeme implementovať v ďalšej časti práce.

[1] Physically based rendering

Implementácia rozoberá použité technológie v závislosti na návrhu aplikácie, ktoré budeme vyberať podľa zložitosti používania a ich väzieb na iné technológie aby aplikácia spĺňala požiadavky, ale na druhej strane aby nebola zbytočne príliš robustná. Najprv vyberieme návrhový vzor, čo bude mať vplyv na výber frameworku a neskôr zvolíme úložisko podľa požiadavku na aplikáciu. Následne nájdeme vhodný framework, ktorý bude spĺňať predpoklady návrhového vzoru a bude kompatibilný s ostatnými technológiami, ktoré budeme používať. Po výbere programovacieho jazyka opíšeme štruktúru projektu a definujeme časti aplikácie. Ďalej popíšeme funkcie každej obrazovky. Na záver tejto časti podrobíme hotovú aplikáciu testovaniu s potenciálnymi užívateľmi, z ktorého spracujeme výsledky a pri závažných nedostatkoch upravíme aplikáciu, tak aby práca s aplikáciou bola intuitívna a príjemná.

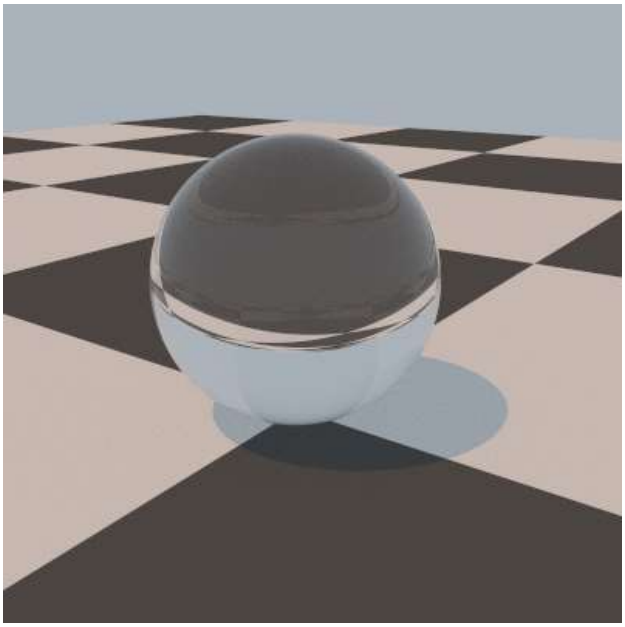
Na záver zhodnotíme výsledky tejto práce a jednotlivých častí. Predstavíme finálnu aplikáciu a navrhneme prípadné ďalšie vylepšenia aplikácie.

# 2 Analýza

Budeme pracovať s rendererom PBRT-v3 [1], ktorý je v súčasnosti najnovšou verziou tohto rendereru. *Obrázok 1* znázorňuje súbor s príponou „pbrt“, ktorý slúži ako vstup pre renderer. V tomto súbore je nadefinovaná celá scéna od rozlíšenia obrázku cez polohu a farbu svetla až k samotným objektom a ich materiálom. Každý materiál má svoje parametre, ktoré majú štandardne nastavené hodnoty. Parametre materiálov sú charakteristické tým, že na ich špecifikáciu môžeme použiť textúry alebo konštantné hodnoty. PBRT má definované štandardné materiály (disney, fourier, glass, hair, kdsubsurface, matte, metal, mirror, mix, plastic, substrate, subsurface, translucent, uber), ktoré môžeme upravovať alebo ich môžeme medzi sebou kombinovať. *Obrázok 1* je príkladom súboru *pbrt*, kde je definovaná celá scéna a po vykreslení tohto súboru rendererom dostaneme *Obrázok 2*. [3]

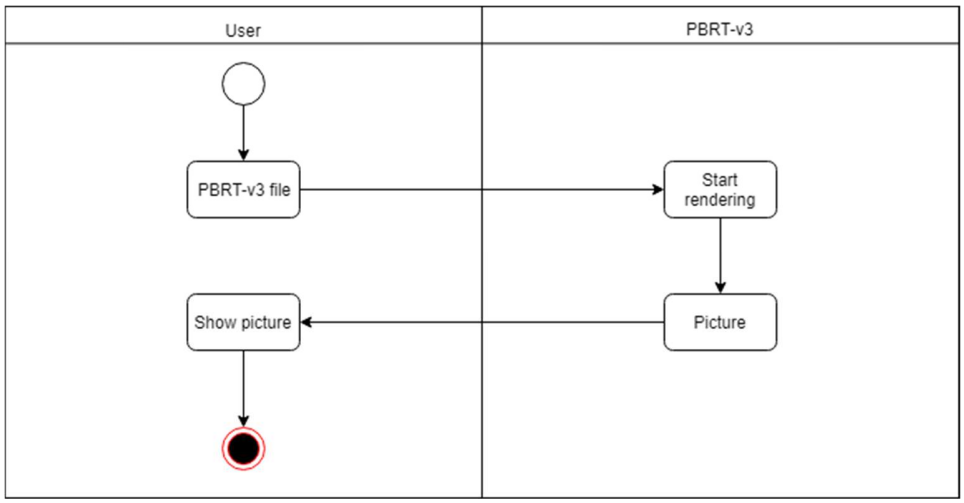
```
LookAt 3 4 1.5  # eye
       .5 .5 0  # look at point
       0 0 1    # up vector
Camera "perspective" "float fov" 45

Sampler "halton" "integer pixelsamples" 128
Integrator "path"
Film "image" "string filename" "simple.png"
     "integer xresolution" [400] "integer yresolution" [400]

WorldBegin

# uniform blue-ish illumination from all directions
LightSource "infinite" "rgb L" [.4 .45 .5]

# approximate the sun
LightSource "distant"  "point from" [ -30 40  100 ]
   "blackbody L" [3000 1.5]

AttributeBegin
  Material "glass"
  Shape "sphere" "float radius" 1
AttributeEnd

AttributeBegin
  Texture "checks" "spectrum" "checkerboard"
          "float uscale" [8] "float vscale" [8]
          "rgb tex1" [.1 .1 .1] "rgb tex2" [.8 .8 .8]
  Material "matte" "texture Kd" "checks"
  Translate 0 0 -1
  Shape "trianglemesh"
      "integer indices" [0 1 2 0 2 3]
      "point P" [ -20 -20 0   20 -20 0   20 20 0   -20 20 0 ]
      "float st" [ 0 0   1 0    1 1   0 1 ]
AttributeEnd

WorldEnd
```

*Obrázok 1: Príklad vstupného súboru PBRT [3]*



*Obrázok 2: Príklad vygenerovanej scény [3]*

## 2.1 Práca s PBRT




*Obrázok 3: Proces vytvárania obrázku*

*Obrázok 3* nám ilustruje vytváranie obrázka s PBRT rendererom. Na začiatku užívateľ vytvorí súbor s definíciou scény a potom cestu k tomuto súboru zadá na vstup rendereru, ktorý sa postará o vykreslenie obrázku. Samotná práca pozostáva z niekoľkých iterácií, až užívateľ dospeje k požadovanému výsledku. Nevýhodou je, že užívateľ musí ovládať syntax *pbrt*. Ten sa však líši aj medzi

jednotlivými verziami rendereru. Ďalšou nevýhodou môže byť aj čas vykresľovania obrázku, ktorý závisí na zložitosti scény a kvalite obrázku, čo v niekoľkých iteráciách môže znamenať stratený čas. [3]

## 2.2 Štruktúra vstupného súboru

Súbor začína sériou direktív popisujúcich kameru, film a svetelné transporty, ktoré sú potrebné pri vykresľovaní scény. Nasleduje definícia sveta, ktorá sa začína direktívou „WorldBegin“ a končí direktívou „WorldEnd“. V definícií sveta nie je dovolené špecifikovať objekty, ktoré sú definované na začiatku súboru. Môžeme však definovať svetlá, materiály, textúry a objekty v scéne. [3]

### 2.2.1 Parametre

Parametre sú definované v zozname za objektom, ktorý popisujú. Zoznam môže obsahovať ľubovoľný počet parametrov a každý parameter je definovaný ako dvojica názov parametru v úvodzovkách a jeho hodnota v hranatých zátvorkách. [3]

```
Material "matte" "rgb Kd" [ .4 .4 .7 ]  "float sigma" [20]
```

*Obrázok 4: Definícia materiálu so zoznamom parametrov [3]*

### 2.2.2 Atribúty

Aktuálny stav grafiky je možné uložiť pomocou atribútov. Začiatok atribútu je definovaný direktívou „AttributeBegin“ a koniec direktívou „AttributeEnd“. *Obrázok 5* zobrazuje kus kódu, kde môžeme vidieť, že na začiatku definujeme materiál *matte*, následne definujeme atribút a v ňom materiál *plastic*. Pretože sa guľa nachádza vo vnútri atribútu bude jej priradený materiál *plastic*, zatiaľ čo kužeľ, ktorý je definovaný mimo atribútu bude z materiálu *matte*. [3]

```
1.  Material "matte"
2.  AttributeBegin
3.     Material "plastic"
4.     Shape "sphere"
5.  AttributeEnd   # back to the "matte" material
6.  Shape "cone"
```

*Obrázok 5: Definícia atribútu [3]*

### 2.2.3 Materiály

Materiály určujú vlastnosti rozptylu svetla na povrchu scény. V súbore sú definované direktívou „Material“, po ktorej nasledujú parametre materiálu ako zachytáva *Obrázok 6*. [3]

```
Material "matte" "rgb Kd" [ .7 .2 .2 ]
```

