FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Import a vizualizace BIM modelů v Unreal Engine s přiřazením textur
Student:	Bc. Tadeáš Bušek
Vedoucí:	Ing. Petr Pauš, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Stavební informační modely (soubory typu BIM) jsou digitální reprezentace fyzických a funkčních vlastností staveb. Cílem této diplomové práce je vytvořit projekt v Unreal Engine, který umožní import BIM souborů, přiřadí vhodné textury a nastaví potřebné parametry pro importované modely. Díky tomu bude možné v prostředí Unreal Engine BIM modely vizualizovat s věrnými texturami i detaily a následně jimi procházet.

Pokyny:
1. Proveďte důkladnou analýzu formátu BIM. Zjistěte, jaké informace o struktuře budov a prvcích jsou v tomto formátu uloženy.
2. Prozkoumejte a analyzujte možnosti Unreal Engine pro import BIM souborů. Zjistěte, jakými způsoby lze z formátu BIM načíst a interpretovat uložená data.
3. Proveďte podrobnou analýzu možností přiřazení a získání textur v rámci projektu v Unreal Engine. Zvažte možnost generování a automatické přiřazování textur s využitím umělé inteligence (AI).
4. Pomocí analýzy a metod softwarového inženýrství navrhněte Unreal Engine projekt, který import BIM modelů s přiřazením textur umožní.
5. Podle návrhu implementujte import BIM a automatizaci přiřazení textur pro více realistickou vizualizaci. Též implementuje možnost procházení importované budovy z pohledu první osoby.
6. Proveďte vhodná testování projektu na dodaných vzorových souborech

---

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 2. června 2023 v Praze.

Diplomová práce

IMPORT A VIZUALIZACE BIM MODELŮ V UNREAL ENGINE S PŘIŘAZENÍM TEXTUR

Bc. Tadeáš Bušek

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Petr Pauš, Ph.D.
9. května 2024

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych vyjádřit své upřímné poděkování vedoucímu mé diplomové práce, Ing. Petrovi Paušovi, Ph.D., za jeho cenné odborné názory, vedení a pravidelné konzultace po celou dobu tvorby této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 9. května 2024

Abstrakt

Tato diplomová práce se zabývá implementací a vizualizací Building Information Modeling (BIM) modelů v Unreal Engine 5. Hlavním cílem práce bylo umožnit společnosti T-SOFT a.s. efektivně importovat BIM modely do Unreal Engine, přičemž byl kladen důraz na automatizaci procesů přiřazení kolizí, světel a materiálů. Práce zahrnuje důkladnou analýzu BIM, formátu IFC a možností, které Unreal Engine 5 nabízí. Dále práce obsahuje návrh uživatelského rozhraní společně s návrhem řešení založeného na analýze. Výstupem práce je popis realizace a funkční prototyp pro import a vizualizaci BIM v Unreal Engine 5.

Výsledky této práce poskytují společnosti T-SOFT a.s. pevný základ pro využívání Unreal Engine 5 k vizualizaci a dynamickým simulacím za využití dat z BIM.

Klíčová slova BIM, IFC, Unreal Engine, Import 3D modelů, Datasmith, Vizualizace 3D modelů, Přiřazení materiálů, Přiřazení kolizí

Abstract

This thesis focuses on the implementation and visualization of Building Information Modeling (BIM) models in Unreal Engine 5. The main goal of the thesis was to enable T-SOFT a.s. to efficiently import BIM models into Unreal Engine, emphasizing the automation of processes for assigning collisions, lights, and materials. The work includes a thorough analysis of BIM, the IFC format, and the capabilities that Unreal Engine 5 offers. Furthermore, the thesis contains a design of the user interface along with a solution design based on the analysis. The output of the work is a description of the implementation and a functional prototype for importing and visualizing BIM in Unreal Engine 5.

The results of this work provide T-SOFT a.s. with a solid foundation for using Unreal Engine 5 for visualization and dynamic simulations utilizing BIM data.

Keywords BIM, IFC, Unreal Engine, Import of 3D models, Datasmith, Visualization of 3D models, Asigning of materials, Assigning of collisions

Seznam zkratek

BIM	Building Information Modeling
UE	Unreal Engine
UI	User Interface
OS	Operating System
COBie	Construction Operations Building Information Exchange
BOS	BIM Object Standard
IFC	Industry Foundation Classes
MVD	Model View Definition
FBX	FilmBox
IBOS	International BIM Object Standard
OBOS	Open BIM Object Standard
CAD	Computer-aided Design
MIT	Massachusetts Institute of Technology
IFC STEP	Standard for the Exchange of Product Model Data
HLSL	High Level Shading Language
SVM	Support Vector Machines
DDS	DirectDraw Surface
LGPL	Lesser General Public License

Introduction

Unreal Engine 5, který byl původně vznikl jako herní engine, se v posledních letech vyvinul do velice rozsáhlé aplikace s velkým množstvím funkcí. Dnes se tento engine používá nejen k vývoji her, ale také v oblastech jako jsou simulace a architektonické vizualizace. Tento rozšířený záběr otevírá nové možnosti pro aplikace v průmyslových a komerčních sektorech.

Společnost T-SOFT a.s. se rozhodla využít Unreal Engine 5 pro realizaci různých typů simulací, jako je například simulace pohybu lidí při vzniku požáru. K efektivnímu provádění těchto simulací je nezbytné disponovat rozsáhlým množstvím informací o budovách, které jsou součástí modelů BIM (Building Information Modeling). Aby bylo možné tyto informace v Unreal Engine využít, je nutné do něj importovat nejen geometrická data, ale i detailní informace obsažené v BIM modelech. Import BIM souborů nejen umožňuje přesné a realistické zobrazení architektonických struktur, ale také je základem pro aplikaci dynamických simulací v připravených modelech.

Klíčovým aspektem využití Unreal Engine pro tyto účely je tedy import modelů budov a následná automatizace vizualizace modelů. Objektů v budově může být vysoké množství, a proto automatizace eliminuje potřebu ručního zpracování a výrazně zvyšuje efektivitu celého procesu. Dalším důležitým krokem je příprava modelů pro simulaci průchodu osob, což vyžaduje specifické úpravy jako je například generování kolizí.

Z těchto potřeb a výzev vyplynulo téma mé diplomové práce. Práce se zaměřuje na zkoumání BIM modelů, analyzování možností Unreal Engine pro import těchto modelů a jejich následnou vizualizaci. Cílem práce je vytvořit prototyp, který prozkoumá klíčové funkcionality Unreal Engine, a jejich následné možné využití pro simulace. Tento prototyp by měl sloužit jako základ pro další vývoj a naplnění cíle T-SOFT a.s. plně využít potenciál moderních vizualizačních a simulačních technologií.

Kapitola 1
Analýza

1.1 BIM – Úvod

Abychom mohli podrobněji rozebrat strukturu, využití a vytváření BIM, musíme si nejprve definovat, co BIM vlastně znamená. BIM, zkratka pro Building Information Modeling, je spíše než konkrétní formát 3D modelů specifikace pro vytváření modelů. Tato specifikace nám popisuje nový myšlenkový směr, jak koukat na stavební konstrukce v digitální podobě a jak je také vytvářet. Samozřejmě, je také nezbytná existence formátů, které tento způsob modelování podporují, ale na tuto problematiku se podrobněji podíváme v následující sekci. 1.5

Protože Building Information Modeling nemá žádnou pevnou definici, podíváme se na definice ze zdrojů a z těch pak zkonstruujeme definici platnou pro tuto práci. První definice je podle konstrukční firmy M.A. Mortenson Company, která říká:

> *„BIM má své kořeny již v počítačem podporovaném výzkumu designu z předchozích desetiletí, ale stále nemá jedinou, široce přijímanou definici. My v společnosti M.A. Mortenson jej chápeme jako inteligentní simulaci architektury. Abychom mohli dosáhnout integrované dodávky, musí tato simulace vykazovat šest klíčových charakteristik. Musí být:*
>
> - *Digitální,*
> - *Prostorová (3D),*
> - *Měřitelná (kvantifikovatelná, dimenzovatelná a dotazovatelná),*
> - *Komplexní (zahrnující a komunikující s designovým záměrem, výkonností budovy, proveditelností a zahrnující postupné a finanční aspekty prostředků a metod),*
> - *Přístupná (celému týmu AEC/vlastníků prostřednictvím interoperabilního a intuitivního rozhraní) a*

- *Odolná (použitelná ve všech fázích životního cyklu zařízení).*

“

[1]

Tato definice nám zadává parametry, které by měla každá technologie Building Information Modeling splňovat, ale k většímu přiblížení si dovolím citovat ještě jednu definici od Patrika Suermana, zaměstnance NBIMS (National Building Information Modeling Standard):

> *„BIM je virtuální reprezentace fyzických a funkčních charakteristik zařízení od jeho vzniku až do konce jeho životnosti. Jako taková slouží jako sdílené úložiště informací pro spolupráci během celého životního cyklu zařízení.“*

[2]

Po spojení těchto dvou definic si můžeme definovat BIM jako Inteligentní simulaci architektury budovy, či jiného fyzického prvku, která ve své digitální podobě obsahuje virtuální reprezentaci jak fyzických, tak funkčních prvků, a to nejen v přítomnosti, ale snaží se zachytit celý životní cyklus, to jest změny, které byly provedeny, kým byly provedeny a tak dále.

Jakožto fyzické prvky Building Information Modeling si pak můžeme představit například 3D geometrii budovy, kterou obsahují i tradiční formáty a způsoby modelování. Building Informtaion Modeling se však snaží sjednotit vše dohromady, a proto umožňuje i přidávání funkčních prvků. Mezi funkční prvky pak patří například materiál, ze kterého je fyzický prvek vytvořen, různé vlastnosti fyzického prvku, ale například i historie daného prvku, ať už se týká jeho vývoje a změn, či evidence, kdo dané změny provedl. Analýze funkčních prvků, se kterými se můžeme setkat, se budeme věnovat později.

1.2 BIM – Historie účel

K pochopení důvodu vzniku Building Information Modeling se velmi stručně podíváme i do historie.

Historicky se používal k návrhům konstrukcí staveb hlavně papír a nákresy, do tohoto stavu zanesl revoluci CAD, zkratka pro Computer Aided Design, který se snažil tyto nákresy digitalizovat, a tak ulehčit jejich vytváření a komunikaci, stále se však jednalo pouze o nákresy a z většiny 2D grafiku. Do toho přišla další revoluce v podobě Building Information Modeling.

Building Information Modeling se narozdíl od svých předchůdců snažil o celkovou virtualizaci budovy, aby se nejednalo pouze o nákresy, ale o realistický 3D model budovy, zachycený i se všemi procesy a užitečnými dodatečnými informacemi. Proto technologie pro Building Information Modeling vznikly jako kombinace 3D geometrie a informační databáze, tyto technologie se však stále

vyvíjí a nyní se například ve většině programů podporující Building Information Modeling používá parametrické modelování. Parametrické modelování je vytváření 3D geometrie za pomoci nastavení parametrů a vztahů, které slouží jako pravidla. Tato pravidla pak sama upravují model při změnách, či hlídají jejich porušení a varují o tom uživatele (to může sloužit při hlídání požadavků na reálnou stavbu).

Všechny standardy a vývoj se pak snaží hlídat a slučovat organizace NBIMS (National Building Information Modeling Standard) nebo také organizace BuildingSMART. Všechen vývoj má pak za cíl realistickou virtualizaci plánované konstrukční budovy, aby došlo k odhalení všech problémů již v počátku plánování, urychlil se návrh a došlo k ušetření nákladů při samotné výstavbě, či došlo k ukončení projektu v rané fázi při příliš vysokých nákladech.

1.3 BIM -Standardy a vlastnosti

Abychom věděli, s jakými daty se můžeme setkat a počítat, tak se v nynější sekci zaměříme na standardy Building Information Modeling s důrazem na vlastnosti, které by podle standardů měl produkt Building Information Modeling obsahovat. Informace do této sekce jsem čerpal převážně z International BIM Object standard (IBOS) [3, 4] ale i z Open BIM Object standard (OBOS) [5]. Protože jsou Building Information Modeling standardy úzce spojené s IFC, čerpal jsem i z IFC standardů od BuildingSMART [6]. Mimo vlastnosti zahrnuté v IFC standardech, může BIM obsahovat i vlastnosti nabízející technologií COBie, proto se stručně zaměříme i na standard COBie.

Standardy pro Building Information Modeling, ať už se jedná o IBOS, či OBOS, říkají informace, které se převážně shodují. Mezi ty pro nás důležité patří například konvence pojmenování, kdy jména by měla být oddělená podtržítkem a pokud se jedná o objekt s rodičem, měl by být rodič zmíněn v prefixu. Konkrétně jména pro BIM objekty, materiály, či přidružené obrázky musí obsahovat části jako: Autor, Zdroj, Typ, Materiál, Podtyp/Kód produktu, Rozlišovač.

Autor a Zdroj popisují, odkud daný objekt pochází (neboli z jaké knihovny je objekt, který používáme a chceme pojmenovat) a poskytovatele použité knihovny, kdy poskytovatel se uvádí kódem obsahujícím tři až šest znaků. Typ říká, s jakým druhem objektu pracujeme. Příkladem mohou být typ Dveře. Materiál je složení objektu. Podtyp/Kód produktu doplňuje dodatečné informace. A v poslední řadě Rozlišovač pak doplňuje informace k jednodušší identifikaci. [3]

Zde jsou uvedeny konkrétní příklady standardů:

> *„ Názvy souborů a BIM objektů by měly být složeny z polí, v následujícím uspořádání:*
> *<Autor>_<Zdroj>_<Typ>_<Podtyp/Kód produktu>_<Rozlišovač1>“*

„ POZNÁMKA 1: Pole podtypu a rozlišovače jsou volitelná a mohou být zařazena podle potřeby. “

„ Název materiálu by měl být složen z polí složen z polí, v následujícím uspořádání:
<Autor>_<Zdroj>_<Materiál>_<Podtyp1>_<Rozlišovač1>“

„ POZNÁMKA 1: Pole podtypu a rozlišovače jsou volitelná a mohou být zařazena podle potřeby. “

[3]

Standardem v oblasti geometrie je použití 3D objektů, které se dělí na dvě skupiny. První skupinou jsou modely, které jsou modelovány uživatelem jako nový objekt s určitým účelem sloužící jakožto komponent do celku. Druhým typem je 3D objekt nějakého typu, který již v Building Information Modeling architektuře existuje. Následně se objekty, pokud k tomu problematika vybízí, mohou shromažďovat do takzvaných assembly. Assembly je součástka skládající se ze základních 3D objektů, umožňující jejich přepoužití později. Celá geometrie se pak skládá z částí zmíněných výše. [5]

Dalším důležitým standardem týkajícím se geometrie je použití jednotek. Jednotky by měly být vždy v metrickém systému v milimetrech, až na výjimečné speciální případy. Geometrie objektů má podle standardu pak odpovídat velikostem jedna ku jedné s reálným světem. [4]

Pro nás nejdůležitějším standardem, abychom věděli, co ve struktuře modelů BIM očekávat, jsou však vlastnosti. Vlastnosti jsou důležité pro poskytování dodatečných informací o objektech zahrnutých v BIM. Jak je zmíněno v OBOS

„Vlastnosti mohou být buď vloženy přímo do objektu nebo propojeny pomocí jedinečného odkazu s externí databází.“

[5]

To znamená, že přiřazené vlastnosti nemusí být přímo v samotné architektuře, ale i v externím zdroji. Pokud je použit externí zdroj k připojení vlastností, jedná se nejčastěji o COBie vlastnosti, ale velké množství softwarů podporujících BIM umožňuje zahrnout COBie přímo v samotném modelu. Vlastnosti také musí být přiřazeny pouze v případě jejich potřeby či účelnosti a nemohou být duplicitní. Tyto standardy se dodržují, aby se udržela jednoznačnost dat a přehlednost v rámci modelu.

Pokud se ve vlastnostech stejného objektu přeci jen duplicitní hodnoty nacházejí (ať už z pohledu názvu, či popisu), mělo by podle standardu dojít

k odstranění redundantních výskytů, a to podle priorit typů vlastností (to znamená preferenci odstranění redundancí z méně prioritních typů). Jak už bylo naznačeno, vlastnosti BIM objektů se dělí do několika typů podle priority a způsobu jejich přidělování. Rád bych však poznamenal, že priorita se používá pouze v případě duplicit, jak je zmíněno výše.

Na prvním místě z pohledu priorit jsou již vestavěné takzvaně hardcoded vlastnosti, což jsou vlastnosti pevně definované v BIM software, který uživatel používá. Tyto vlastnosti by měly být vždy vyplněné a jsou důležité pro správnou interpretaci a funkčnost modelu v daném software.

Dalším typem a na druhém místě z pohledu priorit jsou IFC vlastnosti, zkratka Industry Foundation Classes. Tyto vlastnosti jsou důležité k přenosnosti objektů a modelů mezi různými aplikacemi podporujícími BIM. K přenosu totiž zpravidla dochází za pomoci formátu IFC. IFC vlastnosti se pak dělí na dvě skupiny. První skupinou jsou běžné vlastnosti, takzvaně common. Ty se dají pro každý typ objektu nalézt ve standardu IFC od BuildingSMART a všechny takto určené vlastnosti by měly být pro daný typ objektu vyplněné. Další skupinou jsou dodatečné vlastnosti, které mohou, ale nemusí být vyplněné a slouží více ke specifickým účelům. Konkrétní vlastnosti si ukážeme v sekci věnující se přímo formátu IFC 1.7.

Na třetím místě v prioritě jsou COBie vlastnosti, zkratka pro Construction Operations Building Information Exchange. Tyto vlastnosti slouží k doplnění kompletnějších dat o konstrukci budovy a jejím postupném vývoji. Obsažená data pak převážně slouží k výměně a sdílení informací mezi projektovými týmy, stavebními firmami, majiteli budov, správci nemovitostí a dalšími zainteresovanými stranami v průběhu životního cyklu budovy. COBie je v rámci standardu považována za důležitou část, ale o nutnosti obsažených informací se standard IBOS ani OBOS nezmiňuje. Stručně se budeme COBie věnovat v samotné sekci později 1.4.