*Obrázok 6: Definícia materiálu [3]*

Táto definícia špecifikuje materiál pre všetky nasledujúce objekty až do výskytu ďalšej definície materiálu alebo výskytu atribútu. Parametre materiálov môžeme definovať aj textúrami, ktoré môžu byť použité na špecifikáciu hodnôt meniacich sa vzhľadom na priestor. *Obrázok 6* definuje materiál, ktorý má

rovnakú farbu vo všetkých bodoch. Ak chceme aby farba nebola všade rovnaká, musíme použiť obrázkovú mapu, v ktorej definujeme farbu podľa funkcie na povrchu objektu. To je možné pomocou textúr, ako definuje *Obrázok 7*. [3]

```
Texture "lines-tex" "spectrum" "imagemap" "string filename" "textures/lines.exr"

Material "matte" "texture Kd" "lines-tex"
```

*Obrázok 7: Definícia materiálu pomocou textúry [3]*

Najprv definujeme textúru, ktorá je viazaná na obrázkovú mapu „lines.exr“ a následne ju použijeme v materiáli. Parameter *Kd* už nemá typ *rgb* ale typ *texture* a ako hodnota parametru je názov vytvorenej textúry. [3]

## 2.3 Existujúce riešenia

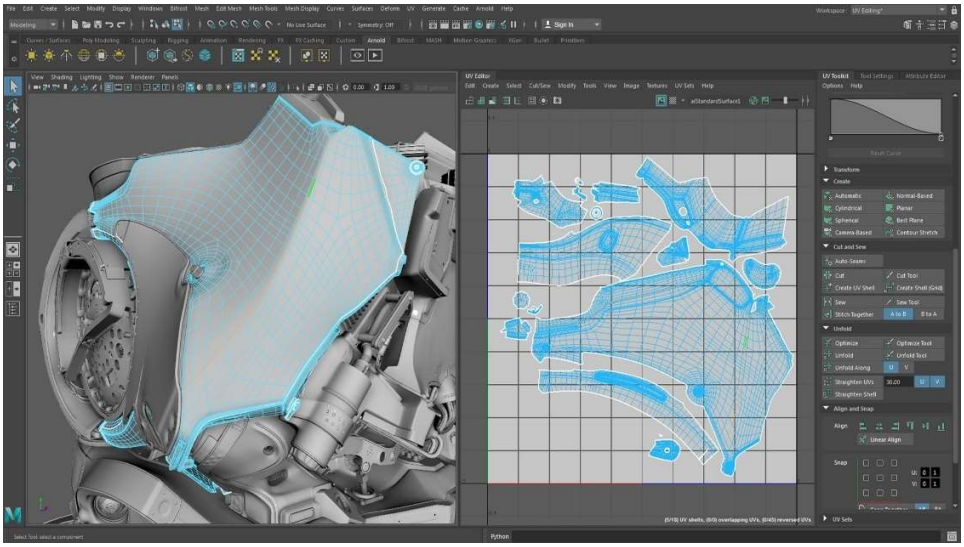
V dnešnej dobe existuje mnoho editorov pre rôzne rendereri, či už profesionálne alebo len pre nadšencov grafiky, ktorý vytváraním obrázkov trávia voľný čas. PBRT nemal to šťastie a editor pre tento renderer neexistuje, ale existujú rôzne rozšírenia pre editori, ktoré môžu byť použité na prácu s PBRT. Najznámejšie editory sú *Autodesk Maya* [4] a *Blender* [5], do týchto editorov je možné si stiahnuť PBRT rozšírenie.

### 2.3.1 Autodesk Maya

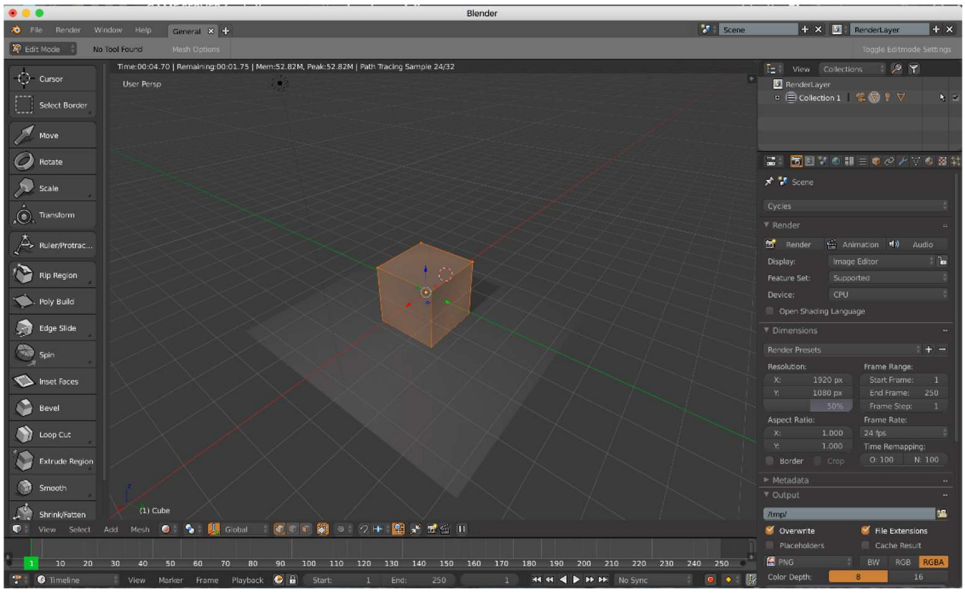
*Autodesk Maya* [4] je profesionálny editor na 3D modelovanie, ktorý má výborné nástroje na vytváranie animácií a preto sa využíva v grafickom priemysle pri vytváraní počítačových hier. Taktiež má veľmi príjemné užívateľské rozhranie, takže užívateľ pracuje s editorom veľmi intuitívne. Najväčšou nevýhodou tohto editoru je, že nie je inovatívny, čo je spôsobené tým že sa využíva v spoločnostiach na vyvíjanie počítačových hier a preto musia byť používané nástroje čo najspoľahlivejšie. Tento editor nie je zadarmo a treba si kúpiť licenciu. *Obrázok 8* zobrazuje editor *Autodesk Maya*. [6]

### 2.3.2 Blender

*Blender* [5] sa pri vytváraní počítačových hier využíva menej, no jeho najväčšou výhodou je, že je zadarmo a nie je potrebné kupovať licenciu. Tento fakt vedie k ďalšej výhode a to takej, že je veľmi jednoduché nájsť materiály na prácu v tomto editore. Existuje mnoho online cvičení, príkladov a fór, takže užívateľ sa v tomto editore rýchlo naučí pracovať a na fórach môže diskutovať s ostatnými užívateľmi pracujúcimi s týmto editorom, ktorý zobrazuje *Obrázok 9*. V kvalite užívateľského rozhrania mierne zaostáva za editorom *Autodesk Maya*. [6]



*Obrázok 8: Autodesk Maya [4]*




*Obrázok 9: Blender [5]*

## 2.4 Funkčné požiadavky (FR[2])

FR1 Načítať zo súboru

- Editor umožní užívateľovi načítať materiály, ktoré sa nachádzajú v súbore.

FR2 Zobraziť materiály

- Editor umožní užívateľovi zobraziť materiály v jednoduchej scéne.

FR3 Editovať materiál

- Editor umožní užívateľovi editovať načítané materiály.

FR4 Uložiť zmeny

- Editor umožní užívateľovi uložiť zmeny, ktoré vykonal.

FR5 Editovať súbor

- Editor umožní užívateľovi editovať materiály a zmeny zapísať do súboru.

FR6 Zobraziť obrázok

- Editor umožní užívateľovi zobraziť vykreslený obrázok.

FR7 Uložiť materiály

- Editor umožní užívateľovi uložiť materiály.

FR8 Načítať uložené materiály

- Editor umožní užívateľovi načítať uložené materiály.

FR9 Editovať uložené materiály

- Editor umožní užívateľovi editovať uložené materiály.

FR10 Vytvoriť materiál

- Editor umožní užívateľovi vytvoriť nový materiál.

FR11 Zmazať materiál

- Editor umožní užívateľovi zmazať uložený materiál.

FR12 Použiť materiály z iného súboru

- Editor umožní užívateľovi použiť materiály z iného súboru a ďalej s nimi pracovať.

FR13 Použiť uložené materiály

- Editor umožní užívateľovi použiť uložené materiály na aktuálne otvorený materiál.

FR14 Vytvárať verzie materiálu

- Editor umožní užívateľovi vytvoriť novú verziu materiálu.

FR15 Zmazať verzie materiálu

- Editor umožní užívateľovi zmazať označenú verziu materiálu.

FR16 Zobraziť verzie materiálu

[2] Functional requirement – funkčné požiadavky

- Editor umožní užívateľovi zobraziť verzie aktuálneho materiálu.

FR17 Uložiť verziu ako nový materiál

- Editor umožní užívateľovi uložiť vybranú verziu ako nový materiál.

FR18 Uložiť verziu namiesto materiálu

- Editor umožní užívateľovi uložiť verziu namiesto aktuálne vybraného materiálu.

## 2.5 Nefunkčné požiadavky (NFR[3])

NFR1 Platforma

- Editor bude multiplatformový.

NFR2 Responzivita

- Editor bude responzívny.

NFR3 Úložisko

- Editor bude obsahovať úložisko na materiály.

NFR4 Bezpečnosť

- Editor bude ukladať údaje na disk aby ich nestratil.

NFR5 Rozšíriteľnosť

- Editor bude pripravený na prípadné rozšírenie.