Na čtvrtém místě z pohledu priority jsou takzvané BOS_general vlastnosti. Tyto vlastnosti by měly být vždy vyplněné a zahrnují základní úroveň informací pro standard. Na tomto základu pak mohou stavět vlastnosti, které jsou na posledním místě priority a jsou zmíněné níže.

Dalšími vlastnostmi, které slouží vždy k určitému účelu, jsou následující:

- **Administrativní vlastnosti**: tyto vlastnosti slouží k zaznamenání informací o správě objektu, například kdo objekt vytvořil, přidal, či upravil, jestli byly provedeny nějaké změny a kdy a kde byly změny provedeny.
- **Proprietární vlastnosti**: tyto vlastnosti slouží pro specifické objekty, které mají určitého výrobce, a proto se týkají konkrétních výrobních detailů. Tyto vlastnosti umožňují určit výrobce, model, kód produktu, popisek produktu, místo a rok výroby.
- **Vlastnosti klasifikace**: tyto vlastnosti slouží k přiřazení určitého typu objektu, například dveře, okno a tak dále. Tyto vlastnosti pak slouží k jednodušší organizaci a lepší strukturovanosti celého modelu.

- **Vlastnosti konstrukčních specifikací**: tyto vlastnosti slouží k vytvoření vazeb objektů na konstrukční specifikace, které jsou pro model relevantní. Tímto se vytváří lehce sledovatelné prostředí pro dodržování norem a požadavků.
- **Vlastnosti správy majetku a zařízení**: tyto vlastnosti slouží pro správu vlastněného majetku a zařízení, týkají se většinou nákupů a proto se do vlastností uvádí doba nákupu, čárový kód, množství v koupené várce, informace o záruce a tak dále.
- **Vlastnosti výkonnosti produktu**: tyto vlastnosti se týkají výkonnosti produktů. Ve vlastnostech se přidávají odkazy na technické dokumenty a certifikace produktu, pro jednoduchý přístup k informacím jako je například nutnost údržby, poruchovost a tak dále.

Všechny zmíněné vlastnosti by pak měly být přehledně rozděleny do množin podle jejich zdroje, pokud to tedy BIM software umožňuje. Příkladem rozdělení je například množina IFC a množina COBie.

Analýza v této sekci nám vnáší důležitý vhled do standardu Building Information Modeling a vlastností, které se v něm mohou nacházet. I přesto, že testovací data jsou ve formátu Industry Foundation Classes (IFC), a proto se na první pohled může zdát analýza vlastností Building Information Modeling příliš podrobná, jedná se o nutný základ, který bude použit při zlepšení navigace v našich testovacích datech a jejich interpretaci. Nyní například víme, že názvy obsahují důležité kontextové informace, které nám pomohou lépe organizovat a rozumět datům při importu do Unreal Engine.

Z pohledu popsaných vlastností také víme, jaké informace a jaké typy informací můžeme v testovacích datech očekávat. Víme že model navržený podle Building Information Modeling, obsahuje nejen IFC vlastnosti, ale také vlastnosti COBie. Ty se zaměřují hlavně na následnou správu, a mohou proto obsahovat užitečné informace pro implementaci procházení budovou. Dalšími vlastnostmi, které budeme moci použít pro snadnější navigaci ve struktuře a k identifikaci objektů v souborech jsou vlastnosti klasifikace. V neposlední řadě víme, že některá data, jako jsou IFC běžné vlastnosti (neboli IFC common properties), by měla být vždy vyplněná. Tím pádem získáváme jistotu, jaké informace můžeme v datech při importu očekávat. Pro získání přesnějších údajů o obsahu informací v našich testovacích souborech však budeme muset využít informací z analyzovaných standardů pro důkladný rozbor našich testovacích dat.

1.4 COBie

V této sekci se budeme stručně věnovat COBie, protože i když to pro naší práci není tak důležitý prvek BIM jako jsou například IFC vlastnosti a IFC formát celkově, jedná se o neodmyslitelnou část BIM.

COBie, zkratka pro Construction Operations Building Information Exchange, je standardizovaný formát, který vznikl jako součást Building Information Modeling a převážně slouží k efektivnímu sdílení informací o zařízeních. Tyto informace zpravidla nejsou geometrického rázu a zaobírají se spíše údržbou, správou a provozem. Data jsou uložena ve struktuře tabulkového rázu, jako je například Microsoft Excel a jsou strukturována do více tabulek, které pokrývají různé aspekty zařízení a jejich umístění v modelu.

Hlavním cílem je usnadnit sdílení klíčových provozních informací mezi dodavateli, majiteli budov a všemi ostatními zainteresovanými stranami, a to během celého cyklu stavby budovy, ale i pak během celého života objektu. Snahou tohoto formátu je také pomáhat předdefinovat všechny důležité údaje k vyplnění potřebné pro efektivní správu a údržbu budovy. [7]

přesto, že COBie vznikl hlavně jako součást Building Information Modeling, je nyní využívaný i mimo něj, k jeho popularitě přispívá i kompatibilita s formátem IFC, což umožňuje jednoduchou interoperabilitu.

Pro lepší představení účelu a charakteru COBie dat zde přidávám příklady vlastností a tabulek v COBie obsažených. Jak už bylo řečeno, struktura COBie je rozdělena na více tabulek. Základní tabulka obsahuje instrukce ke COBie databázi a jejím výstupním souborům. Další tabulka slouží pro firmy a obsahuje kontaktní informace o firmách jakkoliv zapojených do projektu a tím zmíněných v COBie datech. Tabulka pro celé zařízení budovy uvádí například o jaký typ zařízení se jedná, jeho popis a adresu. V souvisejících tabulkách dochází k dělení budovy na tabulky reprezentující různá podlaží, místnosti, až v dělení docházíme k samotným základním objektům a jejich komponentům. Tyto tabulky obsahují vlastnosti jako jméno, popis, rozměry, výrobce, cena, sériové číslo a záruka. [8]

Kromě toho COBie obsahuje i tabulky zaměřené na úkoly a práce nezbytné pro provedení úspěšné údržby. Práce se pak dělí na jejich jednotlivé složky a podúkoly, nazvané eventy. Další tabulky zahrnují koordinaci sloužící pro geometrickou orientaci objektů, informace o rizikách, ale i informace o správě atributů, či dokumentů, které nejsou ve výchozí struktuře COBie a byly dodatečně uživatelem přidány. Toto je výčet sloužící pouze jako příklad, typů tabulek má COBie struktura 19 a v této kapitole nejsou všechny zmíněny. [8]

Z analýzy v této kapitole jsme získali podrobnější pochopení o COBie a jeho vlastnostech, které mohou být přítomny v našich testovacích souborech IFC. Ačkoliv se naše práce primárně zaměřuje na Industry Foundation Classes kvůli jejich obsahu geometrických dat, nezbytných pro vizualizaci v Unreal Engine, je důležité neopomenout i potenciální obsah COBie dat v našich testovacích souborech IFC. COBie data pak mohou nabídnout cenné negeometrické informace, jako jsou detaily o materiálech a osvětlení, které mohou sloužit k finální prezentaci modelů v prostředí Unreal Engine a tím podpořit relaiizaci věrohodné vizualizace importovaných geometrických dat.

Důkladná znalost COBie nám umožňuje efektivněji navigovat v testovacích souborech a vybírat relevantní data pro naše potřeby. Toto porozumění je

klíčové pro optimalizaci výsledku při importu do Unreal Engine, což umožní využít data obsažená v našich testovacích souborech na maximum. Analyzováním COBie komponent jsme tak získali schopnost identifikovat a extrahovat specifické informace, které mohou výrazně přispět k realističtějšímu a interaktivnějšímu zobrazení našich modelů v konečném výsledku.

1.5 IFC

Tuto sekci bych si dovolil započít citací Industry Foundation Classes definice vzané z IFC dokumentace standardu od BuildingSMART, ze které jsem také čerpal informace pro celou tuto sekci.

> *„ IFC, neboli Industry Foundation Classes pro sdílení dat ve stavebnictví a správě zařízení, představuje otevřený mezinárodní standard pro data BIM (Building Information Model), která jsou vyměňována a sdílena mezi softwarovými aplikacemi různých účastníků v sektoru stavebnictví a správy zařízení. Standard obsahuje definice pro data, jež jsou vyžadována pro budovy a infrastrukturní práce po celou dobu jejich životního cyklu. V rámci IFC jsou nyní zahrnuty i infrastrukturní zařízení jako mosty, silnice, železnice, vodní cesty a přístavní zařízení.“*
>
> [6]

Jak je řečeno v definici výše, tak IFC je zkratka pro Industry Foundation Classes a jedná se o mezinárodní otevřený standard, který podobně jako COBie slouží k výměně a sdílení informací o budově a všeobecně infrastrukturách během celého životního cyklu. Na rozdíl od COBie, které se soustředí především na operativní a údržbové informace, IFC zahrnuje také data geometrického rázu a kompletní 3D modely. Toto rozšíření činí IFC nezbytné pro potřeby Building Information Modeling a díky tomu slouží také jako klíčový formát pro export a sdílení mezi aplikacemi podporující BIM.

Nyní se zaměříme na klíčové vlastnosti, které dělají IFC formát nezastupitelným pro správnou funkčnost Building Information Modeling (BIM). První vlastností je interoperabilita, ta umožňuje různým architektonickým softwarům efektivní sdílení dat za pomoci společného formátu IFC. Sdílení díky IFC formátu probíhá bez problémů a nutnosti zdlouhavých úprav pro kompatibilitu. Univerzálnost formátu IFC značně snižuje riziko chyb při přenosu dat mezi platformami. Druhou klíčovou vlastností je standardizace dat, která se týká předdefinovaných specifikací, pokrývajících široké spektrum stavebních prvků a jejich vlastností, včetně materiálů, geometrických charakteristik a funkčních vlastností. Tato standardizace umožňuje konzistentnost pří používání IFC a přesnost v celém projektovém cyklu, od návrhu po realizaci.

Třetí vlastností je MVD, zkratka pro Model View Definition. MVD je možnost IFC obsahovat více pohledů a reprezentací dat. Data mohou být reprezentována různými způsoby, jako 3D modely, či pouze jako 2D pohled a vlast-

nosti objektů pak mohou obsahovat pouze určité množství informací a detailů. Uživatel si může vybrat, jaký pohled chce použít nebo si definovat svůj vlastní pohled. Definice pohledů byla však kvůli nutnosti vysoké interoperability IFC značně omezena. Nastavení těchto pohledů slouží hlavně pro efektivní sdílení v určitých částech projektu, kdy se nemusí sdílet všechny obsažené informace, ale pouze informace vybrané zvoleným pohledem. MVD tedy definuje, jaké informace a data se mají v jaké části projektu sdílet. Mezi základní pohledy patří takzvaný Coordination View, který se zaměřuje na poskytování informací k efektivní koordinaci paralelně probíhajících prací. Jednou z hlavních předností tohoto pohledu je sledování kolizí, které mohou při spojení více návrhů v modelu vzniknout. Druhým základním pohledem je takzvaný Reference View, který se zaměřuje na optimalizaci pohledu pro sdílení, obsahuje proto pouze malé množství základních informací, které jsou používané jako jednoduchá reference. Třetím základním pohledem je Alignment view, který se zaměřuje na stavbu úzkých dlouhých staveb, jako jsou například silnice, či železnice. Proto tento pohled obsahuje například vlastnost IfcAlignment či různé křivky, které jsou klíčové pro správné pozicování a celkové uspořádání zmiňovaných systémů silnic, železnic, či dalších podobných staveb. Zmíněné pohledy jsou pouze základní pohledy zmíněné standardem, na kterých se mohou stavět další, více konkrétní pohledy. [6]

Další vlastností je připravenost formátu IFC pokrýt celý projektový cyklus a nejen pouze jeho část, což je v přímém souladu s definicí Building Information Modeling. Tato vlastnost je dosažena díky rozsáhlému množství informací, typů a vlastností obsažených v IFC standardu, společně s Model View Definitions (MVD). MVD je přizpůsobeno pro různé části projektu, a proto tato komplexnost umožňuje efektivní správu projektu od jeho počáteční fáze návrhu až po finální provoz a údržbu.

V této sekci jsme provedli analýzu Industry Foundation Classes (IFC) a jeho klíčových vlastností, které umožňují efektivní sdílení dat v rámci Building Information Modeling (BIM). Získané informace jsou zvláště užitečné pro pochopení kontextu našich IFC testovacích souborů. Nyní chápeme, že data v našich souborech mohou pocházet z konkrétních Model View Definitions (MVD), které definují, jaké informace jsou zahrnuty a jak jsou prezentovány. Znalost MVD pro nás bude také velice klíčová, pokud budeme potřebovat importovat pouze část z nabízených dat a tím dosáhnout optimalizace importu.

Dalším důležitým poznatkem je schopnost formátu IFC efektivně sdílet data potřebná pro Building Information Modeling. Zjistili jsme, že IFC není jen o geometrických datech, ale umožňuje komplexní sdílení všech relevantních informací o budově. Tato vlastnost z něj činí zásadní nástroj pro sdílení modelů vytvořených pomocí standardů Building Infromation Modeling (BIM). Důležitým prvkem je také univerzálnost IFC formátu při sdílení mezi různými softwary. Tato vlastnost nám definuje IFC jako ideální formát pro výměnu dat, proto pravděpodobně nebude potřeba žádná konverze do vhodnějších formátů.

Konečně, pevná standardizace IFC formátu zajišťuje, že všechna data jsou

konzistentní a snadno interoperabilní mezi různými softwarovými platformami. Tato standardizace je v souladu s našimi zjištěními o BIM a usnadní nám správu a import relevantních informací do našich systémů. Protože je však standardizace struktury důležitá pro náš import, budeme se jí věnovat více i v následující sekci.

1.6 IFC – Struktura

V této sekci se podíváme podrobněji na základní schéma a strukturu IFC, což dále využijeme při analýze testovacích objektů, které v praktické části budeme importovat do Unreal Engine 5. Informace pro tuto sekci jsem čerpal opět z dokumentace Standardu IFC [6].

Základní vrstva struktury Industry Foundation Classes (IFC) je tvořena čtyřmi základními částmi, které se společně označují jako takzvaný core. Tyto části poskytují fundamentální struktury vrstvy, na kterých pak staví všechny ostatní odvozené vrstvy modelu. První z nich, takzvaně IfcKernel neboli jádro, tvoří základ celého schématu a zahrnuje základní objekty a konstrukty, které jsou využívány napříč celým systémem. Druhou částí tvořící core je IfcControlExtension, která se zaměřuje na rozšíření kontroly a managementu v rámci projektů. Třetí důležitou částí je IfcProcessExtension, která se věnuje definici procesů, akcí a činností nutných pro realizaci a údržbu stavebních objektů. Poslední částí je IfcProductExtension, která rozšiřuje jádro o konkrétní fyzické komponenty a produkty, s tvarem a umístěním v modelu. Všechny entity z core vrstvy, či z ní odvozené, však podléhají entitě IfcRoot, která všechny ostatní entity zaobaluje. IfcRoot přidává podléhajícím entitám požadovaný kontext a důležité identifikační údaje.

Jak už bylo zmíněno výše, jádro tvoří úplný základ, který je následně pouze rozšiřován ostatními částmi tvořícími core vrstvu. Jádro tak obsahuje entity, které lze označit jako neuživatelské, jelikož je uživatel přímo nevyužívá. Tyto entity definují základní struktury, objekty a kontrolní funkce, které jsou nezbytné pro bezproblémovou funkci IFC. Jádro mezi jinými zahrnuje klíčovou entitu IfcRoot, která každé podřízené entitě přiděluje unikátní identifikační údaje. IfcRoot také umožňuje přiřazení vlastníka, zaznamenání změn, a definuje název a popis entity. Jak říká dokumentace IFC standardu

> *„V IFC modelu existují tři základní typy entit, které jsou odvozeny od IfcRoot. Ony pak tvoří první úroveň specializace v hierarchii entit.*
>
> - *Definice objektů jsou zobecněním jakékoliv sémanticky zpracované věci (nebo položky) v rámci IFC modelu.*
> - *Vztahy jsou zobecněním všech vztahů mezi věcmi (nebo položkami), které jsou vnímány jako objektivizované vztahy mezi různými entitami.*

- *Definice vlastností jsou zobecněním všech charakteristik, které mohou být přiřazeny definicím objektů.*

“

[6]

Jak je řečeno v citaci, jádro obsahuje i další velmi důležité entity, odvozené od IfcRoot. První z nich, IfcObjectDefinition, je abstraktní entita reprezentující vše od fyzicky existujících prvků, jako jsou zdi, dveře nebo střechy, po definované prostory, jako jsou jednotlivé místnosti. Součástí této entity jsou i nehmotné prvky jako jsou plánovací mřížky nebo virtuální ohraničení, určující velikosti. Dále pokrývá definice pro procesy v modelu, práce v rámci projektu a pracovníky na projektu se podílející. Dále se dělí objekt do 6 hlavních entit: Produkty, Procesy, Řízení, Zdroje, Aktéři a Skupiny.

Druhá entita, IfcRelationship, se zabývá vazbami, které jsou také reprezentovány jako objekty. Tím má každá vazba svůj vlastní identifikátor a může nést dodatečné informace, což přispívá ke svobodě uživatele. IfcRelationship se dále dělí na entity: Přiřazení, Asociace, Dekompozice, Definice, Konektivity a Deklarace.

Třetí z nich, IfcPropertyDefinition, zajišťuje definice vlastností. Jedná se o abstraktní entitu, která může být přiřazena k objektům, a tím jim poskytovat šablony charakteristik nebo celé soubory charakteristik, které pak mohou být uživatelem vyplněny. Obvykle se IfcPropertyDefinition přiřazuje k určitému typu objektů právě jednou, a od tohoto typu pak ostatní podřízené typy charakteristiky a vlastnosti dědí. IfcPropertyDefinition se dělí na dvě základní části: na sady vlastností s jejich šablonami a na výskyty sad vlastností. První část definuje syntaxi a obsahuje typy dat, zatímco druhá část připojuje sdílené vlastnosti k typům objektů v době jejich výskytu.

Jako další příklad klíčové entity v jádru můžeme zmínit IfcUniqueDefinitionNames, která je zásadní pro kontrolu unikátnosti názvů entit v celém IFC modelu. Tato entita kontroluje, zdali jsou jména entit v celém IFC modelu unikátní. Tato entita tak zajišťuje, že každý objekt má jedinečné označení a umožňuje efektivní správu dat softwarům, či prostředím s IFC pracujícím. Tento příklad jasně ilustruje, že jak už bylo řečeno, jádro obsahuje spíše entity, které zaručují vnitřní funkčnost a integritu IFC, či tvoří abstraktní definici pro konkrétnější podentity.

Z analýzy jádra IFC modelu plyne, že pochopení jeho funkcí je klíčové pro celkový přehled o struktuře IFC souborů a zajištění funkčnosti celého systému. Avšak informace získané z dosavadní analýzy jádra jsou příliš abstraktní k tomu, aby nám poskytly přímé návody na manipulaci a import specifických objektů do Unreal Engine. Proto se nyní zaměříme na entity více konkrétní a rozšíříme analýzu i na ostatní části core vrstvy, rozšiřující jádro, či na entity popsané i mimo core schéma, jako jsou například důležité entity IfcMaterial a IfcGeometryResource, nacházející se ve vrstvě obsahující zdroje používané během cyklu stavby.

Prvním důležitým prvkem, který je však stále obsažen v jádru core vrstvy, je entita IfcProject. Tato entita dědí z IfcObjectDefinition, která je zmiňovaná výše a tím přebírá i vlastnění unikátní identifikace a schopnosti přiřazovat jména s popisky. IfcProject je entita, ke které má většinou běžný uživatel přístup a zastřešuje tak celou strukturu konkrétního IFC projektu. Z dokumentace od BuildingSMART lze pak vyčíst, že hlavní účel IfcProject je poskytnutí nezbytných vlastností z dříve zmiňované entity IfcRoot. IfcProject také nastavuje kontext celého projektu jako jsou defaultni jednotky, použitý souřadnicový systém, či přesnost při geometrické reprezentaci. [6] Tato entita je tak pro naší práci důležitá, protože z ní můžeme započít procházení celé geometrické struktury, či se můžeme dozvědět důležitý kontext o použité geometrii.

Druhým důležitým prvkem, který je opět obsažen v jádru core vrstvy, je IfcProduct. Ten reprezentuje abstraktní definici čehokoliv, co se pak nachází v geometrické reprezentaci IFC. IfcProduct má tak vždy pozici, která může být buď relativní k jinému produktu nebo relativní k světovým souřadnicím. Tato entita ve schématu dědí z IfcObject, který dědí z IfcObjectDefinition. Z dokumentace si můžeme uvést obsažené atributy, které nám tak určí základ atributů pro všechny ostatní objekty ve scéně. Podle dokumentace BuildingSMART [6] obsahuje IfcProduct atributy popisující umístění ve scéně nebo jakou produkt používá grafickou reprezentaci pro zobrazení. Atributy popisující relace mezi ostatními objekty obsahují odkazy na vztahy, kdy IfcProduct může být součástí jiných produktů, vůči jakému objektu je atribut souřadnic relativní a v neposlední řadě kde ve struktuře projektu je objekt umístěn. [6] Všechny tyto atributy budou následně velmi důležité pro správné umístění importovaných objektů do scény a do struktury projektu.

Od IfcProduct pak vychází přímo rozšíření jádra takzvané IfcProductExtension, které jádro rozšiřuje o další entity zaměřené primárně na objekty fyzické. [6] Tato část základního schématu tak obsahuje většinu fyzických objektů, které tak mají i geometrickou reprezentaci a musíme je proto zahrnout do importu.

Mezi entity patřící do IfcProductExtension, které budou pro naši práci důležité, patří například IfcSite. Tato entita dědí od IfcSpatialStructureElement, což je podskupina IfcSpatialElement. IfcSpatialElement pak dědí přímo od IfcProduct, který je zmíněný výše. IfcSpatialElement odděluje fyzické prvky umístěné ve struktuře stavebních prvků od prvků, jako je například IfcAnnotation, který přidává informační prvky do modelu. IfcSite slouží jako interpretace pozemku, na kterém je staveniště konstruováno. Jako atributy proto obsahuje zeměpisnou šířku, zeměpisnou výšku a vyvýšení. [6] IfcSite je v IFC struktuře modelu ihned pod IfcProject a proto je pro náš projekt důležitá jakožto komponenta, kterou budeme muset zohlednit v importu.

Než přejdeme k další entitě, je důležité podrobněji prozkoumat IfcSpatialStructureElement. Tato entita umožňuje svým dědičům definovat strukturu prostorového uspořádání pomocí speciálního atributu CompositionType.

Tento atribut nabízí tři možnosti pro určení úrovně seskupování: první, takzvaný COMPLEX, slouží pro skupiny objektů, druhý, nazývaný ELEMENT, je pro jednotlivé objekty, a poslední, s názvem PARTIAL, je pro dílčí segmenty objektů. Každá z těchto úrovní umožňuje přesněji specifikovat, jak jsou objekty organizovány a jak spolu souvisí v rámci projektu. [6]

Další entitou patřící do podschématu IfcProductExtension je IfcBuilding. Ta dědí od IfcFacility a IfcFacility je stejně jako IfcSite podskupina IfcSpatialStructureElement. Ifc Building podle dokumentace reprezentuje konstrukci, co slouží jako úkryt pro osoby a má pevné místo v modelu. IfcBuilding může obsahovat buď pouze jednu stavbu, ale i množinu více staveb rozesetých po IfcSite. Entita, tak může být rozdělena na části pomocí speciálního atributu Composition type, kdy každá část představuje svou samostatnou stavbu. Tuto informaci bude pak nutno při importu zohlednit, protože importované informace pak nemusí být souvislé. IfcBuilding rozšiřuje atributy o referenční výšku, od které se počítá výška budovy a dále o výšku, ve které se nachází umístění budovy (obě výšky se udávají od hladiny moře). Posledním atributem je adresa budovy. [6] Všechny tři atributy však udávám pouze pro referenci, kdyby byly náhodou obsaženy v našich testovacích datech. Tyto atributy jsou totiž už zastaralé a v novějších verzích IFC (4.3.0.0 a novější) se už nepoužívají. Pro analýzu je však IfcBuilding opět velice důležitou komponentou, protože ve struktuře IFC je ihned pod IfcSite a zaobaluje všechny důležité geometrické komponenty budovy.

Třetí enitou patříci do podschématu IfcProductExtension je IfcBuildingStorey. Tato entita reprezentuje podlaží v budovách. Podle dokumentace se však nemusí jednat pouze o jedno podlaží, ale i o skupinu podlaží. Proto podobně jako u IfcBuilding dědí tato entita z IfcSpatialStructureElement a tím implementuje speciální atribut CompositionType sloužící k dělení na jednotlivá patra. Jakožto atribut obsahuje IfcBuildingStorey vyvýšení relativní k referenční výšce budovy. Stejně jako u IfcBuilding se však jedná již o zastaralý atribut. [6] Tato entita je pak napojena v IFC struktuře modelu ihned na IfcBuilding a tak obsahuje důležité informace o struktuře importovaného souboru.

Čtvrtou entitou patřící do podschématu IfcProductExtension je IfcSpace. Tato entita reprezentuje pokoje, či jakékoliv prostory v budově sloužící k nějakému účelu. IfcSpace může být také mimo budovu, a proto může být připojené i rovnou na IfcSite. Jako u předešlých dvou entit Ifc space může tvořit jeden prostor, ale i množinu spojených prostorů, a proto se opět používá speciální atribut CompositionType k možnému rozdělení jednotlivých prostorů. IfcSpace dědí přímo od IfcSpatialStructureElement a mezi atributy, které obsahuje, patří předdefinovaný typ, určený k přiřazení určitého typu pokoje. Dalšími atributy jsou vyvýšení, které určuje, na jakém podlaží se prostor nachází a dva atributy říkající, zdali a jaký má prostor přiřazený strop a zdali a čím je ohraničený a tím oddělený od ostatních "venkovních" prvků. [6] IfcSpace je pak v IFC struktuře modelu napojena přímo na určité podlaží IfcBuildingStorey, či přímo na staveniště IfcSite, pokud se nachází mimo bu-

dovu. Je tedy nutné si dávat pozor na prostory i mimo budovu. Prostory jsou dalším velice důležitým prvkem struktury modelu, protože dělí konstrukci na rozdílné části, které pak mohou obsahovat další fyzické objekty, které jsou různé podle účelu prostoru (například nábytek). Atribut popisující přiřazený strop pak může být v naší práci výhodný například při přiřazení světel.

Další důležitou entitou k analyzování je entita IfcElement. Tato Entita dědí přímo z IfcProduct zmíněného výše. Patří opět do podschéma IfcProductExtension a jedná se o abstraktní typ reprezentující fyzické prvky v struktuře modelu. Narozdíl od předešlých zmíněných se však jedná o přímé fyzické objekty jako je například stěna (IfcWall), dveře (IfcDoor) nebo například nosníky (IfcBeam). Konkrétním typům se budeme věnovat v následující sekci 1.8. Atributy, které se nám bude hodit znát při importu, jsou atributy říkající, zdali tento element nějakým způsobem ovlivňuje jiný element nebo je naopak ovlivňován. Ovlivnění se nastavuje například kvůli možnosti kolizí dvou elementů při pozdějších fázích objektů. Důležitým atributem pro identifikaci je tag označující konkrétní produkt většinou pozicí, či výrobním číslem. Další dvojice atributů umožňuje udělat v elementu takzvané openings, které je pak možné vyplnit jinými elementy. Tato dvojice tak říká, jestli nějaké openings mám a jestli nějaké openings vyplňuji. Openings pak slouží pro vztah například stěna okno, či stěna dveře, kdy stěna má prostor na okno a okno ho pak vyplňuje. Tento atribut by pak v naší práci mohl být velmi důležitý v kontextu rozdělování stěn na ty "obyčejné" a na ty, které potřebují speciální péči, například z důvodu obsahu dveří. Dalšími důležitými atributy jsou dva atributy sloužící k reprezentaci spojení mezi elementy. Toto spojení zpravidla znamená fyzické spojení dvou elementů a je tvořeno entitou IfcRelConnectsElements. Posledními atributy obsaženými v IfcElement jsou atributy říkající, zdali a jaké má element krytí ve formě IfcCovering, zdali a jaké má element ohraničení a zdali a k jaké je přiřazen struktuře. [6]

Tato entita nemá určené přesné místo v IFC struktuře modelu, ale entity odvozené od tohoto abstraktního typu budou buď přímo či nepřímo pod IfcSite. Tato entita je velmi důležitá, protože z ní dědí většina importovaných geometrických dat.

Pro další entitu si budeme muset nejdříve definovat další vrstvu ze schématu IFC. Jedná se o vrstvu Resource definition data. Tato vrstva obsahuje specifikace, které umožňují popsat a klasifikovat zdroje používané v rámci stavebního procesu. Zahrnuje informace o materiálech, zařízeních a pracovních silách, které mohou být opakovaně využívány v různých projektech. [6]. V tomto schématu se pak nachází IfcMaterialResource, který je klíčový pro definování materiálů v objektu, proto bude toto schéma důležité pro správný import, či následné přiřazení materiálů.

IfcMaterialResource je schéma, které obsahuje informace o materiálech. Tyto materiály jsou přiřazovány k elementům ve struktuře IFC, což popisuje, z jakých materiálů se daný fyzický objekt skládá. Materiály se mohou lišit ve způsobu, jakým jsou přiřazeny, ale v IFC souborech slouží především jako

popisný formát poskytující informace o materiálech použitých během stavby.

Jak už bylo řečeno v předešlém odstavci, materiály se mohou lišit podle typu přiřazení k objektu na IfcMaterialLayerSet, IfcMaterialProfileSet, IfcMaterialConstituentSet a IfcMaterial. IfcMaterial je z těchto možností nejjednodušší, kdy se k objektu přiřadí pouze jednotlivá entita IfcMaterial. To ale funguje jen tehdy, pokud má objekt pouze jeden materiál. Druhým nejjednodušším je IfcMaterialConstituentSet, který přiřazuje množinu materiálů pomocí IfcMaterialConstituent. Ten obsahuje pouze materiál a nepovinný textový popis, jak je materiál přiřazen. Jako další atribut může IfcMaterialConstituent obsahovat informaci, jak velkou část objektu materiál v přiřazené množině zabírá. O trochu komplikovanější je pak IfcMaterialLayerSet, který opět obsahuje množinu materiálů, ale tentokrát zaobalených v IfcMaterialLayer. IfcMaterialLayer definuje, jaký materiál je použit a jak tlustá je použitá vrstva tohoto materiálu. Toto se uplatňuje například u Stěn, které mohou být tvořeny různými vrstvami materiálu jako polystyren, beton a omítka. Poslední IfcMaterialProfileSet je sada, která obsahuje jedno nebo více IfcMaterialProfile. Každé IfcMaterialProfile se skládá z materiálu a geometrického profilu, který tento materiál reprezentuje. Geometrický profil je definován pomocí entit odvozených od IfcProfileDef, kde každá odvozená entita specifikuje jiný tvar a jeho rozměry. Tímto způsobem se vytváří průřezová část, k níž je přiřazen konkrétní materiál.

Dále existují IfcMaterialLayerSetUsage a IfcMaterialProfileSetUsage. Tyto množiny jsou napojeny na IfcMaterialLayerSet a IfcMaterialProfileSet, u kterých popisují, jak mají být materiály v množinách použity. Proto obsahují atributy, které udávají i jak jsou materiály orientovány a umístěny.

Protože celou dobu popisujeme IfcMaterial, ale ještě jsme si ho nedefinovali, pojďme tak učinit nyní. Ifc Material je entita představující hmotu, která tvoří určitý objekt. Jakožto atributy má jméno, popis, do jaké kategorie spadá a jestli má reprezentaci IfcMaterialDefinitionRepresentation. IfcMaterialDefinitionRepresentation pak slouží k vizuální reprezentaci materiálu. [6] Tento atribut pro nás tedy bude velmi klíčový k zobrazení materiálů v Unreal Engine.

Již jsme zmínili definování materiálů, ale pro správnou interpretaci dat v geometrickém zobrazení v IFC slouží další entita. Tou entitou je IfcRepresentation, která patří do vrstvy definice datových zdrojů. Způsob, jakým se data mají interpretovat, je specifikován v atributu IfcRepresentationContext, a proto se na ní podíváme blíže. IfcRepresentationContext je abstraktní entita, z níž dědí IfcGeometricRepresentationContext a IfcGeometricRepresentationSubContext. IfcGeometricRepresentationContext definuje kontext používaný k reprezentaci tvarů objektů. K tomu slouží nastavení atributů určujících dimenzi souřadnicového prostoru, přesnost geometrických dat, světový souřadnicový systém pro převody mezi systémy a směr k severu pro orientaci. Pro podrobnější definici kontextu se používá odvozený IfcGeometricRepresentationSubContext, který se zaměřuje na detailnější úroveň interpretace. Na rozdíl

od IfcGeometricRepresentationContext, který se nastavuje pro všechny geometrické reprezentace v projektu, umožňuje IfcGeometricRepresentationSubContext nastavení pro specifické části s širší škálou možností. [6]

Jak bylo již dříve naznačeno, v IFC jsou různé vztahy, které mezi sebou mohou entity udržovat. K tomu slouží v IFC další entita s názvem IfcRelationship. Tato entita patří do jádra a dědí z IfcRoot. IfcRelationship je abstraktní entita, která zahrnuje všechny objektivizované vztahy. Objektivizace vztahu znamená, že vztah mezi dvěma prvky je reprezentován jako objekt, což umožňuje přidružení dalších informací v podobě atributů. IfcRelationship rozšiřuje své potomky o atributy, které definují unikátní identifikátor, jméno s popisem a sledují, kdo relaci vytvořil nebo změnil. Odvozené vztahy z této entity dále specifikují, které entity jsou ve vztahu zahrnuty. [6]

1.7 IFC – Common property

Jaké vlastnosti můžeme očekávat v modelech Building Information Modeling, jsme rámcově probrali již v sekci o BIM standardu 1.3. v této sekci se podíváme na takzvané IFC common properties, neboli běžné vlastnosti IFC, které by měly být povinně vyplněné pro každý objekt. V této sekci si tak přiblížíme, co IFC common properties jsou a jestli informace o nich můžeme využít k importu. Následné informace o nich vychází z dokumentace Industry Foundation Classes od BuildingSMART [6].

Ifc common properties, stejně jako všechny ostatní vlastnosti, jsou přiřazovány k objektům za pomocí entity IfcPropertySet. Tato entita je klíčová pro skupinování vlastností, které definují charakteristiky objektů. Každý IfcPropertySet se skládá z jedné nebo více entit IfcProperty, přičemž každá z těchto entit může mít odlišné typy a struktury.

Typicky je IfcPropertySet přiřazen k neabstraktním entitám, jako jsou IfcBuilding, IfcDoor nebo IfcWindow, a nazývá se podle vzorce `Pset_<jméno entity>Common`. Například `Pset_BuildingCommon` pro budovy obvykle zahrnuje vlastnosti jako jsou referenční čísla, identifikační kódy budovy, počet pater, rok konstrukce a další specifika týkající se bezpečnosti a užitkových vlastností.

Mezi běžné vlastnosti pro elementy jako okna může `Pset_WindowCommon` zahrnovat údaje o úrovni bezpečnosti, požární ochraně, tepelné izolaci a akustických vlastnostech.

Z analýzy však vyplývá, že ačkoliv množina běžných vlastností obsahuje užitečné informace pro konstrukční účely, pro cíle naší práce, které se zaměřují na import a vizualizaci, tyto vlastnosti nejsou kritické.