[3] Non-functional requirement - kvalitatívna požiadavka

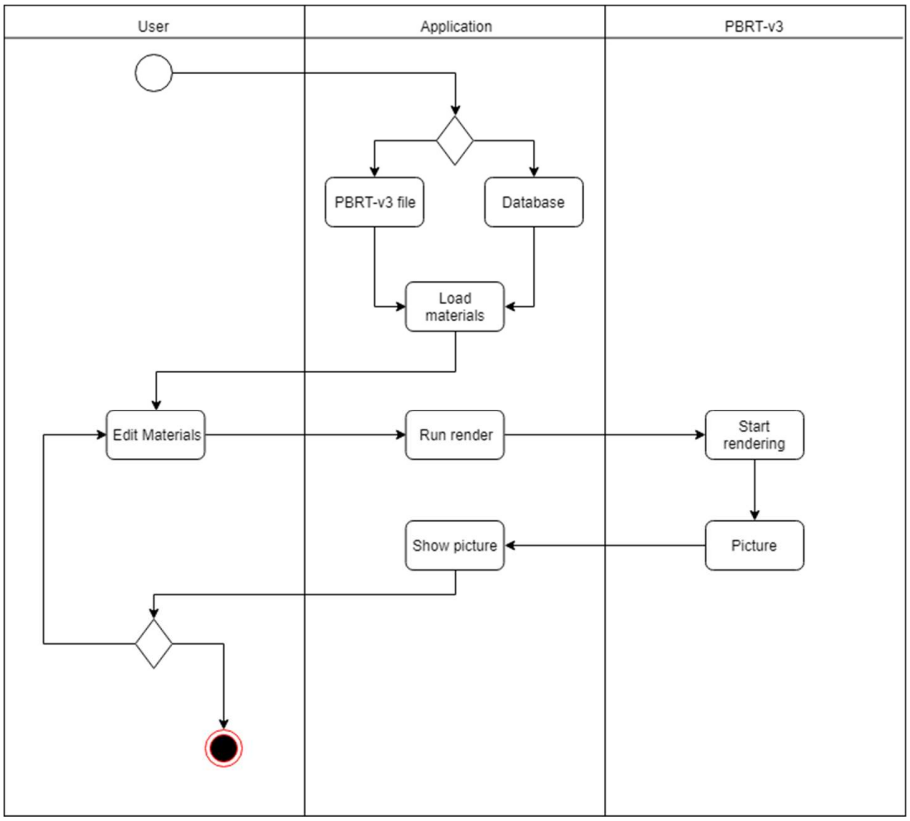
# 3 Návrh

V tejto časti navrhneme editor na základe funkčných a nefunkčných požiadaviek a inšpirujeme sa existujúcimi riešeniami. Výstupom bude prototyp, ktorý podrobíme testovaniu.

## 3.1 Úložisko

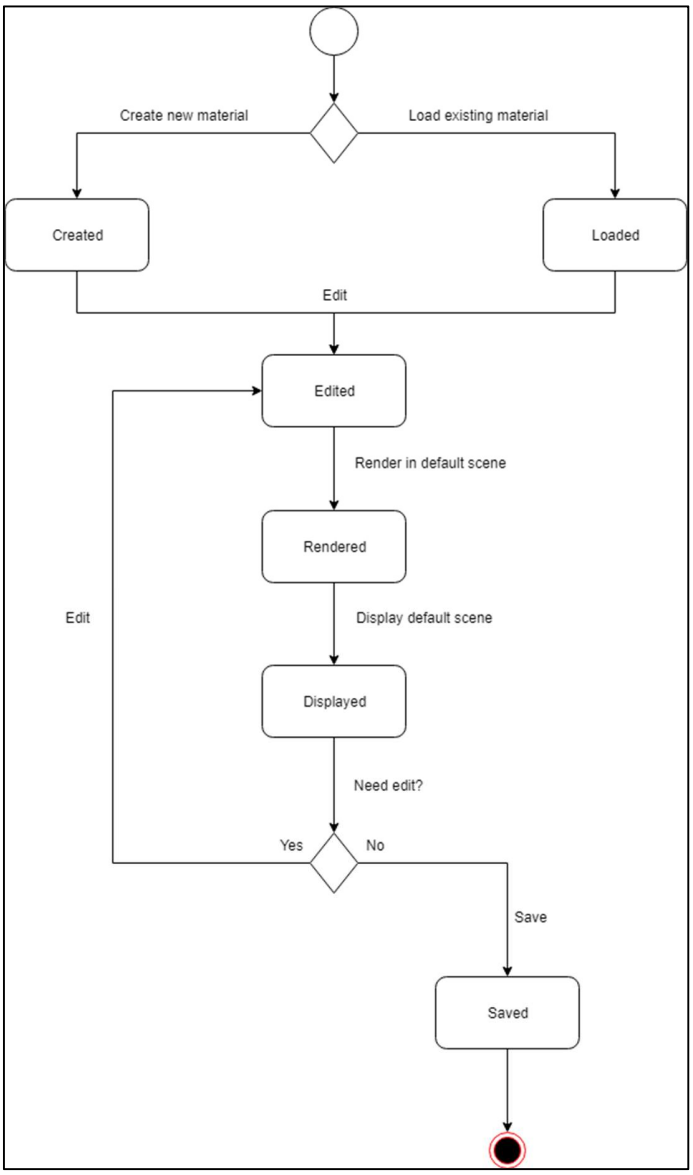
Pri výbere úložiska máme niekoľko možností. Do úvahy prichádza pamäť programu, no to by nebolo veľmi výhodné, pretože pri vypnutí aplikácie by sme prišli o všetky údaje. Navyše pri neočakávaných udalostiach ako zrútenie systému, reštart systému, výpadok prúdu a mnoho ďalších by sme stratili aj všetku našu prácu. Preto potrebujeme spoľahlivé úložisko, čo môže byť priamo disk alebo databáza. Môžeme použiť ukladanie na disk, čím by sme vyriešili problém so stratou súborov, no vyhľadávanie by bolo komplikované a pomalé. Ďalšia možnosť je využiť databázu. Tu narazíme na problém s ukladaním obrázkov, čo v relačnej databáze nie je možné. Preto zvolíme kombináciu databázy a ukladania na disk, pretože potrebujeme ukladať parametre materiálov a zároveň aj vykreslené obrázky materiálov. Obrázky budú uložené na disku a v databáze bude uložená cesta k obrázkom.

## 3.2 Práca s editorom




*Obrázok 10: Proces vytvárania obrázku s editorom*

*Obrázok 10* znázorňuje proces vytvárania obrázku s editorom. Na začiatku sa užívateľ rozhodne či chce pracovať s jeho uloženými materiálmi alebo chce načítať materiály zo súboru. Následne sa mu zobrazia materiály, s ktorými chce pracovať. Úprava môže pozostávať z niekoľkých iterácií. Jedna iterácia nezaberie veľa času, pretože zvolený materiál sa vykreslí do jednoduchej scény, čo je oveľa rýchlejšie ako vykresľovať celú scénu so všetkými materiálmi. Užívateľ taktiež môže použiť materiály, ktoré má uložené v databáze. O vykresľovanie sa stará editor, ktorý spúšťa renderer na pozadí. Po dokončení práce užívateľ jednoducho uloží vykonané zmeny alebo exportuje materiály do súboru, z ktorého materiály vybral.



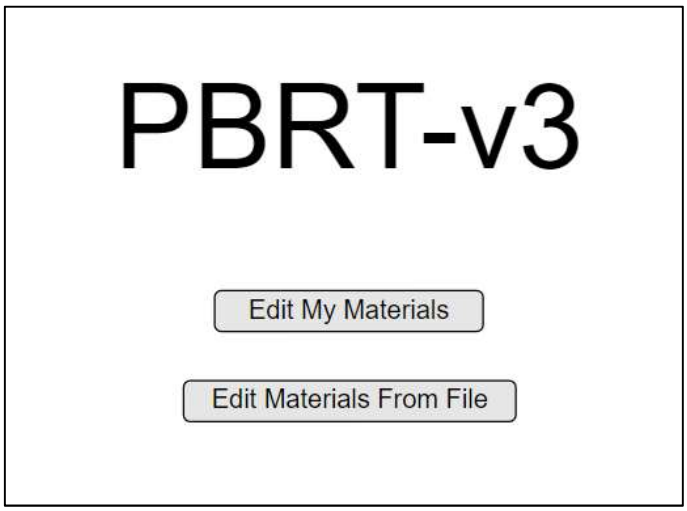

*Obrázok 11: Stavový diagram materiálu*

Stav materiálu je v každom okamihu jednoznačne určený, čo zobrazuje *Obrázok 11*, kde môžeme vidieť stavy, do ktorých sa môže materiál dostať. Po vytvorení je v stave „Created“, to znamená, že potrebuje editáciu. Po editácií sa vykreslí a zobrazí v jednoduchej scéne. Následne sa užívateľ rozhodne, či tento materiál potrebuje ďalšiu editáciu alebo ho chce uložiť. Po uložení prechádza do stavu „Saved“, v ktorom zostáva až do okamihu následnej editácie, a tým prechádza do stavu „Loaded“ a opäť prechádza všetkými stavmi.

## 3.3 Prototyp

Prototyp bol vytváraný pomocou voľne dostupného softvéru *Moqups* [7]. Pri návrhu prototypu sme mali problémy s prechodmi medzi prácou s databázou a súborom. Problém sme vyriešili úvodnou obrazovkou, v ktorej si užívateľ môže vybrať zdroj materiálov. Tento problém sme potrebovali vyriešiť aj pre samotný editor, aby si užívateľ mohol načítavať materiály iné ako vybral na úvodnej obrazovke a používať ich pri editácií materiálov z vybraného zdroja. Z tohto dôvodu sme pridali ďalšiu obrazovku, ktorá slúži na výber materiálu z iného súboru alebo z databázy (ak momentálne pracujeme so súborom). Nakoniec sme došli k záveru, že aplikácia bude pozostávať z troch obrazoviek, a to úvodná obrazovka, editovacia obrazovka a obrazovka na výber iných materiálov.