1.8 IFC – Elementy

Protože Building Information Modeling je úzce spojeno s formátem IFC a naše testovací data jsou taktéž ve formátu IFC, tak se v této sekci podíváme na základní IFC Elementy, které mohou být v IFC modelu obsaženy. Tyto elementy je také možné využít pro přiřazování materiálů, a proto je důležité vysvětlit si význam alespoň některých z nich. Informace v této kapitole jsem čerpal z dokumentace standardu IFC od BuildingSMART [6] a seznam se snaží pokrýt elementy z testovacích souborů.

- **IfcBuildingElementProxy**: Jakýkoliv proxy element definuje výchozí typ, který se přiřadí, pokud neexistuje žádný jiný vhodný typ entity.
- **IfcRoof**: IfcRoof reprezentuje střechu budovy.
- **IfcWall**: IfcWall reprezentuje stěnu budovy.
- **IfcDoor**: IfcDoor reprezentuje dveře budovy.
- **IfcWindow**: IfcWindow reprezentuje okno budovy.
- **IfcSlab**: IfcSlab reprezentuje jakoukoliv podlahu budovy.
- **IfcCovering**: IfcCovering reprezentuje strop, či jakékoliv jiné zastřešení prostoru.
- **IfcFurniture**: IfcFurniture reprezentuje nábytek.
- **IfcStair**: IfcStair reprezentuje celé schodiště.
- **IfcMember**: IfcMember reprezentuje zpevňující prvek, který je určen k udržení váhy a podpoře ostatních elementů.
- **IfcStairFlight**: IfcStairFlight reprezentuje část celého schodiště IfcStair.
- **IfcRailing**: IfcRailing reprezentuje zábradlí.
- **IfcAirTerminal**: IfcAirTerminal reprezentuje jakýkoliv výstup ze vzdušného potrubí.
- **IfcDuctFitting**: IfcDuctFitting reprezentuje spojení mezi potrubím a IfcAirTerminal.
- **IfcFireSuppressionTerminal**: IfcFireSuppressionTerminal reprezentuje jakýkoliv výstup ze struktury proti požáru (například protipožární rozprašovače vody).
- **IfcBeam**: IfcBeam reprezentuje nosník, který stejně jako IfcMember slouží k přenosu váhy.
- **IfcFooting**: IfcFooting je další element k přenosu váhy. Tentokrát ale váhu přenáší v základu budovy na povrch v okolí.

1.9 Software pro vizualizaci BIM

Pro snadnější analýzu testovacích IFC souborů je vhodné použít určité softwary, které prohlížení BIM modelů podporují. Z postupů, které slouží v těchto softwarech k vizualizaci geometrických prvků BIM se pak také můžeme inspirovat v naší práci, proto si v této sekci stručně zmíníme několik zdarma dostupných aplikací podporujících formát IFC.

1.9.1 BIMvision

> *„BIMvision [9] je bezplatný prohlížeč modelů IFC. Umožňuje zobrazovat virtuální modely pocházející z CAD systémů jako Revit, Archicad, BricsCAD BIM, Advance, DDS-CAD, Tekla, Nemetschek VectorWorks, Bentley, Allplan a dalších, bez nutnosti vlastnit komerční licence těchto systémů nebo mít k dispozici specifické prohlížeče od jednotlivých výrobců. BIMvision vizualizuje BIM modely vytvořené ve formátu IFC 2×3 a 4.0. Disponuje mnoha vestavěnými funkcemi a je prvním prohlížečem s rozhraním pro pluginy.“ “*
>
> [9]

Jak uvádí citace z oficiální webové stránky, BIMvision je freeware zaměřený na vizualizaci BIM. Tento software se neomezuje pouze na geometrickou reprezentaci, ale zvládá zobrazit i další data, jako jsou vlastnosti, umístění, klasifikace a vztahy. BIMvision zobrazuje hierarchii entit ve standardní IFC struktuře (což znamená, že vše zastřešuje projekt, pod kterým je staveniště obsahující stavby, ty se dělí na podlaží a podlaží dále obsahují prostory s dalšími elementy). Software je také schopen strukturovat entity podle předem nastavených klasifikací, skupin, typů objektů nebo umístění ve vrstvách modelu.

Geometrická data jsou vizualizována s využitím určité formy ambientního osvětlení, aby byla docílena viditelnost všech objektů. Vizuální zobrazení materiálů objektů (obarvení) je v BIMvision přiřazováno k objektům v následujícím pořadí: na prvním místě je obarvení podle přiřazeného materiálu (obyčejný IfcMaterial nestačí, musí být přiřazen grafický materiál s nastavenou barvou) a na druhém místě je nastavení materiálu podle typu IfcElementu (například IfcWindow, neboli okno, je skleněné, IfcRoof, neboli střecha, je červená, a tak dále). Tento typ přiřazení materiálů a osvětlení by pak mohl být velmi účinný pro vizualizaci v Unreal Engine.

1.9.2 FreeCAD

FreeCAD [10] je freeware, který poskytuje prostředí pro parametrické modelování. Není sice na BIM zaměřený, ale umí IFC soubory vizualizovat a pomocí dostupných zásuvných modulů (Arch [11] a BIM Workbench [12]) i ve velké míře upravovat. Schopnost FreeCAD pracovat s IFC je z velké míry závislá na

knihovně IFC Openshell, kterou si analyzujeme později v samostatné podsekci 1.12.2.

FreeCAD, stejně jako BIMVision, umožňuje zobrazení jak informační, tak geometrické stránky IFC souboru, avšak geometrická část je značně omezena. Model není nijak obarven a spíše než o 3D reprezentaci jde u FreeCad o zobrazení 2D obrázků z různých předdefinovaných pohledů. I přesto, že je FreeCAD v rámci informační složky IFC slabší než BIMview, zvládá zobrazit všechny důležité IFC atributy elementů a IFC strukturu.

1.9.3 Xbim toolkit

Xbim toolkit [13] je sada open-source nástrojů určená pro práci s BIM, specificky pro .NET vývojáře. Tento toolkit tak rozšiřuje .NET prostředí a umožňuje manipulaci společně s vytvářením BIM modelů ve formátu IFC. Součástí xBIM je také aplikace Xbim Xplorer, která demonstruje schopnosti těchto nástrojů a na kterou se podíváme blíže v této podsekci.

Xbim Xplorer, stejně jako obě předešlé aplikace, nabízí vizualizaci geometrických dat a zobrazuje informační stránku modelu. Aplikace poskytuje 3D vizualizaci modelů s automaticky přiřazenými materiály podle typu IFC entit. Automatické přiřazení materiálů ve vizualizaci se neaplikuje, pokud je již nastavený nějaký specifický grafický materiál. Vizualizace tohoto softwaru je tak podobná jako u BIMvision. Xbim Xplorer věrně zobrazuje všechny IFC atributy, ale může mít problémy se zobrazením materiálů přiřazených skrze množinu (například IfcMaterialLayerSet). Co se týče zobrazované struktury, Xbim Xplorer je věrný základní hierarchii IFC a žádné jiné zobrazení struktury nenabízí.

1.9.4 BlenderBIM

BlenderBIM [14] je zásuvný modul, který rozšiřuje funkcionalitu Blenderu o možnost práce s IFC soubory. Tento modul je postaven na Python knihovně IfcOpenShell, kterou si podrobněji představíme později. Co se týče vizualizace, BlenderBIM umožňuje import geometrie z IFC formátu přímo do 3D grafického softwaru Blender, kde s ní může uživatel pracovat s minimálními změnami, stejně jako s jakýmkoliv jiným 3D objektem v Blenderu. Modul nenabízí automatické přiřazování materiálů, ale zachovává stejnou strukturu jako hierarchie IFC. Co se týče informační stránky, BlenderBIM neposkytuje mnoho informací uvnitř, ale kompenzuje to možností exportovat vlastnosti COBie do tabulky Excel.

1.10 Analýza testovacích souborů

Testovací soubory nám byly poskytnuty dva. Každý soubor je však naprosto jiný, a tak si v této kapitole rozebereme oba dva. Analýza nám pak poskytne

důležité informace o tom, jaká struktura a jaké informace se mohou při importu vyskytovat.

1.10.1 Analýza ZaB

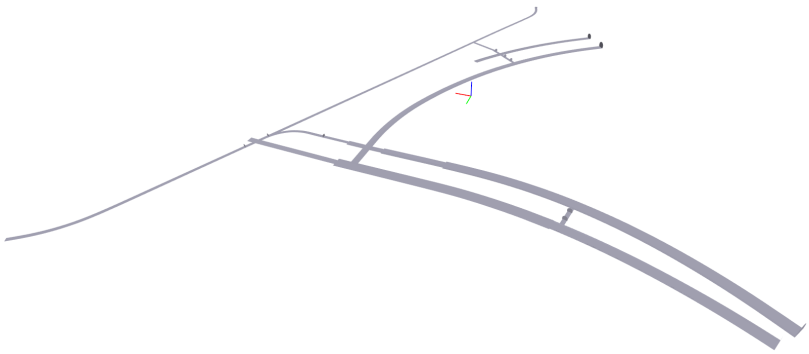
Jako první soubor k analýze použijeme ten jednodušší, nazvaný *ZaB_only_floors_doors_doorWalls.ifc*. Ten, jak název napovídá, obsahuje převážně podlahy, dveře a stěny kolem dveří. Významnou část modelu tvoří podlaha rozprostírající se do rozsáhlého systému chodeb, které jsou často přerušovány stěnami s dveřmi. Základní tvar objektu můžeme vidět na obrázku 1.1.

Hlavními problémy, které mohou nastat při importu do Unreal Engine, jsou nedostatečně vymodelované modely ve 3D (všechny objekty jsou reprezentovány pouze jako 2D plochy, bez hloubky či objemu). Druhým problémem, na který se budeme muset zaměřit, je nesprávné měřítko modelu, kde například dveře mají výšku pouze 4 milimetry, což je oproti reálné výšce (okolo dvou metrů), velký rozkol s reálným světem. Tyto problémy ještě více komplikuje absence detailních informací o komponentách, neboť všechny elementy jsou typu IfcBuildingElementProxy a obsahují jen základní atributy jako globální identifikátor, typ a popis. Typ je konzistentní pro všechny objekty a popis "tessellated proxy"naznačuje použití pouze zjednodušené geometrie. Typ IfcBuildingElementProxy je použit pro nespecifikované stavební elementy, což naznačuje, že model není plně specifikován, čemuž nahrává i absence jakýchkoliv materiálů (grafických i pouze informačních).

Struktura modelu odpovídá tradiční hierarchii IFC, kde projekt zastřešuje entita IfcProject, následovaná entitou IfcBuilding, která obsahuje všechny nespecifikované entity IfcBuildingElementProxy. Testovací soubor je ve formátu IFC 4.0.0. Z analýzy vyplývá, že tento testovací soubor porušuje BIM formát, ať už ve škálování, které neodpovídá 1:1, či nedostatečným poskytnutím informací o modelu samotném. V naší praktické části se tak budeme muset zaměřit i na modely, které nejsou dostatečně definované.

1.10.2 Analýza University Building Model

Druhým testovacím souborem, který je o poznání rozsáhlejší, je soubor `University Building Model.ifc`. Tento model zobrazuje budovu univerzity kompletně vybavenou nábytkem, jako jsou lavice a židle. Model odpovídá tradiční IFC hierarchii, kde model je zastřešen projektem (IfcProject), obsahujícím jedno staveniště (IfcSite). Staveniště zahrnuje univerzitní budovu (IfcBuilding), která je rozčleněna na patra (IfcBuildingStorey). Patra jsou v budově tři, kdy dvě první obsahují stěny (IfcWall) s okny (IfcWindow) a třetí je rezervováno pouze pro střechu (IfcRoof). První dvě patra jsou propojena schody (IfcStair) a obsahují pět přímých typů potomků: Již zmiňované stěny (IfcWall) a okna (IfcWindow), dveře (IfcDoor), nosné desky sloužící jako podlahy (IfcSlab) a prostory (IfcSpace). Schody (IfcStair) se v hiaerarchii také



Obrázek 1.1 Obrátzek ukazující vizualizaci testovacího souboru ZaB_only_floors_doors_doorWalls.ifc

nachází ihned pod podlažím, ale jsou obsaženy pouze v prvním patře. Schody se skládají z několika částí. První komponentou jsou ramena schodiště (IfcStairFlight), což je assembly komponenta složená z jednotlivých schodů, tvořící nepřerušovanou část schodů až do části podlahy (IfcSlab), která je také částí schodů. Dalšími komponenty jsou takzvané pruty (IfcMember), sloužící jako statická podpora pro stabilitu schodiště. Poslední komponentou je zábradlí (IfcRailing).

První dvě patra jsou rozdělena do různých prostor, přičemž každý prostor obsahuje specifické entity. Všechny prostory v těchto patrech obsahují vzdušné větrání (IfcAirTerminal a IfcDuctFitting), rozprašovače proti požáru (IfcFireSuppressionTerminal) a strop (IfcCovering). Prostory, které slouží jako učebny, jsou navíc vybaveny nábytkem (IfcFurniture), čímž se liší od ostatních prostor, které mohou být určeny pro jiné účely (například prostor chodby).

Vzdušné větrání se skládá z difuzoru (IfcAirTerminal) a potrubní spojky (IfcDuctFitting), která difuzor napojuje na potrubí. Místnosti, kde se nábytek nachází, obsahují dohromady následující typy nábytku: studentské židle, učitelské židle, židle na kolečkách, tři typy studentských lavic, učitelské stoly a čtyři typy poliček.

Materiál je v tomto modelu přiřazen k entitám ve třech formách. Některé typy nábytku (IfcFurniture) mají přiřazený materiál pouze za pomoci entity IfcMaterial, neboli mají přiřazený pouze jeden IfcMaterial. Takto přiřazeny jsou materiály pojmenované “Birch” (Bříza), “Laminate, Ivory, Matte” (Laminát, slonová kost, matný), “Wood – Stained” (Dřevo – mořené). Druhá forma přiřazení je přiřazení více vrstev materiálu za pomoci IfcMaterialLayerSet. Tato metoda umožňuje složitější konstrukce, kde jednotlivé vrstvy mohou mít různé materiálové vlastnosti. Materiály jsou jako vrstvy přiřazeny ke

stěnám (IfcWall), ke stropům (IfcCovering), ke střeše (IfcRoof) a k podlahám (IfcSlab). Konkrétně jsou materiály přiřazeny takto:

- Stěna prvního typu:
  - ”Gypsum Wall Board”(Sádrokartonové stěny)
  - ”Metal Stud Layer”(Vrstva s kovovými kolíky)
- Stěna druhého typu:
  - ”Brick, Common”(Běžné cihly)
- Strop:
  - ”Aluminum”(Hliník)
  - ”Clad – White”(Bílý obklad)
- Střecha:
  - ”Default Roof”(Základní střecha)
- Podlaha:
  - ”Concrete, Cast-in-Place gray”(Šedý beton, litý na místě)

Posledním typem přiřazení materiálu je v tomto modelu za pomoci IfcMaterialConstituentSet. Materiál takto přiřazen k některým typům nábytku (IfcFurniture), k oknům (IfcWindow) a ke dveřím (IfcDoor). Konkrétně jsou materiály přiřazeny následovně:

- Nábytek prvního typu:
  - ”Linen, Beige”(Lněné, béžové)
  - ”Steel, Paint Finish, Dark Gray, Matte”(Ocel, povrchová úprava, tmavě šedá, matná)
- Nábytek druhého typu:
  - ”Steel, Chrome Plated”(Chromovaná ocel)
  - ”Textile – Slate Blue”(Šedě modrý textil)
- Nábytek třetího typu:
  - ”Laminate, Ivory, Matte”(Laminát, slonová kost, matný)
  - ”Steel, Chrome Plated”(Chromovaná ocel)
  - ”Cherry”(Třešeň)
- Dveře prvního typu:

- ”Door – Frame”(Dveře – Rám)
- ”Door – Panel”(Dveře – Výplň)

- Dveře druhého typu:
  - ”Paint – Sienna”(Barva – Sienna)
  - ”Wood – Birch – Solid Stained Light Low Gloss”(Dřevo z břízy masivní, světle zbarvený, matný)

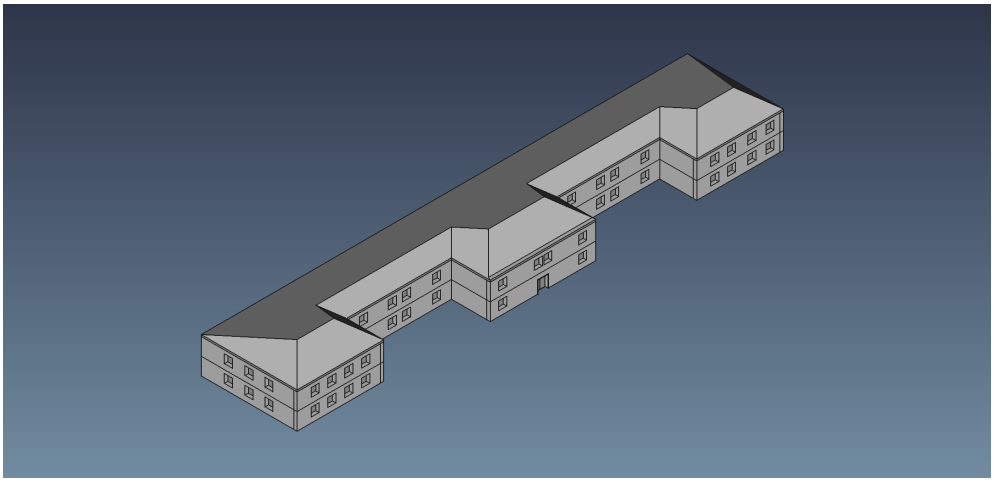
- Okno:
  - ”Glass”(Sklo)
  - ”Sash”(Rámy)

Žádný z materiálu nemá přiřazený materiál ve smyslu počítačové grafiky, to znamená, že materiály bude nutno přiřadit podle informací obsažených v IFC entitách jako je IfcMaterial, IfcMaterialLayer a IfcMaterial. Všechny entity reprezentující materiál obsahují název, ale pokud se podíváme na rozbor jaký objekt obsahuje jaké materiály, můžeme si všimnout, že většina objektů obsahuje více než jeden materiál. Dalším problémem je, že ne všechny názvy obsahují přímo název pro materiál (například “Door – Frame” nebo “Sash”). Proto pokud budeme chtít využít IFC materiály k přiřazení materiálů v Unreal Engine, budeme muset získat i další informace kromě pouze názvu.

K tomu nám pomůže knihova IfcOpenShell, kde její python funkce “ifcopenshell.util.element.get_material()” nám vrátí všechny připojené materiály (ať už se jedná o samotný IfcMaterial, či jednu z množin materiálu). Díky této funkci se můžeme detailně projít atributy, které jsou k materiálům připojeny.

I přesto, že k tomu nebyl tvořen, k výběru preferenčního materiálu by nám mohl sloužit atribut “Priority” obsažený v IfcMaterialLayer. Tento atribut však v našem modelu není pro žádnou z vrstev materiálu nastavený. Atribut, který už nastavený je a bude možno využít jeho informační hodnoty je atribut “LayerThickness”, určující tloušťku vrstvy. U IfcMaterialConstituent by nám mohl pomoci atribut “Fraction”, určující poměr materiálu, ten v našem modelu však také není nastaven. Materiály tak nepůjdou přímočaře přidělit a řešení tohoto problému se budeme blíže věnovat až v návrhu.

Poslední složkou, nutnou k analýze testovacího souboru, jsou obsažené vlastnosti a atributy. Tento model má ke každému objektu přiřazené globální id, tag, název, o jaký typ IFC entity se jedná a také typ objektu (sloužící k seskupování objektů podle typu). Některé elementy mají i informace o své velikosti, kdy například okna a dveře mají nastavenou celkovou výšku a šířku, či schody, které mají nastavené vyvýšení a počet obsažených schodů. Každý element má pak vyplněn své IFC common properties, zmiňované v předešlých sekcích.K detekci, jaké stěny obsahují okna či dveře, je pak možné použít vyplněné atributy “HasOpenings” a “FillsVoids”, které říkají, zdali má element



Obrázek 1.2 Obrátzek ukazující vizualizaci testovacího souboru University Building Model.ifc

(stěna) nějaké otvory a jestli element (dveře nebo okno) je v nějakém otvoru obsažen.

Škálování celého modelu odpovídá podle obsažených rozměrů realitě a jednotky jsou v celém modelu nastaveny na metry. Testovací soubor je ve formátu IFC 4.0.0. a jedná se o model dobře strukturovaný, s velkým množstvím popisných informací a o model konstruovaný z velké míry v souladu se standardy. Vizualizaci geometrické složky modelu můžeme vidět na obrázku 1.2.

1.11 Unreal Engine

Tato sekce se věnuje představení Unreal Engine 5 [15], který je v době psaní této práce na trhu již dva roky a získal si solidní postavení mezi vývojářskými nástroji. Unreal Engine 5 je významně vylepšen oproti svému předchůdci, Unreal Engine 4, a nabízí rozšířené možnosti ve všech aspektech, což jej činí preferovanou volbou pro nové projekty. Tento engine je známý svou schopností zvládat komplexní grafické a simulační úlohy, což jej činí ideálním nástrojem pro naši práci s importem dat. V této sekci se tedy stručně zaměříme na popis Unreal Engine 5.

Unreal Engine 5 je nástroj pro tvorbu 3D grafiky v reálném čase. Unreal engine je od společnosti Epic Games a je k použití zdarma, pokud zisk z vytvořeného díla nepřekročí určitou peněžní hodnotu. Mezi klíčové funkcionality Unreal Engine 5 patří jejich osvětlovací model Lumen, který umožňuje dynamické globální osvětlení a řešení odrazů v reálném čase. Díky této funkci a mnoha dalším tak Unreal Engine umožňuje uživateli vysokou míru realismu ve svých projektech. Unreal Engine 5 se ale také pyšní nástroji na vysokou míru optimalizace. Díky těmto vlastnostem se jedná o vysoce oblíbený herní engine.

1.11.1 Unreal Engine – Způsoby importu

Protože se naše práce zaobírá převážně importem, v této podsekci se zaměříme na možnosti importu, které Unreal Engine 5 nabízí, s důrazem na import souborů ve formátu IFC.

1.11.1.1 Běžný import

Unreal Engine 5 nejlépe pracuje s formátem modelů FBX a pro import těchto souborů je možné použít již vestavěný import FBX modelů. Tento import lze vyvolat v sekci se jménem Content Drawer, kde po kliknutí na tlačítko "Import" se zobrazí prohlížeč souborů našeho hardwaru. Po vybrání námi zvoleného FBX modelu se zobrazí dialogové okno s možnostmi, zdali chceme přidat transformaci modelu, jak se má nakládat s texturami a materiály, zdali se mají všechny části modelu spojit do jedné, zdali se mají vygenerovat kolize, nebo také například zdali se má model importovat jako Nanite (Speciální Unreal Engine struktura modelu, která slouží k optimalizaci renderování). Import po zvolení vytvoří takzvaný "static mesh" pro každou část modelu (pokud nebyly u importu spojeny) a pokud bylo zvoleno importování textur, importují se i textury jako materiály.

Problémem však je, že náš model je ve formátu IFC. Model IFC, jak už bylo zmíněno, se skládá z geometrické a informační složky. Geometrická složka se pak dá převést na formát FBX a tak dosáhnout jejího úspěšného importu. Tento převod se dá například provést za pomoci softwaru Blender a zásuvného modulu BlenderBIM. Takto se dá ale importovat pouze geometrická složka modelu a bylo by nutné použít dalších nástrojů pro import složky informační.

1.11.1.2 Import za pomoci kódu

Unreal Engine 5 poskytuje rozsáhlé API pro import modelů, a to i za pomoci zdrojového kódu. API je dostupné v jazyce C++, na kterém je Unreal Engine postavený, ale i v jazyce Python, který je v Unreal engine převážně používán ke skriptování a automatizaci.

1.11.1.3 C++

> *„Unreal Engine je převážně napsán v jazyce C++ a je rozdělen do modulů. Každý modul nese určitou funkcionalitu a je pro uživatele dostupný a využitelný i v jiném modulu. Tento přístup tedy umožňuje programátorovi zahrnutím odpovídajícího modulu do jeho kódu využívat jakékoliv veřejné funkcionality, které využívá Unreal Engine.“*
>
> [16]

Jak říká citace, Unreal Engine je postavený z modulů, které uživatel může dále využít. K importu souborů se dá využít modul "UnrealEd" s třídou

“FFbxImporter”, který zprostředkovává funkcionality běžné pro import modelu ve formátu FBX. Tato cesta umožňuje import automatizovat, ale stále se potýká s problémem převodu IFC formátu do FBX formátu.

Pokud máme informace o geometrii v podobě obsažených trojúhelníků, vrcholů, normál a tak dále, můžeme však také využít modulů “MeshDescription” s třídou “FMeshDescription” v kombinaci s modulem “Engine” s třídou UStaticMesh k vytvoření geometrie přímo v Unreal Engine. K získání geometrických dat, ale i dat informačních ze souboru IFC, lze pak využít knihovny IFC++ a IfcOpenShell pro C++, o kterých se více zmíníme později.

1.11.1.4 Python

V dnešní době je nedílnou součástí Unreal Engine i skriptování za pomoci jazyku Python, který má na starosti spoustu procesů Unreal Enginem používaných. Python má také rozsáhlé API, které je možné využít k importu, či k vytvoření geometrie přímo v Unreal Engine. Klíčovou třídou je takzvaná “AssetTools”. AssetTools nám poskytuje funkce import_asset používanou k běžnému importu a creat_asset k vytvoření geometrie.

Pro běžný import se dále musí použít třída FbxImportUI, která zajistí správné nastavení možností pro import. Pokud se místo importu bude geometrie přímo vytvářet, bude podobně jako u C++ potřeba získat o geometrii data. Pro získání dat je možné použít knihovnu IfcOpenShell, která je vytvořena i pro Python.

1.11.1.5 Datasmith

Všechny dosud zmíněné postupy přinášejí komplikace spojené se získáváním geometrických dat z IFC souborů nebo s převodem formátů. Jako řešení slouží oficiální zásuvný modul Datasmith. Datasmith je integrován přímo do funkcionalit Unreal Engine a nabízí rozšíření pro import CAD souborů, včetně formátu IFC.

Datasmith byl vytvořen jako cesta k importu celých scén do Unreal Engine při zachování jejich struktury. Pokud by byl použit některý z dříve zmíněných přístupů, musely by být objekty importovány po jednom a v Unreal Engine pak zase skládány do původní struktury. Dalším přístupem by bylo celý objekt spojit a importovat jako jeden velký model, to by poté však zkomplikovalo práci s jednotlivými elementy modelu v Unreal Engine. Datasmith všechny tyto problémy řeší importem celé scény do struktury zvané “Datasmith scene“, která udržuje původní strukturu. [17]

Při importu vznikne v Unreal Engine “Datasmith scéna”, složka s materiály, složka s geometrií a případně například složka s texturami (záleží, co vše importovaná scéna obsahuje). Datasmith scéna je speciální soubor v Unreal Engine, který, jak už bylo řečeno, obsahuje všechny informace o hierarchii modelu. Obsahuje data, jak jsou všechny objekty uspořádány ve scéně, jak jsou přiřazeny materiály, textury a další potenciálně importované informace

(například světla). Celou Datasmith scénu je možno přetáhnout do jakékoliv Unreal Engine úrovně a tím celý importovaný model s dodrženou hierarchií zobrazit. Celou hierarchii zastřešuje ve scéně takzvaný "Datasmith Scene Actor", který slouží pouze jako obálka pro další objekty. Pod "Datasmith Scene Actor" se nachází soustava takzvaných "Actors", která slouží k dodržení hierarchie z IFC. Hierarchie se snaží být dodržena co nejvěrněji společně s použitými názvy. Pro zobrazení samotného geometrického prvku ve scéně slouží speciální actor, takzvaný "Static Mesh Actor". Ten do hierarchie vkládá informace o tom, jaký element (Static Mesh) má být na jakém místě použit. Pro maximalizaci optimalizace Datasmith jakékoliv objekty se stejnou geometrií importuje pouze jako jeden "Static Mesh", který je pak přepoužíván vícekrát. Static Mesh Actor obsahuje mimo geometrii také transformaci, říkající, v jaké formě má být daný Static Mesh použit.

"Datasmith Scene" také obsahuje funkci nazývanou "roll back", kdy všechny změny, provedené v geometrii, v tagu, či v přiřazení materiálů, textur, světla jsou zaznamenávány. To umožňuje uživateli navrátit importovanou scénu do původní podoby. Pokud dojde k jakékoliv změně zdrojového IFC souboru, Datasmith umožňuje importovanou scénu znovu načíst. Znovunačtení probíhá se snahou o co nejmenší ztrátu odvedené práce uživatele. Proto všechny změny a použité instance scény se snaží být zachovány. [17]

Datasmith má tři základní cesty použití. První cesta požaduje kromě zásuvného modulu Datasmith pro Unreal Engine i stažení zásuvného modulu Datasmith pro aplikaci, ze které chceme scénu importovat. Zásuvný modul je například dostupný pro SketchUp nebo 3ds Max. Datasmith je v tomato přístupu použit nejprve k exportu scény ve formátu udatasmith. Udatasmith je formát, se kterým Datasmith nativně pracuje. Proto následný import za pomoci Datasmith v Unreal Enigne probíhá velice přímočaře.

Druhou cestou je využití formátů, které datasmith již podporuje. Těmi jsou například takzvané nativní CAD formáty. V tomto případě, protože je formát podporovaný, není potřeba používat žádných dodatečných zásuvných modulů k exportu, ale je možné použít originální formát a pouze scénu importovat za pomoci Datasmith v Unreal Engine.

Třetí cesta využívá podobný přístup, jako cesta první. Pokud požadovaná aplikace nepodporuje oficiální Datasmith zásuvný modul a Datasmith nepodporuje formát pro aplikaci nativní, je možné použít zásuvné moduly třetí strany pro export ve formátu FBX. Datasmith, stejně jako běžný Unreal Engine import, podporuje formát FBX. Proto je následný import celé FBX scény za pomoci Datasmith přímočarý.

Datasmith importer má v Unreal Engine své uživatelské rozhraní, ale je možné funkcionality využít i ve zdrojovém kódu. Unreal Engine poskytuje API s Datasmith funkcionalitami pro C++, pro Python, a některé funkce poskytuje i pro Unreal Engine blueprint.

Pro import souboru ve formátu IFC je nutné použít Datasmith rozšíření, nazvané "Datasmith CAD", kdy je funkcionalita rozšířena o podporu IFC

souborů. Datasmith CAD umožňuje import celé struktury, ale i velké části IFC vlastností. Importované vlastnosti jsou v Unreal Engine uložené přímo k objektům struktury za pomocí speciálních metadat nazvaných "Datasmith User Data". I přes vysokou míru importovaných vlastností, Datasmith neimportuje všechny informace, ani všechny informace využitelné k naší práci. Například nedochází k importu materiálů přiřazených pomocí entity IfcMaterialConstituentSet a dochází tak k importu pouze určitých skupin materiálů. Další užitečnou neimportovanou informací je informace o vložených elementech. Například vložení elementu dveří do elementu stěn. Vlastnosti jako jsou běžné IFC proměnné, id, název, tag nebo typ ifc entity, jsou všechny v importu obsaženy.

Import pomocí Datasmith umožňuje nastavení určitých možností importu. První skupina určuje, co vše bude v importu zahrnuto. Z důvodu importu celé scény mohou být přítomny i jiné prvky, než pouze model a jeho materiály, proto v importu mohou být zahrnuty: geometrie, materiály a textury, světla, kamery a animace. Druhá skupina možností importu obsahuje vytvoření takzvaných Lightmap UVs, které slouží k výpočtu osvětlení a tvorbě stínů. Možnosti nabízejí, zdali se mají Lightmap UVs pro model vygenerovat a rozsah jejich kvality. Pokud není model již tvořen trojúhelníkovou sítí (jedná se například o parametrické modely z CAD), je nutné model takzvaně teselovat, aby mohl být v Unreal Engine vykreslený bez komplikací. Datasmith při importu umožňuje ovlivnit i nastavení tohoto procesu. Při teselaci dochází k ovlivnění geometrie, a proto může být nastavená odchylka, o kterou se může výsledný model a jeho normály lišit. Dále je možné nastavit, jak maximálně velké mohou trojúhelníky tvořící síť být (slouží pro kontrolu přesnosti), či zdali a jak moc se mají sousední plochy spojovat (více spojení vede k větší optimalizaci, ale současně k větší odchylce od původní geometrie).

1.12 Knihovny pro práci s IFC

Protože v sekci o importu do Unreal Engine jsme zmiňovali možnosti využívající knihovny pro práci s formátem IFC, v této sekci se na ně stručně zaměříme.

1.12.1 IFC++

IFC++ [18] je open source projekt, který podléhá licenci MIT. Tento projekt poskytuje knihovnu, která se snaží o efektivní práci s formátem IFC, což zahrnuje import i export. IFC++ se podle jejich dokumentace specificky zaměřuje na standard IFC STEP, zkratka pro Standard for the Exchange of Product Model Data. STEP využívá metodologie a strukturu definovanou v ISO 10303, která je však standardem podle BuildingSMART. [18]

IFC++ poskytuje mimo knihovnu také aplikaci SimpleViewerExampleQt, která slouží jako demo funkcionalit knihovny.

1.12.2 IfcOpenShell

IfcOpenShell [19] je open source sada nástrojů pro práci se soubory IFC. IfcOpenShell nabízí rozsáhlé API jak pro C++, tak pro python. Příklady jeho využití pro C++ můžeme vidět u dříve zmíněných zásuvných modulů pro FreeCAD. Příklad využití pro Python lze nalézt v zásuvném modulu BlenderBIM pro Blender, který je přímo od IfcOpenShell.

Jádro IfcOpenShell se nachází v jeho C++ verzi a python pouze zprostředkovává přístup k těmto C++ nástrojům. Všechna funkcionalita, která je obsažena v C++, je však i v IfcOpenShell-Python. IfcOpenShell a IfcOpensShell python jsou licencovány pod LGPL-3.0-or-later.

Mezi hlavní funkcionality podle dokumentace patří:

”

- *Prohlížení modelů, včetně prostor, vlastností a vztahů*
- *Úpravy a extrakce atributů a vlastností*
- *Přesun objektů a změna jejich geometrie*
- *Tvorba nových objektů s použitím prvků knihovny*
- *Správa systémů klasifikace, dokumentů a knihovních odkazů*
- *Generování 2D výkresů, harmonogramů a vytváření listů v tabulkách*
- *Prozkoumání a úprava modelů strukturální analýzy*
- *Připojení a správa distribučních systémů a portů*
- *Tvorba stavebních harmonogramů, analýza kritických cest a generování sekvencí animace*
- *Tvorba nákladových rozvrhů, použití vzorců a odvozování množství z prvků modelu*
- *Detekce kolizí a správa problémů pro koordinaci modelu*

“

[19]

1.13 Přiřazení materiálů v Unreal Engine

Tato práce se kromě importu Building Information Modeling do Unreal Engine zaobírá i následnou vizualizací modelu. Hlavní složkou vizualizace bude přiřazení materiálů a textur. Proto se v této sekci podíváme na možnosti přiřazení materiálů z modelu Building Information Modeling.

1.13.1 Materiál v Unreal Enigne

Nejdříve se podíváme na reprezentaci materiálu v Unreal Engine. Unreal Engine používá materiál jako definici vizuálního vzhledu povrchu. Tato definice se může skládat z více složek jako je barva, lesklost, průhlednost, hrubost a mnoho dalších. Do materiálu se mohou zahrnout i textury, či jakékoliv mapy určující vlastnosti povrchu.