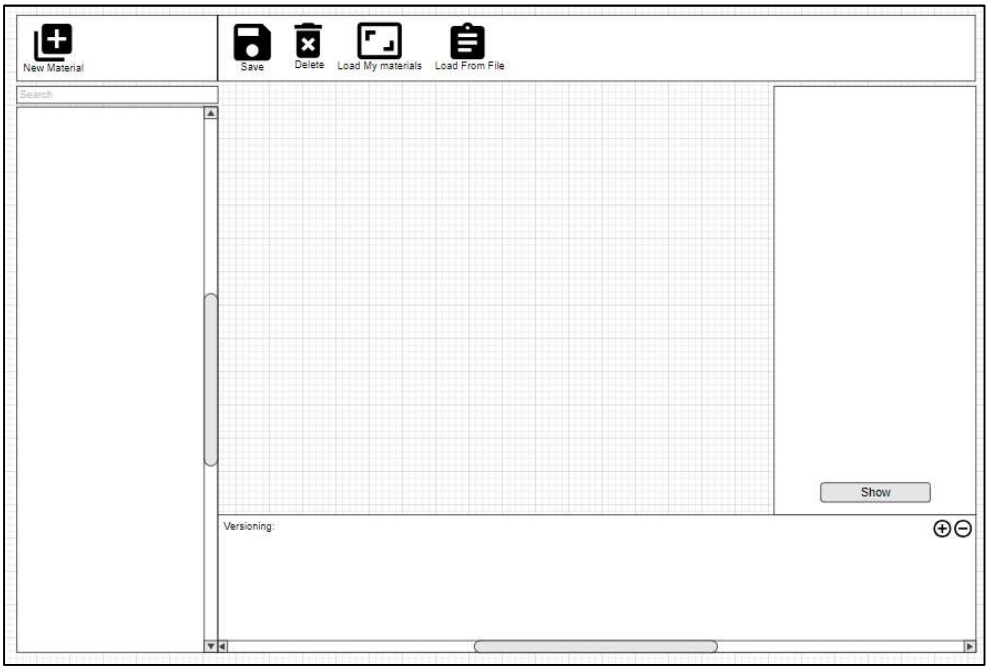
### 3.3.1 Úvodná obrazovka




*Obrázok 12: Úvodná obrazovka*

*Obrázok 12* znázorňuje úvodnú obrazovku, ktorá sa otvorí po spustení aplikácie. Na tejto obrazovke bude mať užívateľ dve možnosti. Vybrať materiály uložené v databáze tlačidlom „Edit My Materils“ alebo vybrať súbor, z ktorého sa načítajú materiály tlačidlom „Edit Materials From File“.

### 3.3.2 Editor




*Obrázok 13: Editor*

*Obrázok 13* zobrazuje hlavnú obrazovku aplikácie, ktorým je editor materiálov. Obrazovka je rozdelená na päť častí. V hornej časti je panel nástrojov, v ktorom môžeme vytvárať nové materiály, ukladať zmeny, mazať materiály a načítavať materiály z databáze alebo z iných súborov. V ľavej časti sa zobrazia materiály podľa možnosti, ktorá bola zvolená užívateľom na úvodnej obrazovke. Prvok zoznamu bude pozostávať z obrázka materiálu a názvu materiálu. Po výbere konkrétneho materiálu sa v strednej časti obrazovky zobrazí obrázok materiálu a v pravej časti sa zobrazí názov materiálu a jeho parametre, ktoré bude môcť užívateľ upravovať. Po stlačení tlačidla „Show“ sa vykreslí a zobrazí obrázok materiálu. Dolná časť obrazovky patrí panelu s verziami, kde bude prvok pozostávať z obrázka a názvu, podobne ako v paneli materiálov v ľavej časti obrazovky. Po vybraní inej verzie materiálu sa zobrazí obrázok a parametre vybranej verzie. Verzie je možné pridávať tlačidlami „+“ a „-“, taktiež pri zmene typu materiálu alebo načítaní materiálu z databáze alebo súboru sa vytvorí nová verzia s načítaným materiálom. Ak chce užívateľ verziu uložiť ako zvolený materiál, stlačí tlačidlo „Save“ a verzia, ktorá je vybraná nahradí originálny materiál.

### 3.3.3 Načítavacia obrazovka


*Obrázok 14: Načítavacia obrazovka*

Po výbere „Load My materials“ alebo „Load From File“ sa zobrazí načítavacia obrazovka, ktorú ilustruje *Obrázok 14*. Obrazovka v dolnej časti zobrazí materiály zo zvoleného zdroja a po stlačení tlačidla „Choose“ sa zvolený materiál načíta do editoru ako nová verzia materiálu, kde ho môžeme ďalej editovať alebo uložiť.

## 3.4 Testovanie

Testovanie prebehlo postupne s troma účastníkmi, ktorí sa pohybujú v oblasti grafickej tvorby. Testovali sme predstavený prototyp, z ktorého sme vytvorili Lo-Fi[4] prototyp [8], ktorý sme účastníkom na začiatku testovania predstavili a podali im základné informácie o funkciách aplikácie. Následne dostali desať úloh, ktoré museli intuitívne splniť. Postup práce bol zaznamenávaný a nejasnosti boli ihneď vysvetlené. Na záver dostali účastníci priestor na vlastné pripomienky.

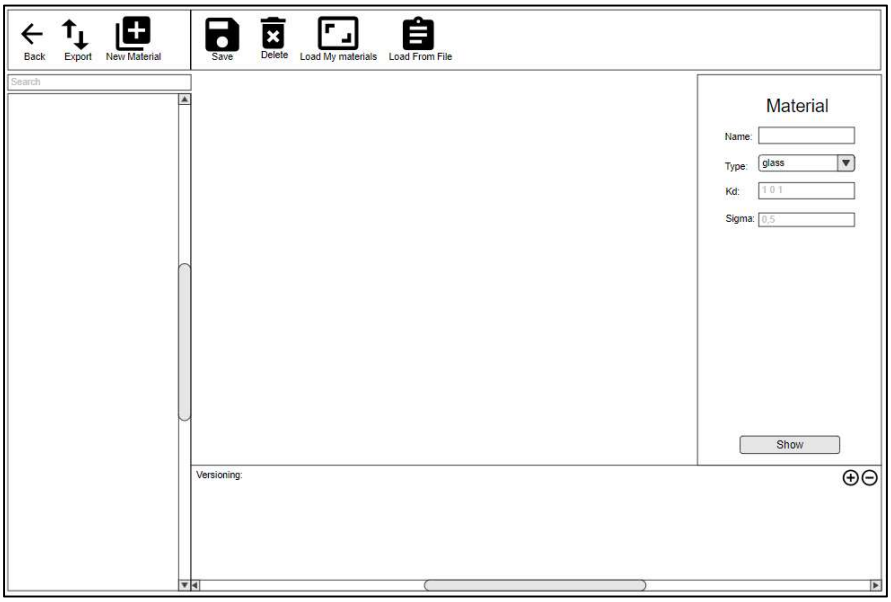
**Úlohy:**

1. Vytvorte ružový matný materiál
2. Vytvorte zelenú verziu materiálu z úlohy 1

[4] Low Fidelity

3. Uložte ružovú verziu z úlohy 1 a zelenú verziu z úlohy 2 odstráňte
4. Upravte ružový materiál na matný biely materiál a materiál uložte
5. Načítajte materiály zo scény „lte-orb-roughglass“, ktorá je uložená na disku
6. Uložte modrý materiál zo súboru z úlohy 5 do databázy
7. Vytvorte novú verziu materiálu z úlohy 6, ktorá je načítaná z materiálu v kroku 4 a uložte zmeny do súboru
8. Vyberte druhý materiál zo scény v kroku 5
9. Načítajte materiály zo scény v súbore „lte-orb-silver“ z disku a materiál z úlohy 8 nahraďte modrým materiálom z tejto scény
10. Otvorte materiály z databázy a vymažte materiál z úlohy 4

Pri testovaní sme zistili malé nedostatky, ktoré sa objavili na obrazovke *Editor*. Účastníkom sa nepáčil názov tlačidla „New Material“ pri práci so súborom, keďže pri práci so súborom nemá toto tlačidlo zmysel, pretože v súbore sú pevne určené objekty a ich materiály a tým pádom nie je možné vytvoriť nový materiál v súbore. Preto toto tlačidlo slúži na ukladanie zvoleného materiálu do databázy. Po objasnení účastníci súhlasili, no odporúčali zmeniť názov tlačidla pri práci so súborom na „Save to My Materials“. Ďalším nedostatkom bola absencia tlačidiel na export materiálov naspäť do súboru, z ktorého boli načítané a tlačidlá pre krok na úvodnú obrazovku, kde je jediná možnosť na výber zdroja materiálov, s ktorým chceme pracovať. *Obrázok 15* predstavuje editor, ktorý bol prepracovaný podľa pripomienok účastníkov.




*Obrázok 15: Editor*

# 4 Implementácia

Táto kapitola sa zaoberá implementáciou aplikácie, ktorú sme navrhli v predchádzajúcej kapitole. Na začiatku predstavíme použité technológie, ďalej rozoberieme štruktúru projektu a popíšeme jednotlivé časti a obrazovky aplikácie. Na záver pridáme výsledky užívateľského testovania.

## 4.1 Použité technológie

Pri vyberaní programovacieho jazyka sme sa rozhodovali medzi jazykom C# (WPF[5]) [9] a jazykom Java (JavaFX) [10]. S požiadaviek na aplikáciu sme nakoniec vybrali jazyk Java, kvôli jeho podpore vývoja multiplatformových aplikácií. Ako správcu balíčkov sme zvolili *Apache Maven* [11], vďaka ktorému sme mohli použité balíčky spravovať v súbore *pom.xml*.

### 4.1.1 Návrhový vzor

Po vybraní jazyka sme sa mohli zamerať na výber návrhového vzoru. V súčasnosti poznáme tri návrhové vzory MVC[6], MVP[7] a MVVM[8]. JavaFX je dizajnovaná pre prácu s MVC, čo je základný vzor na rozdelenie logiky aplikácie a užívateľského rozhrania. MVC je dobrou voľbou pri tvorbe webových aplikácií, pretože webová štruktúra prirodzene podporuje rozdelenie komponentov pre vzor MVC. Tento vzor má však neefektívny prístup k údajom vo View a predpokladá rozdelenie častí View a Controller, čo v praxi nezodpovedá stavu mnohých frameworkov. Preto sme použili návrhový vzor MVVM, ktorého výhodami sú, že podporuje používanie väčšieho počtu View na rovnaké dáta a taktiež podporuje automatickú synchronizáciu dát medzi časťami View a ViewModel. Táto synchronizácia môže ovplyvňovať výkon aplikácie, no v našom prípade je to zanedbateľné. [12]

### 4.1.2 Framework

Podľa vybraného jazyka a návrhového vzoru sme museli nájsť vhodný framework, ktorý bude podporovať naše požiadavky. Zvolený framework sa nazýva **mvvm(FX)** [13], čo je open source framework na implementáciu MVVM vzoru pre jazyk JavaFX. Tento framework je však nekompatibilný s verziou Java 9 a vyššie, preto sme na implementáciu použili verziu Java 8.

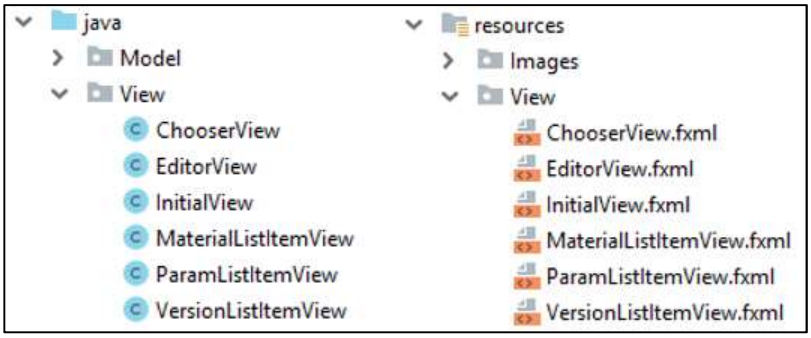
Framework implementuje View a ViewModel pomocou rozhraní „FxmlView<>“ a „ViewModel“.

```
public class ChooserView implements FxmlView<ChooserViewModel>
public class ChooserViewModel implements ViewModel
```

*Obrázok 16: ViewModel a View triedy*

[5] Windows Presentation Foundation
[6] Model-View-Controller
[7] Model-View-Presenter
[8] Model-View-ViewModel

View pozostáva z dvoch častí a to je súbor s príponou *fxml* a trieda v jazyku *JavaFX*. Tieto súbory musia mať rovnaký názov a byť umiestnené v jednej zložke. Ďalšou možnosťou je súbory *fxml* umiestniť do zložky *resources*, ale v tomto prípade musí byť dodržaná rovnaká štruktúra súborov ako v zložke, kde je umiestnená trieda v jazyku Java. View môžeme taktiež implementovať ako jednu triedu, ktorá má rovnakú funkciu ako v predchádzajúcom prípade, no naviac v nej musíme definovať všetky objekty, ktoré sme definovali v súbore *fxml*. To je neprehľadné a výsledná trieda bude obsahovať veľa kódu, v ktorom sa definuje vzhľad obrazovky.



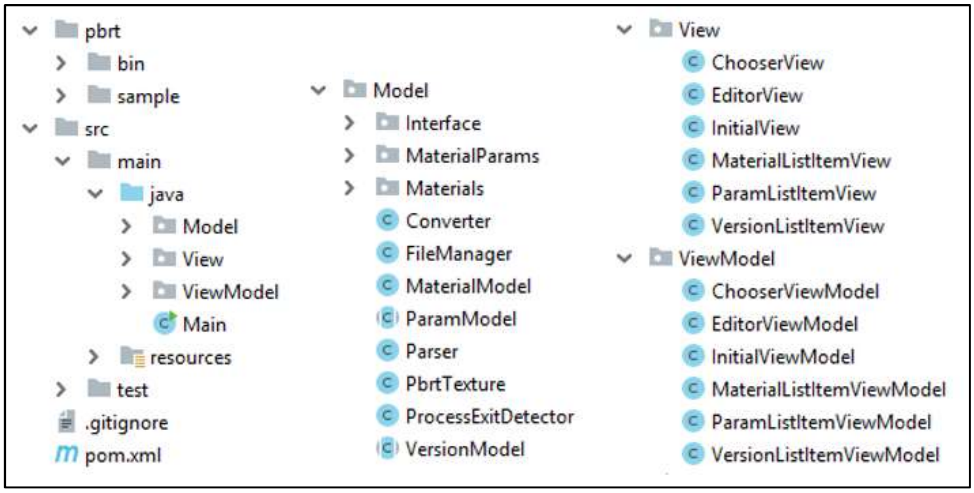

*Obrázok 17: Štruktúra View*

### 4.1.3 Úložisko

Ako úložisko sme použili databázu, ktorou je súbor vo formáte JSON[9]. Túto alternatívu sme zvolili pre jednoduchú prácu s týmto formátom a fakt, že tento formát dokáže niesť informácie, ktoré potrebujeme. Použitie zložitejšej databázy by mohlo zbytočne zaťažovať našu aplikáciu a pridávať jej ďalšie zbytočné závislosti. Všetky ukladané informácie sú v textovej podobe až na obrázky. Tie sú uložené na disku a v databáze môžeme nájsť cestu k nim. Obrázky a súbor, ktorý obsahuje databázu môžeme nájsť v zložke „pbrt“ v projekte.

[9] JavaScript Object Notation [14]

## 4.2 Štruktúra projektu




*Obrázok 18: Štruktúra projektu*

*Obrázok 18* znázorňuje štruktúru projektu. Projekt obsahuje zložku *pbrt*, v ktorej sa nachádza samotný renderer a ďalšie súbory, ktoré sú potrebné pre správny chod aplikácie. Môžeme tu nájsť súbor „.bat“, ktorým sa spúšťa renderovanie, súbor „materials“, ktorý je použitý ako databáza a v tejto zložke sú taktiež uložené vykreslené obrázky, ktoré sa používajú v aplikácií.

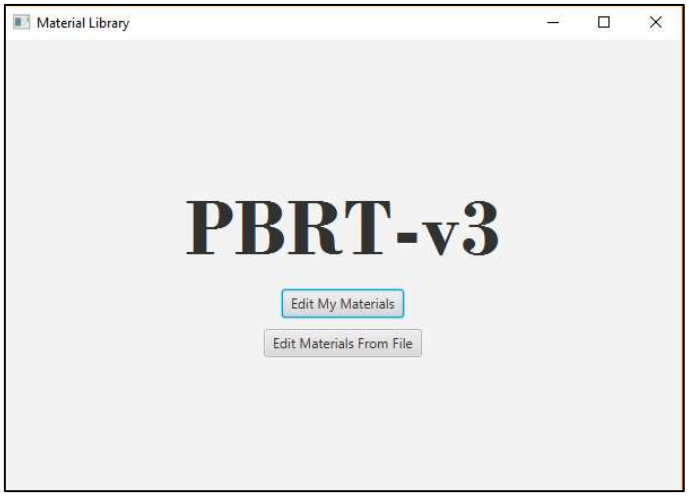
Zdrojový kód projektu sa nachádza v zložke „src“. Kód je rozdelený na zdroje a kód v jazyku Java. Zdroje obsahujú ikony použité v aplikácií a obrázky základných materiálov. Kód v Jave je rozdelený podľa architektúry na tri časti a to „Model“, „View“ a „ViewModel“. Môžeme vidieť triedu „Main“, tá však slúži iba na spúšťanie aplikácie. Model obsahuje základnú logiku aplikácie ako je konvertovanie a parsovanie či už formátu pbrt alebo JSON, ktorý sa používa ako databáza v súbore „materials“. Model ďalej obsahuje triedy na uchovávanie materiálov, verzií a parametrov. Každý typ materiálu má vlastnú triedu, ktorá obsahuje štandardné parametre materiálu a tieto triedy sú uložené v zložke „Materials“. Triedy parametrov sú uložené v zložke „MaterialParams“. Môžeme tu nájsť triedu „PbrtTexture“, ktorú predstavuje trieda, ktorá reprezentuje textúry a triedu „ProcessorExitDetector“, ktorá slúži na detekciu koncu procesu. Ako môžeme vidieť „View“ a „ViewModel“ obsahujú triedy s podobnými názvami, ktoré sa viažu ku konkrétnej obrazovke.

## 4.3 Funkcie aplikácie

Ako architektúru aplikácie sme zvolili MVVM, ktorý nám rozdeľuje aplikáciu na tri logické časti *Model*, *View* a *ViewModel*. *Model* obsahuje triedy, ktoré predstavujú materiály, verzie, parametre materiálov a triedy na prácu s rendererom a databázou. Časti *View* a *ViewModel* sú úzko prepojené

a môžeme tu rozlišovať dva druhy podľa ich obsahu. *View* a *ViewModel*, ktoré definujú jednotlivé obrazovky a *View* a *ViewModel*, ktoré definujú objekty použité v zoznamoch obrazoviek.