Všechny prvky se nastavují v editoru materiálu za pomocí systému uzlů. Klíčovým uzlem je takzvaný "Main Material Node", který přebírá jednotlivé složky jako vstupy. Do těchto vstupů se pak napojují buďto textury, mapy, či číselná data, které říkají, jak se výsledný materiál má chovat.

Napojená data se mohou také parametrizovat. Parametry je možné nastavovat ve vytvořených instancích materiálu. Díky tomu je pak možné stejný materiál s menšími úpravami přepoužít za pomoci vytvořených instancí materiálu.

Takto vytvořené materiály Unreal Engine automaticky zkompiluje do shaderů, které jsou napsány v HLSL, zkratka pro High Level Shading Language. Shadery pak slouží při vykreslování, kdy je Unreal Engine volá pro zpracování vizuálních vlastností materiálu. [20]

1.13.2 Možnosti přiřazení

Jak již bylo zmíněno v analýze testovacích souborů, naše soubory IFC neobsahují žádné grafické materiály, které by mohly být importovány a pouze přiřazeny. Proto bude nutné materiály přiřadit pouze za pomocí obsažených vlastností.

1.13.2.1 Přiřazení za pomoci IfcMateial

Při přiřazení za pomoci IfcMaterial čelíme několika problémům. Prvním problémem jsou elementy, které obsahují více než jeden materiál. Ty se v našich testovacích datech dělí na dvě skupiny – elementy, které obsahují více vrstev materiálů a elementy které mají různě části z různých materiálů.

Elementy z různých vrstev si můžeme představit jako stěnu, která je tvořena z vícero úrovní, například základ, omítka, zateplení. Tyto materiály jsou přiřazeny za pomocí IfcMaterialLayerSet. Jako materiál k přiřazení by pak bylo vhodné vybrat vrstvu vnější, protože ta je jako jediná viditelná. K rozeznání vnější vrstvy v Industry Foundation Classes nejsou přímo určené žádné datové struktury, ale zpravidla se tato informace zaznamenává pomocí pořadí materiálů v množině (například omítka, cihly, omítka).

Elementy, které mají různé části z různých materiálů jsou přiřazeny za pomoci IfcMateialConstituentSet. Při tomto typu přiřazení je objekt většinou složen z více částí, kdy každá část obsahuje jiný materiál. Příkladem může být například židle, kdy konstrukce je tvořena z kovu, ale například opěradlo a sedadlo je polstrované a pokryté textilem. V tomto případě však zpravidla

nedochází k žádnému významu pořadí, ani Industry Foundation Classes neobsahují žádné specializované datové struktury k určení, jaký materiál patří k čemu. Industry Foundation Classes obsahuje sice popis materiálu, kde tyto informace mohou být zaznamenány, ale naše testovací soubory popis vyplněný nemají ani zde není žádný standardizovaný formát k jednoduché automatizaci. Dalším problémem tohoto přiřazení je rozlišení různých částí geometrie. Industry Foundation Classes neobsahuje žádné atributy, které by říkaly, že tato část modelu je opěradlo, ani že židle opěradlo vůbec obsahuje.

Pokud by se nám podařilo správně přiřadit materiály, dalším problémem, kterému čelíme jsou názvy materiálů. Materiály v našich testovacích souborech jsou velmi specifické, každý má jinou strukturu a občas se nejedná ani o materiál, ale pouze o část daného objektu. Příklad velmi specifického názvu je Steel, Paint Finish, Dark Gray, Matte. Takhle definovaný materiál by musel být rozebrán na části. Části by pak musely být identifikovány ve smyslu jaká část ovlivňuje jakou složku materiálu (například kovovost materiálu, barvu materiálu, a tak dále). Tento proces by byl velmi náročný až nemožný, protože testovací soubory neudržují žádnou definovanou strukturu materiálu, a tak materiál může obsahovat cokoliv (například u okna je jeden materiál definovaný pouze jako rám). Možným řešením by však mohla být umělá inteligence.

Umělá inteligence by se dala použít dvěma způsoby. Prvním je využití umělé inteligence pro generování obrázků, které by se pak použily jako textury tvořící materiál. Druhým je přiřazení komplikovaných a rozsáhlých názvů k již vytvořeným základním materiálům. V obou případech by musela být testovací data specifického rázu a rozsáhlá, aby bylo možné pokrýt všechny materiály, které se mohou v modelu nacházet.

1.13.2.2 Použití umělé inteligence pro generování obrázků jako textur

Ke generování textur za pomoci umělé inteligence již některé aplikace existují. Jako příklad si můžeme uvést Texture Lab [21] od Hugo Duprez, či AI Texture Generator [22] od Polycam. Obě tyto aplikace slouží ke generování volně využitelných 3D textur, uzpůsobených k použití v 3D grafických aplikacích.

Bohužel ani jedna z obou aplikací neposkytují žádné API, které by mohlo být v našem projektu integrováno a jsou určené pouze k použití skrze jejich zamýšlené uživatelské rozhraní. Ani jedna z těchto aplikací také nepopisuje AI model, který byl ke generování použit.

Z analýzy známých AI modelů pro generování obrázků by pro náš problém bylo nejlepší použít Transformer Model. Transformer Model se specializuje na kombinaci pochopení kontextu z textu a následné generování obrázku. Transformer Model používá například AI generátor obrázků DALL-E od OpenAI, ale i samotný Chat GPT.

1.13.2.3 Přiřazení komplikovaných názvů k materiálům

Pokud se jedná o přiřazení komplikovaných názvů k materiálům, žádnou aplikaci řešící tuto problematiku se mi nepovedlo nalézt, aplikace by totiž musela být velice specificky navržená pro náš problém.

Řešením by mohlo být použití strojového učení, které je hojně využíváno ke klasifikaci. Mezi algoritmy strojového učení, které bychom mohli využít, patří například logistická regrese a SVM, zkratka pro Support Vector Machines, které by měli být dostatečné pro řešení našeho problému.

Hlavním problémem budou pak testovací data, které budou muset obsahovat velké množství údajů, jako jsou všemožné typy stromů, typy kovů, textil, stavební materiály jako cihly, sádrokarton a tak dále, které budou použity pro klasifikaci. Testovací data budou muset být označena materiály, které budou v Unreal Engine dostupné.

1.13.2.4 Přiřazení materiálů za pomoci IFC entit

Protože materiály nemají žádnou danou strukturu, ani nemají dané, k jaké části objektu jsou přiřazeny, jejich přiřazení bude velmi náročné bez použití lidské práce, či aspoň výkonného modelu umělé inteligence. Všechny objekty jsou však v IFC souborech rozčleněny do IFC entit. Každý objekt může obsahovat pouze jednu IFC entitu, a až na proxy entity, používané v době, kdy objekt nepatří do žádné předdefinovaných entit, jsou všechny IFC entity předem definované. Jak již bylo řečeno výše, taktiku vizualizace za pomoci IFC entit také využívá velké množství softwarů o vizualizaci se snažících.

Řešením v tomto případě by tak bylo definování, k jaké IFC entitě bude přiřazen jaký materiál, s možností uživatele toto nastavení upravit a následně přiřazovat materiály podle těchto nastavení.

1.14 Obsažené funkcionality a použité technologie

1.14.1 Import IFC souboru

Pro import soboru je potřeba použít jeden ze zmiňovaných způsobů v podsekci 1.11.1 jakožto zvolený způsob importu byl zvolen zásuvný modul Datasmith, přesněji Datasmith CAD. Hlavními důvody jsou obsažená podpora souborů ve formátu IFC a obsah všech potřebných funkcionalit pro import. Dalším důvodem je nativnost zásuvného modulu a z toho vyplývající dobrá kompatibilita s funkcemi Unreal Engine společně s dostupným API jak pro C++, tak pro Python.

1.14.2 Dodatečná práce s IFC souborem

Protože Datasmith neimportuje úplně všechny atributy, bude nejspíše nutné využít dodatečné knihovny pro práci s IFC souborem. Jako knihovna pro tuto

operaci byla zvolena knihovna IfcOpenShell, přesněji knihovna pro Python. Tato volba byla uskutečněna z důvodu rozsáhlosti funkcí obsažených v knihovně a možnosti použití knihovny s jazykem Python.

Python byl pak vybrán z důvodu možnosti skriptování bez sestavení celé aplikace a jednoduššího importu Python knihovny do Unreal Engine oproti importu C++ knihovny.

Jakožto atributy, které by mohly být užitečné, a proto doimportované do Unreal Engine, byly vybrány atributy materiálů (hlavně IfcMaterialConstituentSet, který není pomocí Datasmith importován) a dvojice atributů "FillsVoids" s "HasOpenings", které je možné využít k identifikaci stěn se dveřmi a okny.

1.14.3 Přiřazení materiálů

Materiály budou přiřazovány primárně za pomoci typů IFC entit. Jako sekundární možnost bude využito i přiřazení materiálů za pomoci entit IfcMaterial, ale bude nutný vstup uživatele.

1.14.4 Použité Technologie

Vývoj bude probíhat v Unreal Engine, protože to požadoval zadavatel. Specificky pak bude použit Unreal Engine 5, jakožto nejmodernější verze Unreal Engine.

Funkcionalita v Unreal Engine bude implementována za pomocí kombinace Python, C++ a Blueprint. Blueprinty jsou předdefinované funkce obsažené v Unreal Engine modulech stejně jako všechny ostatní Unreal Engine funkce. Rozdílem je, že tyto funkce jsou dostupné i pro Blueprint prostředí. S dostupnými Blueprint funkcemi, takzvanými Blueprint uzly, je možné dané Blueprint uzly spojovat v grafech, a tak vytvářet funkcionality na úrovni funkcionalit dosažených v kódu. Velkými výhodami Blueprint technologie je jednoduchost použití, grafická reprezentace a možnost snadného ladění chyb. Bluepirnt je bezproblémově integrován do Unreal Engine, a tak nepřináší téměř žádná omezení. Pro rozšíření funkcionalit se může použít i v kombinaci s C++, jehož vytvořené funkce můžeme zpřístupnit Blueprint prostředí a tak funkce využívat. Další výhodou je i možnosti spouštění Python skriptů přímo z Blueprint grafu.

1.15 Požadavky

V této sekci se zaměříme na funkční a nefunkční požadavky, které by měl náš Unreal Engine projekt splňovat. Rozsah požadavků byl sestaven podle zadání této práce a upraven za pomoci konzultací s vedoucím této práce.

1.15.1 Funkční požadavky

V této sekci se budeme zabývat funkčními požadavky. Ty definují, jaké funkce bude vytvářený Unreal Engine projekt obsahovat a nabízet uživateli. Funkční požadavky, zde zmíněné, jsou rozděleny na dvě části. První část se týká funkčních požadavků z pohledu uživatelského rozhraní a druhá se zabývá vnitřními funkcionalitami projektu.

1.15.1.1 Uživatelské rozhraní

V této podsekci se budeme zaobírat funkčními požadavky úzce spjatými s vytvářením uživatelského rozhraní. Proto popisují funkce, které v něm budou přímo obsažené.

F1: Umožnění importu IFC modelu – uživatelské rozhraní poskytne uživateli možnost importu modelů ve formátu IFC. Import bude nabízet nastavení možností importu uživatelem, či nastavení předdefinovaných parametrů pro optimální import.

F2: Zobrazení importovaných modelů – pro následující práci s importovaným modelem uživatelské rozhraní nabídne přehled importovaných modelů, které jsou dostupné k úpravě.

F3: Zobrazení dostupných materiálů k dodatečnému importu – uživatelské rozhraní zobrazí přehled materiálů, které jsou obsaženy ve zdrojovém souboru. Tímto uživateli poskytne možnost volby importu pouze relevantních materiálů.

F4: Přiřazení materiálů – uživatelské rozhraní poskytne možnost automatického přiřazení materiálů k jednotlivým objektům. Uživatelské rozhraní nabídne dva módy této akce: Mód s použitím pouze IFC entit k rozřazení materiálů a druhý mód, kdy, pokud to bude možné, bude upřednostněno přiřazení podle IFC materiálu.

F5: Generování světel – uživatelské rozhraní umožní generování světel ke stropům objektu (pokud jsou přítomny). Uživateli nabídne možnost výběru svítivosti a barvy světla.

F6: Odstranění vygenerovaných světel – uživatelské rozhraní umožní opět odstranit všechna vygenerovaná světla.

F7: Přiřazení kolizí pro průchod scény – uživatelské rozhraní umožní automatické přiřazení jednoduchých kolizí pro umožnění průchodu scény z pohledu první osoby.

1.15.1.2 Funkce Unreal Engine Projektu

Funkce zde zmíněné nemají žádný větší vliv na tvorbu uživatelského rozhraní, nýbrž na funkcionalitu procesů na pozadí.

F8: Podpora kompletních i nekompletních IFC souborů – Unreal Engine projekt umožní import jak IFC souborů, které jsou plně definované podle standardů Building Information Modeling, tak IFC souborů, které velkou

část atributů a vlastností definovanou nemají. Mimo import je umožněna i manipulace s oběma typy souborů, avšak v případě nekompletních souborů je velice omezena.

F9: Ukládání importovaných modelů do Unreal Engine – při importu do místa, kde se již model se stejným názvem nachází, dojde k nahrazení starého modelu modelem novým.

F10: Úprava přiřazených materiálů – projekt umožní uživateli jednoduchou úpravu použitých materiálů a pravidel pro přiřazení za pomoci uživatelského rozhraní Unreal Engine.

F11: Speciální kolize – projekt umožní pro určité typy objektů, specificky pro stěny obsahující dveře, speciální řešení kolizí pro umožnění průchodu celé budovy.

1.15.2 Nefunkční požadavky

V této podsekci se zaměříme na nefunkční požadavky, které nám určují požadované kvality projektu, ale také jeho omezení.

NF1: Technologie – řešení bude vypracováno jako Unreal Engine Projekt v Unreal Engine 5.1.1. Hlavní funkcionalita bude obsažená v uživatelském rozhraní určeném pro úpravy v editoru (mimo běh projektu). K vývoji bude použita kombinace Python skriptů, C++ kódu a vše bude sjednoceno za pomocí blueprint technologie.

NF2: Dokumentace – kód i blueprint uzly budou náležitě okomentovány s dostupnou uživatelskou příručkou k použití.

NF3: Zaobalení – z důvodu potenciálu pro použití funkcionalit projektu k vytvoření samotného zásuvného modulu budou hlavní prvky projektu zaobaleny a drženy oddělené v samostatných složkách. Tento přístup přispěje ke snadné přeměně na zásuvný modul.

NF4: Rozšiřitelnost funkcionalit – protože se projekt zabývá pouze základními funkcemi zaměřenými na základní vizualizaci a import, projekt bude koncipován tak, aby byl lehce rozšiřitelný.

Kapitola 2
Návrh

V této kapitole se zaměříme na návrh vyvíjeného projektu v Unreal Engine. Návrh bude inspirován požadavky a bude vycházet z analýzy možností a omezení Unreal Engine a IFC souborů.

2.1 Definice pojmů

Pro lepší pochopení následujících postupů se v této kapitole podíváme na definici pojmů v návrhu praktické části využívaných.

Asset – Asset je soubor v Unreal Engine, který má koncovku .uasset. Tento soubor může obsahovat téměř cokoliv, od materiálů, tabulky, obrázky, hudbu až po jednotlivé modely. Soubor je ve formátu, se kterým Unreal Engine umí pracovat a všechny obsažené assety je možné najít v takzvaném content browser.

Blueprint třída – Blueprint třída je speciální typ assetu. Jeho cílem je rozšířit Unreal Engine funkcionalitu ve stejném principu jako C++ třída. Místo C++ se v této třídě využívá vizuální programování, za pomoci takzvaných blueprint uzlů. Blueprint uzly se ve třídě spojují dohromady a tím vytváří složitější funkcionality.

Parent blueprint třída – Ve stejném smyslu jako C++ třída může dědit své vlastnosti a funkcionality od rodičovské třídy, může i blueprint třída dědit své vlastnosti od rodiče. Parent blueprint třída je tedy blueprint třída, pod kterou je možné vytvořit třídu vlastní a tím zdědit dané vlastnosti.

Blueprint uzel – Blueprint uzel představuje C++ funkci v blueprint světě. Všechny C++ funkce zpřístupněné blueprint jsou dostupné jako blueprint uzly.

Widget – Widget je blueprint třída, která umožňuje vytváření uživatelského rozhraní pro využití v Unreal Engine.

Editor utility widget – Editor utility widget je speciální typ widgetu, který je dostupný pouze v editoru. Tím může využívat funkcionalit, které nejsou určeny pro užívání za běhu aplikace.

Static mesh – Static mesh je typ assetu reprezentující geometrii jednoho objektu v Unreal Engine.

Actor – Actor reprezentuje cokoliv, co je určeno pro existenci ve scéně. Pokud je například static mesh ve scéně, musí být pouze jako komponenta nějakého actor. Zpravidla static mesh actor.

String table – String table je tabulka se strukturou klíč a hodnota. Je v ní možné pomocí klíče vyhledávat.

Datasmith scene – Datasmith je zaměřený na import kompletních scén, proto datasmith scene je asset, reprezentující strukturu celé importované scény.

2.2 Diagramy

V této sekci se podíváme na diagramy reprezentující případy užití. Diagramy reprezentují jeden souvislý případ užití, který z důvodu rozsáhlosti byl rozdělen na více menších diagramů. Pro zachování návaznosti se diagramy vždy odkazují na diagram následující.

Diagramy, ukázané v této sekci, popisují nejprve import modelu IFC 2.1 s následným importem informací o materiálech 2.2. Poté je v diagramech ukázán možný průběh přiřazení materiálů 2.3 s vygenerováním světel 2.4 a nastavením kolizí 2.5. Poslední diagram zobrazuje konečnou část, a to průchod importovanou budovou z pohledu první osoby. 2.6

2.2.1 Diagram Importu

V tomto diagramu, zachyceném na obrázku 2.1, je popsán možný průběh importu modelu. Uživatel otevře projekt obsahující vytvořenou funkcionalitu a otevře vytvořené uživatelské rozhraní. V uživatelském rozhraní bude mít přehled o již importovaných modelech, ale uživatel si vybere importovat model nový. Po výběru importu nového modelu se otevře uživatelské rozhraní s možností zadání cesty k IFC modelu. Jako další nabídne uživatelské rozhraní na výběr možnost importu s nastavením či bez nastavení parametrů. Při výběru importu s nastavením parametrů se otevře uživatelské rozhraní, ve kterém má uživatel možnost nastavit parametry pro import modelu, v opačném případě (při výběru importu bez nastavení parametrů) vyplní parametry funkcionalita projektu. Po vyplnění všech potřebných parametrů dochází k využití Datasmith zásuvného modulu k importu modelu a vytvoření všech potřebných souborů. V případě selhání je uživatel upozorněn ve stále otevřeném uživatelském rozhraní. V případě úspěchu pokračuje uživatel s importem materiálů 2.2.2. V uživatelském rozhraní, zmíněném v minulé sekci 2.2.1 si uživatel ze seznamu modelů vybere nově importovaný model IFC. Projekt pak otevře uživatelské rozhraní přímo pro daný importovaný model. Zde si uživatel vybere import materiálu. Podle toho, zdali IFC soubor obsahuje nějaké materiály či ne, budou zobrazeny dané materiály či hláška o nepřítomnosti materiálů. Uživatel

je vybídnut k výběru materiálů, jejichž informace chce do Unreal Engine importovat a následně import započne. Všechny importované materiály jsou pak přiřazeny k odpovídajícím objektům importovaného modelu a pokud se již nenachází v tabulce pravidel (tabulka určující jaký materiál v Unreal Engine je přiřazen k materiálu IFC), jsou do ní vloženy. Následně pokračuje scénář diagramem přiřazení materiálů 2.2.3.

2.2.2 Diagram Importu Materiálů

V tomto diagramu, zachyceném na obrázku 2.2, pokračuje uživatel v diagramu Importu modelu 2.2.1 importem informací o materiálu.

2.2.3 Diagram Přiřazení Materiálu

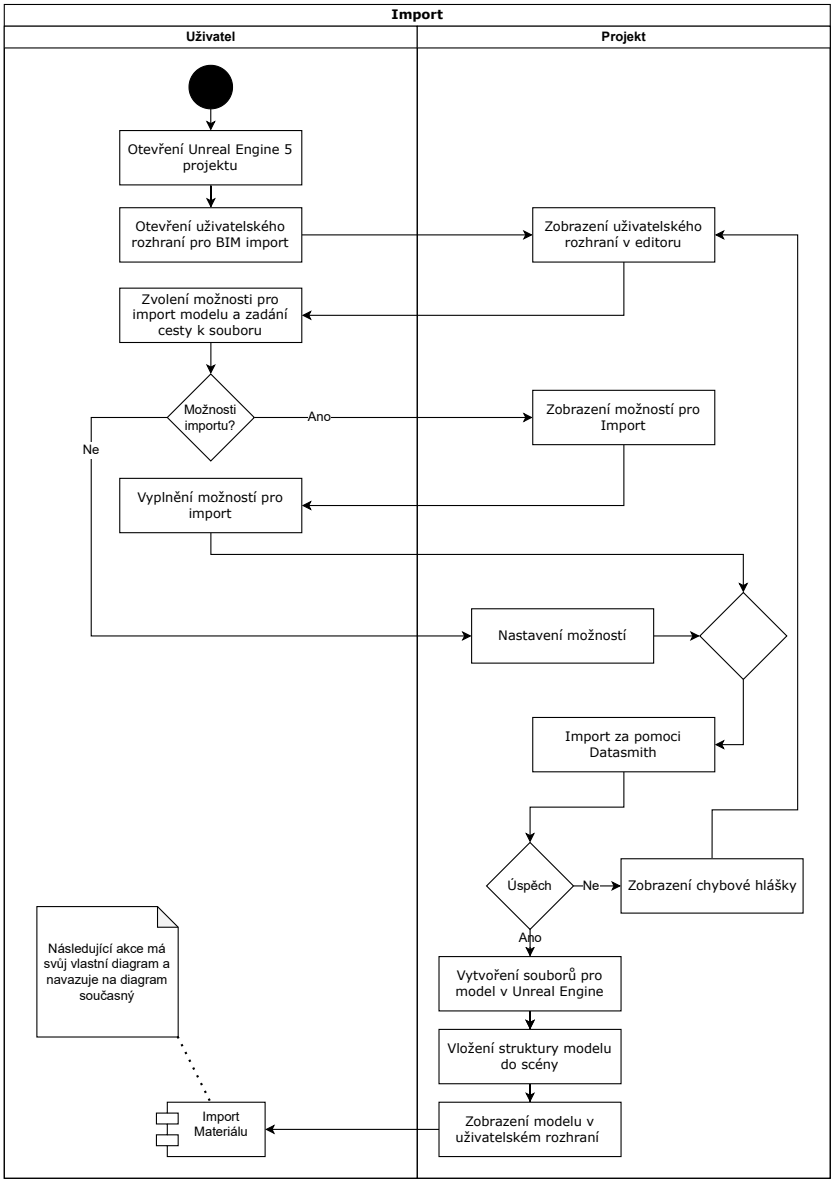
V tomto diagramu, zachyceném na obrázku 2.3, pokračuje uživatel v diagramu 2.2 přiřazením Unreal Engine materiálů k importovaným objektům. Aby mohlo přiřazení proběhnout podle očekávání, uživatel nejprve otevře tabulku s pravidly (zmíněnou v minulé kapitole 2.2.1), kde je potřeba přiřadit správné Unreal Engine materiály k vloženým IFC materiálům. Po zadání pravidel do tabulky uživatel uloží změny a vrátí se do uživatelského rozhraní pro importovaný model. Zde má uživatel na výběr mezi pravidly IFC entit a pravidly IFC materiálů. Při výběru pravidel IFC materiálu se projekt snaží použít importované IFC materiály společně s nalezenými pravidly. Pokud však materiály či pravidla nejsou nalezeny, použije se automaticky tabulka pro IFC entity. Přiřazení tak probíhá podle jedné z těchto tabulek. Pokud však selže i tabulka pro IFC entity, použije se základní výchozí materiál. Po určení Unreal Engine materiálu, který bude použit, projekt materiál přiřadí k daným objektům. Scénář dále pokračuje generováním světel 2.4.

2.2.4 Diagram Generování Světel

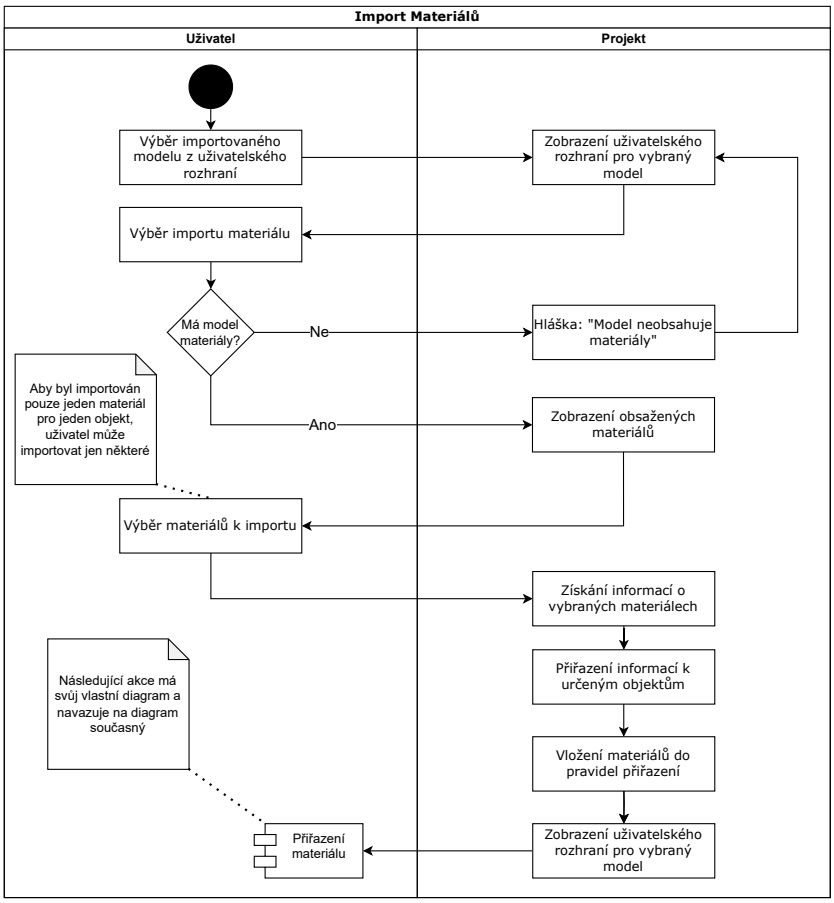
V tomto diagramu, zachyceném na obrázku 2.4, pokračuje uživatel v diagramu 2.2.3 generováním světel. Uživatel, ve stále otevřeném uživatelském rozhraní pro importovaný IFC model, vybere možnost generování světel. V uživatelském rozhraní uživatel nastaví svítivost společně s barvou světla a spustí generování světel. Projekt pak projde celý IFC model a vyhledává jakýkoliv objekt označený IFC entitou jakožto strop. Pokud takový objekt najde, vygeneruje bodové světlo a přiřadí ho k objektu. Světlo je následně modifikováno za pomoci zadaných parametrů. Scénář dále pokračuje nastavením kolizí 2.2.5.

2.2.5 Diagram Nastavení Kolizí

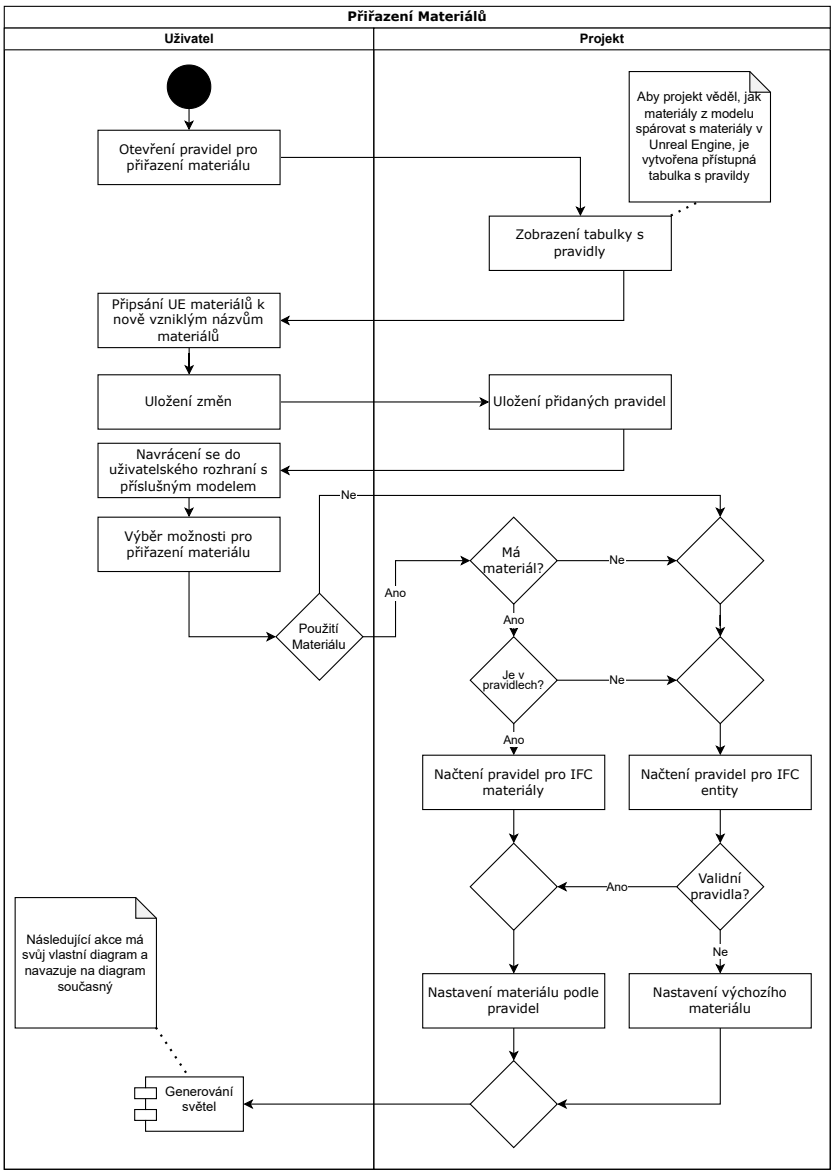
V tomto diagramu, zachyceném na obrázku 2.5, uživatel vybere nastavení kolizí (stále ve stejném uživatelském rozhraní pro importovaný IFC model). Projekt



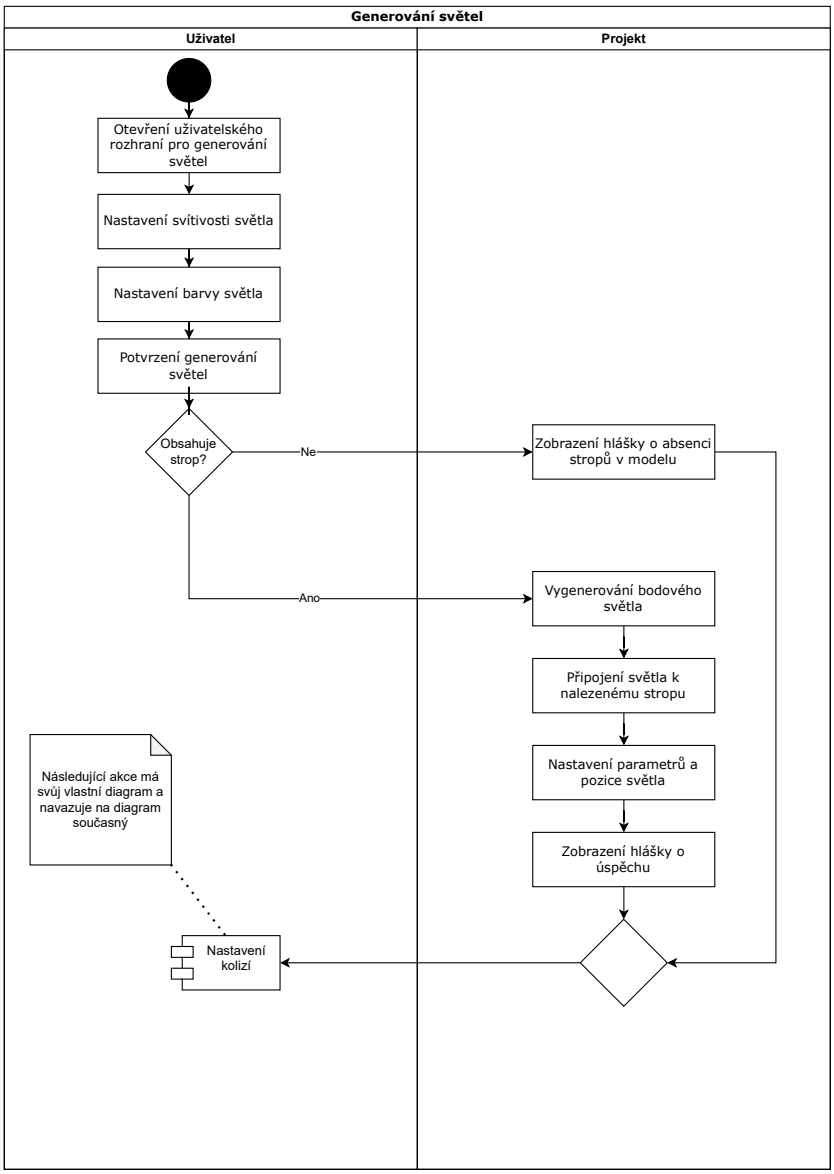
Obrázek 2.1 Diagram ukazující průběh importu modelu do Unreal Engine pomocí funkcionalit projektu



Obrázek 2.2 Diagram ukazující průběh importu informací o IFC materiálech do Unreal Engine pomocí funkcionalit projektu



Obrázek 2.3 Diagram ukazující průběh přiřazení Unreal Engine materiálů k odpovídajícím geometriím pomocí funkcionalit projektu



Obrázek 2.4 Diagram ukazující průběh generování světel pro model pomocí funkcionalit projektu

pak začne pro každý objekt nastavovat kolize. Z většiny použije jednoduché box kolize, které objekt obalí do kvádru, sloužícího k detekci kolizí. Pokud se však jedná o dveře či stěnu obsahující otvor na dveře, nastavení kolizí se liší. Aby byl umožněn průchod dveřmi, nenastavují se pro ně žádné kolize a ze stejného důvodu (k umožnění průchodu) se u zdí s otvorem nastavují komplexní tvary k detekci kolizí. Po dokončení je zobrazena hláška o výsledku. Scénář dále pokračuje posledním diagramem pro průchod scény 2.2.6.