### 4.3.1 Úvodná obrazovka




*Obrázok 19: Úvodná obrazovka*

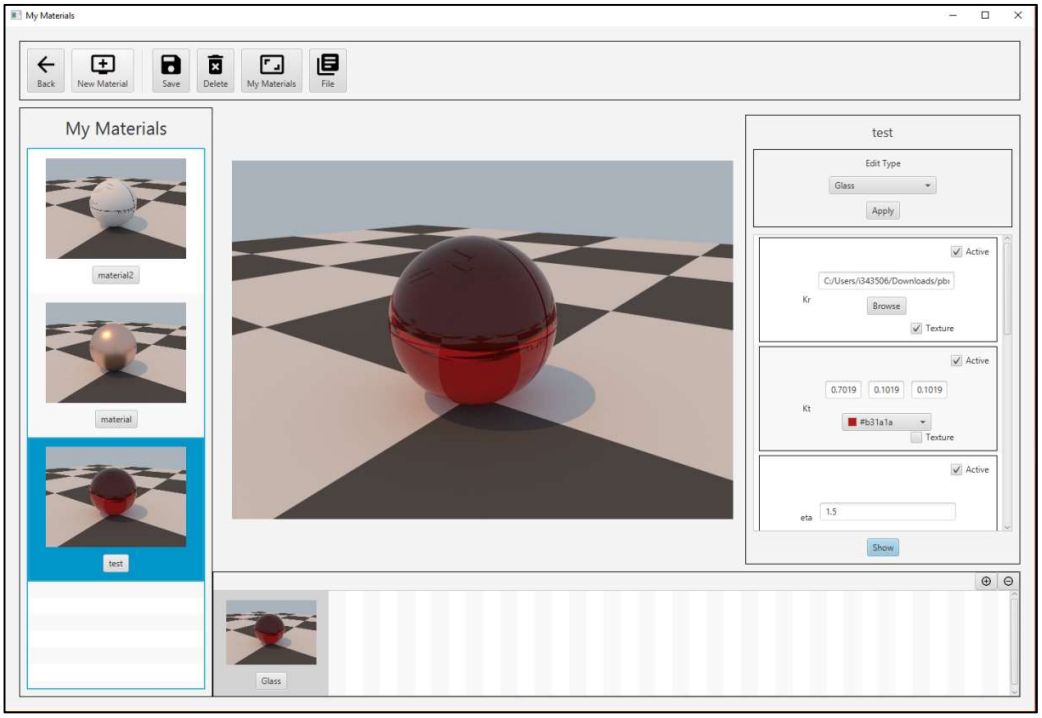
*Obrázok 19* znázorňuje úvodnú obrazovku aplikácie, ktorá slúži na vybranie zdroja materiálov, s ktorými chce užívateľ pracovať. Vizuálna stránka tejto obrazovky je definovaná vo View s názvom *InitialView* a logika je definovaná v triede *InitialViewModel*. Obrazovka sa zobrazí ako prvá po spustení aplikácie a pri jej inicializácií sa načítajú materiály z databázy a vytvorí sa hlavná obrazovka aplikácie *Editor*, ktorá sa zatiaľ nezobrazuje pretože nemá inicializované potrebné parametre. Načítanie materiálov je realizované triedou *Converter*, ktorá má za úlohu konverziu typu JSON na objekty a naopak. Trieda *InitialViewModel* obsahuje referenciu na konvertor, na ktorom sa zavolá metóda *getMaterialsFromJSON()*, ktorá z databázy načíta uložené materiály. Ako vidíme obrazovka obsahuje dve tlačidlá, a to tlačidlo „Edit My Materials“ a tlačidlo „Edit Materials From File“.

Prvé tlačidlo slúži na prácu s materiálmi, ktoré sa nachádzajú v databáze. Po kliknutí na toto tlačidlo vznikne udalosť, na ktorú zareaguje metóda *db()*, ktorá nastaví editoru materiály načítane z databázy, názov okna (v tomto prípade *My Materials*), nastaví editor na prácu s databázou a zobrazí okno editora.

Kliknutím na druhé tlačidlo chce užívateľ pracovať s materiálmi so súboru. Po kliknutí sa podobne ako pri tlačidle „Edit My Materials“ vytvorí udalosť, no na túto udalosť reaguje metóda *file()*, ktorá nastaví editor na prácu s materiálmi so súboru a vytvorí dialógové okno na výber súboru. Po vybraní súboru sa vytvorí nová instncia triedy *Parser*, ktorá berie zadaný súbor ako parameter. Následne sa skontroluje, či zadaný súbor existuje a či má potrebný formát. Keď prebehne kontrola úspešne, zavolá

sa metóda *getMaterials()*, ktorá zo zadaného súboru vyberie všetky materiály, tie sú nastavené editoru a zobrazí sa okno s editorom.

### 4.3.2 Editor




*Obrázok 20: Editor*

*Obrázok 20* zobrazuje hlavnú obrazovku aplikácie a to je *Editor*. Pred zobrazením tejto obrazovky sa z úvodnej obrazovky nastavia materiály, s ktorými sa bude pracovať. Tieto materiály sú zobrazené v zozname vľavo, no pred zobrazením je každý materiál zabalený do triedy *MaterialListItemViewModel*. Pri zabaľovaní sa zabalia aj verzie materiálu do triedy *VersionListItemViewModel* a podobne sa zabalia aj parametre všetkých verzií do triedy *ParamListItemViewModel*. Tieto triedy držia referenciu na objekt, ktorý zabaľujú a obsahujú atribúty typu *Property*, ktoré odpovedajú atribútom zabalenej instancie a sú naviazané na atribúty v triede *EditorView*. Každá trieda obsahuje metódu *save()*, ktorá slúži na ukladanie atribútov do referenčnej instancie a metódu *reloadFromModel()*, ktorá slúži na načítavanie atribútov z referenčnej instancie.

#### *4.3.2.1* Metóda *setMaterial()* a *setVersion()*

Metóda *setMaterial()* slúži na obnovenie vzhľadu editoru pri zmene vybraného materiálu. Táto metóda zmení názov, zoznam verzií a obrázok predošlého materiálu na aktuálne vybraný materiál, vyberie prvú verziu materiálu a zavolá metódu *setVersion()*, ktorá podobne aktualizuje atribúty verzie.

#### 4.3.2.2 Vytváranie materiálu

Vytváranie nového materiálu sa líši pri práci s databázou a pri práci so súborom. Po kliknutí na tlačidlo „New Material“ zobrazené pri práci s databázou alebo na tlačidlo „Add to My Materials“, ktoré je zobrazené v prípade, keď pracujeme so súborom sa vytvorí udalosť, na ktorú reaguje metóda *getNew()*. Táto metóda vytvorí dialógové okno, ktoré požaduje meno materiálu. Po zadaní mena sa funkcia líši vzhľadom na aktuálne používané materiály.

Pri práci s databázou sa vytvorí materiál typu *Disney*, ktorý sa pridá do zoznamu materiálov a následne sa označí. Ako posledné sa zavolá metóda *setMaterial 4.3.2.1*.

Pri práci so súborom nedáva zmysel vytvárať nový materiál, keďže súbor má presný počet materiálov. Preto sa vybraný materiál zo súboru uloží do databázy. Najprv sa materiálu nastaví meno, následne sa pridá medzi načítané súbory z databázy a na záver sa zavolá metóda *toJSONFile()* v triede *Converter*, ktorá berie ako parameter materiály, ktoré sú touto metódou uložené do databázy.

#### 4.3.2.3 Ukladanie materiálu

Pri kliknutí na tlačidlo „Save“ vzniká udalosť, ktorú odchytáva metóda *getSave()*. Táto metóda uloží vybraný materiál a vybranú verziu. Následne vytvorí nový materiál s touto verziou a do zoznamu pridá ostatné verzie, ktoré obsahoval zvolený materiál. Zvolený materiál vymaže a na jeho miesto vloží nový materiál, ktorý bol vytvorený. Nový materiál označí a zavolá metódu *setMaterial() 4.3.2.1*. Nakoniec metóda podľa zdroja materiálov uloží materiály. Pri práci s databázou sa zavolá metóda *toJSONFile()* z triedy *Converter*, ktorá uloží materiály do databázy a pri práci so súborom sa zavolá metóda *export()* z triedy *Parser*, ktorá uloží materiály do aktuálne otvoreného súboru. Obidve metódy berú ako parameter materiály, ktoré majú byť uložené.

#### 4.3.2.4 Mazanie materiálu

Tlačidlom „Delete“ vymažeme zvolený materiál aj s jeho verziami. Po kliknutí na tlačidlo sa vytvorí udalosť, na ktorú reaguje metóda *getDelete(). Metóda* najprv skontroluje počet materiálov, ktorý musí byť väčší ako 1 aby bolo možné vymazať materiál a potom zobrazí dialógové okno s potvrdením, či chce užívateľ naozaj vymazať vybraný materiál. Po potvrdení je tento materiál vymazaný zo zoznamu, vyberie sa materiál, ktorý bol vyššie od vymazaného materiálu a zavolá sa metóda *setMaterial() 4.3.2.1*. Materiál stále nie je vymazaný z databázy a preto, ak chceme materiál vymazať definitívne musíme uložiť zmeny použitím tlačidla „Save“.

#### 4.3.2.5 Zmena typu materiálu

Zmena typu je realizovaná pomocou rozvíjajúceho zoznamu a následnom kliknutí na tlačidlo „Apply“, ktoré vytvorí udalosť a na ňu reaguje metóda *getTypeChoose()*. Následne sa vytvorí nová verzia

s príslušným typom a štandardnými parametrami, ktorú môžeme ďalej upravovať. Táto verzia sa označí a zavolá sa metóda *setVersion() 4.3.2.1*.

#### 4.3.2.6 Zobrazenie materiálu

Na zobrazenie materiálu slúži tlačidlo „Show“. Toto tlačidlo vyvoláva udalosť, ktorá je zachytená metódou *getShow()*, ktorá uloží vybraný materiál a vybranú verziu a zavolá metódu *createSmallImage()*. Táto metóda sa nachádza v triede *FileManager* a slúži na vytvorenie vstupného súboru pre renderer a na jeho spustenie. Vykresľovanie sa skladá z dvoch etáp. Prvá etapa vykreslí malý obrázok, ktorý je vykreslený veľmi rýchlo, no nemá dostačujúcu kvalitu. Druhá etapa vykreslí veľký obrázok, čo zaberie viac času, ale obrázok má lepšiu kvalitu a je možné lepšie vidieť detaily materiálu. Každé vykreslenie prebieha vo vlastnom vlákne, takže aplikácia je aj počas vykresľovania funkčná. Po dokončení vykresľovania sa zobrazí obrázok v editore a z disku sa vymažú nepotrebné súbory, ako malý obrázok a súbory s príponou *pbrt*.

#### 4.3.2.7 Práca s verziami

Vytvárať nové verzie môžeme troma spôsobmi. Prvý, je popísaný kapitole v *4.3.2.5*, druhým spôsobom je zvolenie tlačidla „+“, čím sa vytvorí udalosť, na ktorú reaguje metóda *getAdd()*. Metóda vytvorí novú verziu na základe typu poslednej zvolenej verzie. Posledný spôsob je, že sa vyberie materiál z iného súboru alebo z databázy a to pomocou tlačidiel na prehľadávanie iných zdrojov materiálov v hornej časti editoru. Po pridaní novej verzie sa táto verzia označí a zavolá sa metóda *setVersion() 4.3.2.1*.

Vymazanie verzie je realizované pomocou tlačidla „-“, ktoré však funguje len v prípade, keď je počet verzií väčší ako 1 a udalosť tohto tlačidla je zachytávaná metódou *getRemove()*. Táto metóda skontroluje počet verzií, pretože materiál musí mať minimálne jednu verziu a taktiež index verzie, pretože prvá verzia nemôže byť vymazaná. Ak sú podmienky splnené zvolená verzia sa vymaže zo zoznamu, označí sa predošlá verzia a zavolá sa metóda *setVersion() 4.3.2.1*. Pri práci s verziami nemôžeme zabúdať, že sa pri stlačení tlačidla „Save“ sa uloží vybraná verzia ako hlavná a materiál sa zmení na zvolenú verziu.

#### 4.3.2.8 Zmena názvu materiálu a verzie

Zmena názvu materiálu alebo verzie je veľmi jednoduchá. Slúži na to tlačidlo s názvom materiálu alebo verzie v zozname materiálov a verzií. Tieto tlačidla vytvárajú udalosti, na ktoré reaguje metóda *onChangeName()*. Metóda zobrazí dialógové okno s textovým poľom, ktoré obsahuje aktuálne meno. Toto meno je možné zmeniť a po potvrdení sa zmení názov materiálu alebo verzie.

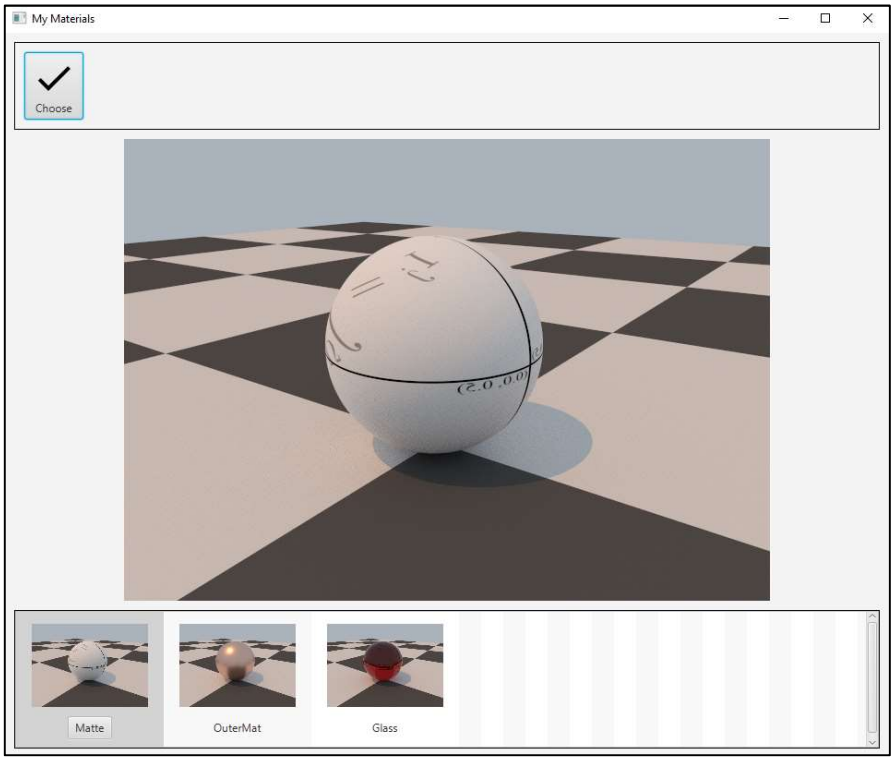
#### 4.3.2.9 Výber iných materiálov

Na výber iných materiálov slúžia tlačidlá „My Materials“ a „File“, ktoré sa nachádzajú v hornej lište.

Pre výber materiálov z databázy je potrebné zvoliť prvé tlačidlo, ktoré vytvára udalosť, ktorú zachytáva metóda *getFromDb()*. Metóda nastaví názov (v tomto prípade My Materials) novému oknu a zavolá metódu *getMaterialsFromJSON()*, ktorá vráti materiály z databázy. Následne materiály zabalí do triedy *VersionListItemViewModel* a zobrazí okno na výber materiálov, ktorému nastaví tieto materiály na zobrazenie.

Pre výber materiálov z iného súboru je potrebné kliknúť na druhé tlačidlo. To vyvolá udalosť, na ktorú reaguje metóda *getFromFile()*, ktorá vytvorí dialógové okno na výber súboru, z ktorého budú načítané materiály. Nastaví sa názov okna a vytvorí sa instancia triedy *Parser* so zadaným súborom, kde prebehne kontrola na existenciu súboru a správnosť formátu. Následne sú pomocou parseru načítané materiály zo súboru. Tieto materiály musia byť pred zobrazením vykreslené, pretože nie sú uložené v databáze a tým pádom nemáme uložené ich obrázky. Vykreslenie je realizované metódou *createImages()*, ktorá sa nachádza v triede *FileManager* a táto metóda vytvorí vstupné súbory pre renderer a následne ho spustí pre všetky materiály načítané zo súboru. Každé vykresľovanie prebieha v dvoch etapách ako v kapitole *4.3.2.6*. Materiály sa potom zabalia do triedy *VersionListItemViewModel* a zobrazí sa nová obrazovka.

### 4.3.3 Chooser




*Obrázok 21: Chooser*

*Obrázok 21* ilustruje poslednú obrazovku, ktorá slúži na výber materiálov z iných zdrojov. Obrazovke je pri inicializácii nastavený zoznam materiálov, ktoré sú už zabalené a zobrazia sa v zozname, ktorý sa nachádza v spodnej časti obrazovky. Obrazovka obsahuje jedno tlačidlo „Choose“, ktoré pri vyvolaní udalosti spustí metódu *getChoose()*, ktorá vybraný materiál pridá do verzií materiálu, ktorý je aktuálne zvolený v editore a následne zatvorí okno.

## 4.4 Testovanie

Na záver implementácie prebehlo užívateľské testovanie, ktoré malo odhaliť nedostatky v rozhraní a chyby v aplikácií. Testovanie prebehlo podobne ako v časti *3.4 Testovanie* s troma účastníkmi a boli použité rovnaké úlohy. Účastníci dostávali úlohy, ktoré pri plnení komentovali a počas plnenia dostávali doplňujúce otázky ohľadom práce s užívateľským rozhraním.

### 4.4.1 Hodnotenie

Úloha 1

- Vytvorte ružový matný materiál

Účastníci videli aplikáciu prvý krát, čomu nasvedčovala mierna neistota pri práci na úlohe, no táto úloha bola veľmi jednoduchá a preto bola vyriešená pomerne rýchlo a bez problémov.

Úloha 2

- Vytvorte zelenú verziu materiálu z úlohy 1

Úloha bola riešiteľná dvoma spôsobmi a to použitím tlačidla na pridávanie verzií alebo použitím tlačidla na aplikovanie typu materiálu, čím sa vytvorí nová verzia. Všetci užívatelia použili prvý spôsob, ktorý je intuitívnejší pri práci s verziami, pretože na pridávanie verzií je nad panelom verzií.

Úloha 3

- Uložte ružovú verziu z úlohy 1 a zelenú verziu z úlohy 2 odstráňte

Pri tejto úlohe si jeden z účastníkov nebol istý ako uložiť verziu, pretože tlačidlo „Save“ sa nachádza v hornej časti obrazovky. Následne odstránenie verzie už účastníkom nerobilo problémy.

Úloha 4

- Upravte ružový materiál na matný biely materiál a materiál uložte

Táto úloha bola vyriešená veľmi rýchlo a účastníci nenarazili na žiadne prekážky pri jej plnení.

Úloha 5

- Načítajte materiály zo scény „lte-orb-roughglass“, ktorá je uložená na disku

Pri tejto úlohe bolo potrebné opäť otvoriť úvodnú obrazovku a vybrať súbor pomocou prehliadača. Tu všetci účastníci narazili na mierny nedostatok, a to, že užívateľ nie je dostatočne informovaný o zdroji materiálov, ktoré sa momentálne zobrazujú v editore. Po preskúmaní a vyhodnotení tohto problému sme do aplikácie pridali nové pole, ktoré zobrazuje názov súboru, z ktorého sú načítané aktuálne materiály.

Úloha 6

- Uložte modrý materiál zo súboru z úlohy 5 do databázy

V tejto úlohe sme museli účastníkom vysvetliť, že pri práci zo súborom nie je možné vytvárať nové materiály, no vybraný materiál je uložený do databázy. Toto chovanie je dané z logiky vytvárania scény a nie je možné ho zmeniť. Po vysvetlení účastníci túto úlohu zvládli.

Úloha 7

- Vytvorte novú verziu materiálu z úlohy 6, ktorá je načítaná z materiálu v kroku 4 a uložte zmeny

Táto úloha je zložitejšia a účastníci sa opäť sťažovali na nedostatočnú transparentnosť zdroja materiálov. Bolo nutné použiť obrazovku na výber materiálov, ktorú účastníci ešte nepoužili, no po miernom nasmerovaní všetci účastníci splnili zadanie.

Úloha 8

- Vyberte druhý materiál zo scény v kroku 5

V tejto úlohe nebol odhalený žiaden nedostatok.

Úloha 9

- Načítajte materiály zo scény v súbore „lte-orb-silver“ z disku a materiál z úlohy 8 nahraďte modrým materiálom z tejto scény

Keďže táto úloha je opäť zložitejšia a je potrebné pracovať s dvoma scénami naraz, boli účastníci mierne nasmerovaní k správnemu postupu. Tento postup nakoniec zhodnotili ako jednoduchý a odporučili ho ponechať v tomto tvare.

Úloha 10

- Otvorte materiály z databázy a vymažte materiál z úlohy 4

V tejto úlohe jeden účastník našiel nedostatok, že po vymazaní materiálu je potrebné uložiť zmeny aby sa materiál pri reštarte aplikácie opäť nezobrazoval v zozname. Po rozbore tohto nedostatku sme sa rozhodli ponechať funkcionalitu nezmenenú.

# 5 Záver

V tejto práci sme sa zaoberali návrhom a implementáciou editoru materiálov pre renderer PBRT, ktorý v súčasnosti nemá vlastný editor, no existujú rozšírenia do iných editorov, ktoré podporujú PBRT. Do editorov *Autodesk Maya* a *Blender* takéto rozšírenia môžeme stiahnuť a pracovať v nich aj s PBRT. Pre vývoj editoru materiálov pre PBRT sme sa rozhodli hlavne kvôli zložitej editácií textových súboroch a pomalému vykresľovaniu zložitých scén.

Na začiatku sme si ukázali textový súbor PBRT a vykreslený obrázok týmto rendererom. Ďalej sme popísali prácu s PBRT bez editora a definovali funkčné a nefunkčné požiadavky na editor. Následne sme začali s výberom úložiska, kde sme zvolili kombináciu databázy s ukladaním priamo na disk. Údaje o materiáloch budú uložené v databáze a obrázky na disku. Keď sme mali vybrané úložisko mohli sme začať vytvárať koncept editoru, kde sme definovali proces vytvárania obrázkov pomocou editoru a stavy materiálu stavovým diagramom. Potom sme na základe funkčných a nefunkčných požiadaviek mohli vytvoriť prototyp editoru. Prototyp pozostáva z troch obrazoviek, ktoré na seba nadväzujú a určujú s akým typom úložiska užívateľ v danej chvíli pracuje. Užívateľ môže plynule striedať medzi databázou a materiálmi so súboru. Tieto materiály môže jednoducho vytvárať, ukladať, mazať alebo kopírovať. Taktiež môže využívať verzovací systém, kde môže ľubovoľne pracovať s verziami a kedykoľvek zvoliť danú verziu ako originálny materiál alebo ju vymazať. Na záver sme prototyp podrobili testovaniu, ktoré splnilo svoju úlohu a našli sme malé nedostatky v prototype. Tieto drobné chybičky sme upravili a vytvorili nový prototyp, ktorý slúžil ako základ aplikácie, ktorú sme v ďalšej časti vytvárali.

Kapitolu implementácia sme začali výberom technológií, ktoré budeme v aplikácií používať. Pri tomto výbere sme museli dávať pozor na kompatibilitu jednotlivých častí a výber technológií, ktoré spĺňajú naše požiadavky. Ako návrhový vzor aplikácie sme zvolili vzor MVVM, ktorý rozdeľuje aplikáciu na tri logické časti Model, View a ViewModel. Časti View a ViewModel sú prepojené pomocou viazania dát, čo je charakteristické pre MVVM a týmito väzbami sa zmenšuje veľkosť kódu aplikácie. V závislosti na návrhovom vzore a vybranom jazyku sme zvolili framework, ktorý bude podporovať MVVM a bude kompatibilný s vybraným jazykom, ktorým je Java. Vybrali sme framework **mvvm(FX)**, ktorý je však kompatibilný len s verziou Java 8, ktorú sme nakoniec použili. Ako úložisko sme zvolili textový súbor, v ktorom sú uložené informácie o materiáloch vo formáte *JSON*. Toto úložisko je najjednoduchšia voľba, pretože vzhľadom na funkcie aplikácie nie je potrebné implementovať zložitejšiu databázu. Po výbere technológií sme mohli začať implementovať a vytvorili sme projekt, ktorého štruktúru sme ďalej opisovali. Následne sme popísali funkcie aplikácie. V tejto časti sme predstavili jednotlivé obrazovky aplikácie a popísali sme ich funkcionalitu a naznačili implementáciu. Na záver sme

podrobili aplikáciu testovaniu, ktoré prebehlo s tromi účastníkmi. Účastníkom bolo postupne zadaných desať úloh, ktoré mali splniť. Počas plnenia úloh svoje kroky komentovali a v prípade, že nevedeli ako pokračovať boli nasmerovaní k ďalšiemu postupu. Výsledky testovania sme zhodnotili a upravili jeden nedostatok, ktorý našli všetci užívatelia.

Na záver môžeme zhodnotiť, že sme vytvorili funkčnú aplikáciu, ktorá slúži ako knižnica materiálov pre PBRT-v3 renderer. Pri ďalšom rozširovaní aplikácie by som navrhoval použitie robustnejšieho úložiska, čo by mohlo viesť k vytvoreniu aplikácie, kde každý užívateľ bude mať uložené materiály na serveri, ku ktorým bude mať prístup a bude s nimi môcť pracovať. Ďalej by som navrhoval upraviť vykresľovanie obrázkov, pretože momentálne sa po spustení rendereru zobrazí konzolové okno, v ktorom môže užívateľ vidieť aktuálny stav vykresľovania, čo pri testovaní nevadilo, no z estetického hľadiska by to malo byť upravené.

Táto aplikácia určite nemôže konkurovať profesionálnym editorom ako je *Autodesk Maya* alebo *Blender*, no môže poslúžiť ako odrazový mostík pre vývoj originálneho editoru pre PBRT renderer, čo by určite potešilo priaznivcov tohto rendereru.

# Literatúra


[1] Pbrt-v3 [online]. Matt Pharr, Wenzel Jakob, Greg Humphreys, 2015. © 2019 GitHub, Inc. [22.10.2018]. Dostupné z: https://github.com/mmp/pbrt-v3/

[2] Matt Pharr, Wenzel Jakob, Greg Humphreys. Physically Based Rendering: From Theory To Implementation [online]. 3. vydanie. Copyright: © Morgan Kaufmann, 2017 [15.03.2019]. Dostupné z: http://www.pbr-book.org/.

[3] Pbrt-v3 – Fileformat [online]. Matt Pharr, Wenzel Jakob, Greg Humphreys, Copyright© 2004-2018 Matt Pharr, Wenzel Jakob, and Greg Humphreys. All rights reserved. [10.11.2018]. Dostupné z: https://www.pbrt.org/fileformat-v3.html

[4] Autodesk Maya [online]. [15.12.2018]. © 2019 Autodesk Inc. All rights reserved. Dostupné z: https://www.autodesk.com/products/maya/overview

[5] Blender [online]. The Free and Open Source 3D Creation Suite. [15.12.2018]. Dostupné z: https://www.blender.org/

[6] 3D modeling: Blender Vs. Maya [online]. Jordan Charlton, 2013. [05.01.2019]. Dostupné z: https://www.cia.edu/blog/2013/11/3d-modeling-blender-vs-maya

[7] Moqups [online]. © 2019 - S.C Evercoder Software S.R.L. [15.01.2019]. Dostupné z: https://moqups.com/

[8] Low-fi prototyping: What, Why and How? [online]. 2016. © Copyright MOBGEN. [17.12.2018]. Dostupné z: https://www.mobgen.com/low-fi-prototyping/

[9] What is WPF? [online]. © wpf-tutorial.com 2007-2019. [15.02.2019]. Dostupné z: https://wpf-tutorial.com/about-wpf/what-is-wpf/

[10] JavaFX [online]. Monica Pawlan, 2013. Copyright © 2008, 2013, Oracle and/or its affiliates. All rights reserved. [15.02.2019]. Dostupné z: https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

[11] Apache Maven [online]. Copyright ©2002–2019 The Apache Software Foundation. All rights reserved. [16.02.2019]. Dostupné z: https://maven.apache.org/

[12] A Journey through the Land of Model-View-Design Patterns [online]. Artem Syromiatnikov, Danny Weyns, 2014. [17.02.2019]. Dostupné z: https://ieeexplore.ieee.org/document/6827095

[13] MVVM for JavaFX [online]. 2017. © 2019 GitHub, Inc. [20.02.2019]. Dostupné z: https://github.com/sialcasa/mvvmFX/wiki

[14] Introducing JSON [online]. [25.2.2019]. Dostupné z: https://www.json.org/

# Príloha A

## Zip Archív – zdrojové kódy aplikácie

Obsah zip archívu:

- /Src – zdrojový kód
- /pbrt – pbrt rendered, súbory na prácu s rendererom (databáza, súbor pre unittesty)
- pom.xml – súbor na správu balíčkov (Apache Maven)
- readme.txt – návod na spustenie aplikácie

# Príloha B

## Zip Archív – Spustiteľná aplikácia

Obsah archívu:

- /textures – textúra
- /pbrt_files – súbory pbrt, kde sú nadefinované scény
- /pbrt - pbrt rendered, súbory na prácu s rendererom (databáza, úložisko pre vygenerované obrázky)
- /bsdfs – bsdfs súbor
- pbrt_library.jar – spustiteľná aplikácia
- readme.txt – návod na spustenie aplikácie