2.2.6 Diagram Procházení ve Scéně

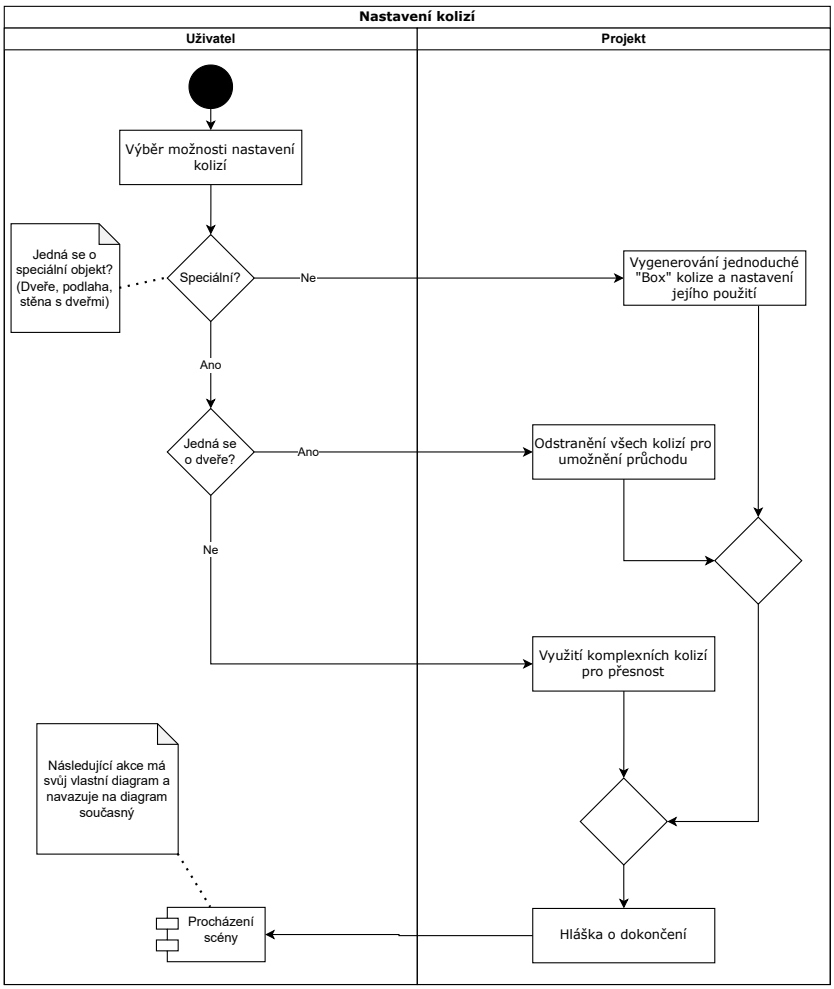
V tomto diagramu, zachyceném na obrázku 2.6, nehraje roli naše uživatelské rozhraní, proto ho uživatel zavře. Uživatel umístí startovní pozici na správné místo ve scéně (v rozmezí importované budovy a startovní pozice nesmí kolidovat s žádným objektem). Následně uživatel spustí celý projekt a tím se zrodí postava k procházení na vybraném místě. Dál už uživatel pouze prochází v osvětlené budově s přiřazenými materiály.

2.3 Log

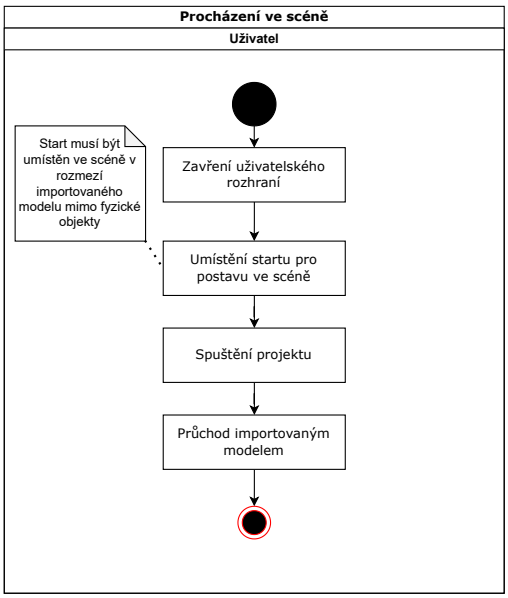
Aby uživatel byl obeznámen s výsledky operací a s probíhajícími procesy, uživatelské rozhraní bude obsahovat log. Pro větší přehlednost bude log v uživatelském rozhraní oddělený od Unreal Engine logu.

2.4 Import IFC modelů

K importu bude využit nativní zásuvný modul Datasmith, přesněji Datasmtith CAD. Datasmith CAD podporuje modely IFC společně s importem celých scén a zachováním jejich hierarchií. Proto je Datasmith CAD ideální volbou pro tento úkol. Datasmith bude použit za pomoci python API, které Unreal Engine nabízí a integrován do našeho uživatelského rozhraní bude přes blueprint uzel Execute Python Scrip. Import je navržen tak, aby nabídl uživateli možnost nastavit jeho vlastní parametry, ale i použít parametrů výchozích. K samotnému importu budou použity Unreal Engine python třídy jako Datasmtih Import Options k nastavení možností importu a Import Factory Task s Datasmith Import Factory k nastavení celkového kontextu. Importovaný model se bude automaticky ukládat do složky “Datasmith Import” k jednoduchému udržení přehledu o importovaných modelech. Problematickým souborem je testovací soubor `ZaB_only_floors_doors_doorWalls.ifc`, který je příliš malý, a proto by se mohlo do návrhu importu zařadit i automatické škálování (nastavení měřítka). V testovacím souboru však není nic, podle čeho by se dalo škálování provést a podle standardu by měly být modely v měřítku jedna ku jedné, proto bylo z návrhu škálování odstraněno.



Obrázek 2.5 Diagram ukazující průběh nastavení kolizí pro geometrie v Unreal Engine pomocí funkcionalit projektu



Obrázek 2.6 Diagram ukazující průchod vizualizací modelu v Unreal Engine pomocí funkcionalit projektu

2.5 Zobrazení importovaných modelů

Protože u některých operací je pracováno s actory, je nutné, aby se model nacházel ve scéně. Zobrazení importovaných modelů je proto navrženo tak, aby zobrazilo pouze modely nacházející se ve scéně. K tomu slouží DatasmithSceneActor, který slouží jako obálka struktury modelu ve scéně. K získání všech modelů ve scéně je použit blueprint uzel Get All Actors Of Class s nastavením třídy právě na DatasmithSceneActor.

2.6 Dodatečný import parametrů

Jak už bylo řečeno v analýze, IFC modely obsahují také další užitečné vlastnosti a parametry, které zásuvný modul Datasmith CAD neimportuje. Proto se budeme v této části návrhu věnovat jim.

2.7 Has Openings a Fills Voids

Atributy Has Openings a Fills Voids jsou přiřazené ke každému IFC elementu a obsahují data o elementech vložených do jiných elementů. Jak ukázala analýza, v testovacích souborech mají tyto parametry vyplněné například okna a dveře (Fills Voids) a stěny (Has Openings). Je tak možné rozlišit, zdali mají stěny v sobě okna, dveře, obojí, či ani jedno. Tato informace se bude velice hodit při nastavování kolizí. Import těchto atributů je tedy navržen tak, aby probíhal v rámci importu samotného modelu a informace tak byla vždy dostupná. K importu bude využita knihovna IfcOpenShell – Python a bude prozatím probíhat pouze pro jednotlivé stěny (entity typu IfcWall). IfcOpenShell projde pro nalezené stěny všechny vztahy v atributu HasOpenings a pokud nalezne okno (IfcWindow), nastaví si pro entitu stěny flag HasWindows, pokud nalezne dveře (IfcDoor), nastaví si pro entitu stěny flag HasDoors. Nastavené flagy budou po importu v Unreal Engine přířazeny k jednotlivým objektům reprezentujícím stěnu. Přiřazení proběhne za pomoci vlastnosti metadata, kterou Unreal Engine nabízí. Přesněji budou metadata přiřazena pro objekty Static Mesh, protože k těm se bude následně přiřazovat kolize.

2.8 IFC Materiály

Velká část elementů má v modelu přiřazeny IFC materiály, některé z nich však nejsou za pomoci Datasmith importovány, proto náš projekt bude navržen tak, aby materiály doimportoval. Import materiálů bude probíhat odděleně od importu modelu a bude využívat své vlastní uživatelské rozhraní. Importovat se budou informace o všech materiálech, a aby se předešlo situacím kdy jeden objekt obsahuje více materiálů, uživatel si bude muset vybrat pro každý

objekt pouze jeden. Sada dostupných materiálů bude získaná za pomoci IfcOpenShell – Python, kdy bude využita funkce by_type, umožňující filtr entit v IFC modelu podle typu. Po výběru materiálů uživatelem se vytvoří seznam těch, které jsou určeny k importu. Následně bude využito blueprint uzlů "Get All Actors Of Class" a "Get Attached Actors" k průchodu hierarchie objektů. Pomocí IFC ID, které ke každému objektu při importu přiřadil Datasmith, bude vyhledána odpovídající entita v IFC modelu. Za pomoci funkce ifcopenshell.util.element.get_material z IfcOpenShell – Python budou získány informace o materiálu a ty pak porovnány se seznamem materiálů určených k importu. V případě shody bude jméno materiálu přiřazeno za pomoci metadat k odpovídajícímu Static Mesh. Po dokončení jsou nově importované materiály přidány do tabulky pravidel, které se věnuje následující sekce 2.9.

2.9 Tabulky Pravidel

Aby uživatel mohl mít kontrolu nad přiřazovanými materiály bez nutnosti zásahu do kódu, jsou pro projekt navrženy takzvané tabulky pravidel. Tyto tabulky využívají Unreal Engine technologie String Table. String Table je využita z důvodu struktury klíč a hodnota. V klíči se budou nacházet jména IFC materiálů, sloužící k rychlému vyhledávání v tabulce a v hodnotě se budou nacházet jména Unreal Engine materiálů či cesty k nim. Aby mohlo být vyplněno pouze jméno Unreal Engine materiálu, musí se daný materiál nacházet ve složce `Materials_to_assign`, v opačném případě musí být vyplněna referenční cesta k materiálu. Referenční cestu k jakémukoliv assetu je možno získat za pomoci kliku pravým tlačítkem na asset a výběru možnosti "Copy Reference". Tabulky pravidel jsou navrženy dvě. Jedna pro přiřazení podle IFC entit a jedna pro přiřazení podle IFC materiálů. Uživatel má přístup k oběma. Tabulka podle IFC entit pak funguje na stejném principu jako tabulka podle IFC materiálů, pouze má jako klíč typy entit. Z důvodu omezené množiny typů IFC entit a všech elementů mající určený typ IfcEntity, je tabulka pravidel podle IFC entit preferovaná. Tabulka přiřazení podle entit je tedy použita jako defaultní volba, pokud není možné použít druhou tabulku z důvodu neúplných informací.

2.10 Přiřazení Materiálu

K přiřazení materiálů jsou použity tabulky pravidel zmíněné v minulé kapitole. K manipulaci s tabulkami jsou využity C++ moduly Unreal Engine, umožňující vyhledávat v tabulce podle klíče a tabulky načítat. Uživatel si vybere, jakou tabulku chce mít jako preferenční a následně se pomocí blueprint uzlu Get Assets by Path načtou všechny Static Mesh pro vybraný importovaný model. Všechny Static Mesh se nacházejí ve složce vybraného modelu, v podsložce Geometry. Načtenými Static Mesh se následně iteruje a podle vy-

brané preferenční tabulky se hledají metadata o IFC typu či IFC materiálu. Informace získané z metadat jsou vyhledány v odpovídající tabulce pravidel a v případě úspěchu je tímto identifikován Unreal Engine materiál, který se podle názvu či cesty načte a přiřadí. Pokud dojde k neúspěchu (chybí odpovídající klíč, není přiřazen IFC materiál nebo IFC entita), záleží další postup na tom, jaká tabulka byla použita jako preferenční. Pří neúspěchu v tabulce pravidel pro IFC materiály je v návrhu použita jako náhrada tabulka pravidel podle IFC entit, která by měla být vždy validní. Pokud však vznikne problém i s touto tabulkou, je použit základní výchozí materiál. Přiřazení bude probíhat pomocí blueprint uzlu Assign Material.

2.11 Generování Světel

Z analýzy standardů IFC a testovacích souborů vyplývá, že v testovacích souborech žádné informace využitelné ke generování světel nejsou přítomny. Testovací model University Building Model.ifc má však informace o stropech v objektu. Protože světla se pak zpravidla nacházejí na stropech, pro tento návrh je k umístění generovaných světel využita entita stropu (IfcCovering).

Projekt projde pro vybraný model celou strukturu, hledajíc všechny actory reprezentující strop a pro každý z nich vygeneruje bodové světlo. Bodové světlo je vždy umístěné směrem dolů od stropu a k přesnému umístění jsou využity informace o rozměrech objektu. Přibližné rozměry objektu jsou získány za pomocí takzvaných bounds, kdy Unreal Engine automaticky ohraničí všechny objekty osově zarovnaným kvádrem pro zlehčení některých výpočtů a procesů.

Po zjištění souřadnic pro umístění světla je světlo vygenerováno blueprint uzlem Spawn Actor Point Light a ke stropu přiřazeno za pomoci blueprint uzlu Attach Actor To Actor. Barva světla je navržena tak, aby mohla být nastavena uživatelem za pomocí RGB vektoru. Svítivost je nastavena za pomocí velikosti stropu, ale uživatel je schopen nastavit koeficient, kterým je velikost stropu přenásobena.

Aby nedocházelo ke generování světel na místě, kde již světla vygenerována jsou, nastaví se při spuštění generace flag Lights uložený v metadatech. Generování světel tak probíhá pouze pokud je flag nastaven na false.

2.12 Odebrání Světel

Uživatelské rozhraní bude také nabízet možnost odebrání světel. To bude probíhat podobným způsobem jako generování světel. Projekt projde všechny actory určeného IFC modelu a pokud má actor nastaven flag Lights na true, pokusí se odebrat všechna připojená bodová světla. Flag je pak opět nastaven na false.

2.13 Nastavení Kolizí

Kolize jsou podle návrhu přiřazovány přímo k Static Mesh modelu. Z důvodu optimalizace bude většina kolizí řešena pouze za pomoci jednoduchého kvádru obalujícího celý Static Mesh. Kvádr byl vybrán z důvodu charakteristiky geometrie většiny objektů a bude generován automaticky za pomocí blueprint uzlu "Add Simple Collisions" s nastavením "Box collisions". Jsou zde ale také objekty, jejichž kolize musí být řešena zvlášť. Z analýzy testovacích souborů se jedná o elementy jako jsou stěny (IfcWall), dveře (IfcDoor) a podlaha (IfcSlab). Pro podlahu a stěny je nutné mít kolize velmi přesné, jinak znemožní průchod. U podlahy by znemožnily jednoduché kolize přechod mezi jednotlivými patry po schodech. Pro stěny by pak jednoduché kolize znemožnily průchod dveřmi. Aby v rámci optimalizace nemusela být přiřazena komplexní složitá kolize ke všem stěnám, je využito flagů, popsaných v podsekci Has Openings a Fills Voids 2.7 k identifikaci různých typů stěn. Stěny se dělí na stěny s dveřmi a bez oken, stěny s dveřmi a s okny a stěny ostatní. Stěnám s dveřmi a bez oken je nutné nastavit komplexní kolize, aby byl umožněn průchod dveřmi. Pokud mají stěny dveře a okna zároveň, může se jednat o indikaci, že stěna je venkovní a lepší možností bude nastavení jednoduchých kolizí (pro zamezení průchodu mimo budovu). Pro větší univerzálnost je však nastavena komplexní kolize i pro tento případ. Komplexní kolize se bude nastavovat za pomocí python scriptu. V tom musí být upravený flag nacházející se v body_setup třídě se jménem collision_trace_flag. Poslední možností speciálních kolizí jsou dveře. Pro ty budou kolize odebrány za pomoci blueprint uzlu Remove Collisions.

2.14 Průchod scény

Průchod scény využívá již vytvořenou postavu dostupnou v Unreal Engine šablonách, ta je pouze upravena tak, aby splňovala naše požadavky. Pro kompletní funkčnost procházení bude stačit pouze umístit objekt PlayerStart na správné místo ve scéně a zapnout projekt. PlayerStart objekt se ve scéně již bude nacházet a bude nutné ho pouze přemístit na správné místo. Po rozmyšlení možných automatizací umístění Player Start na správné místo bylo rozhodnuto ponechání umístění v návrhu na uživateli. Automatizace by byla příliš náročná, protože modely neobsahují jednotné informace, podle kterých by mohlo být umístění provedeno a pro uživatele se jedná o poměrně jednoduchý úkon. Mimo umístění startovní pozice je nutné mít nastavené všechny kolize pro úplnou funkcionalitu procházení.

2.15 Neúplný model

Výše popisované funkčnosti mají zamýšlený efekt pouze v případě, kdy jsou v modelu k dispozici potřebné parametry. Může se však stát, že model potřebné

parametry nemá, tedy z našeho pohledu se jedná o neúplný model. Návrh projektu pro neúplný model vynechává či omezuje většinu funkcionalit. Neúplný model je ten, který nemá plně definované své objekty a neobsahuje téměř žádné vlastnosti, proto není možné využít žádné jiné informace kromě těch geometrických. Příkladem je testovací soubor `ZaB_only_floors_doors_doorWalls.ifc`. Pro tento soubor návrh pracuje s jinými pravidly. Protože nebude možné přiřadit světla ke stropům (model žádné stropy nemá), přiřadí funkcionalita emisní materiál pro zajištění osvětlení. Nastavení kolizí pak nastaví jednoduché kolize ke všemu, protože model nemá žádné stěny nebo dveře identifikované. Protože model neobsahuje žádné IFC materiály, nepůjdou žádné doimportovat a přiřazení bude tak probíhat pouze podle IFC entit. Import neúplného modelu pak bude probíhat stejně jako u úplného modelu.

2.16 Uživatelské rozhraní

Tato sekce se věnuje návrhu uživatelského rozhraní. K tvorbě uživatelského rozhraní bude využita blueprint třída dědící z Editor Utility Widget. Ta je specializovaná na tvorbu uživatelského rozhraní, ale je určena pouze k použití v editoru (mimo běh projektu). Protože uživatelské rozhraní našeho projektu je určeno pouze pro editor, jedná se o vyhovující možnost parent blueprint třídy pro naše použití. Při vytváření uživatelského rozhraní máme zpřístupněné grafické prvky jako tlačítko, text, textové pole a tak dále, ke kterým můžeme přiřazovat funkcionalitu za pomocí blueprint uzlů.

2.17 Uživatelské rozhraní

Tato sekce se věnuje návrhu uživatelského rozhraní. K tvorbě uživatelského rozhraní bude využita blueprint třída dědící z Editor Utility Widget. Ta je specializovaná na tvorbu uživatelského rozhraní, ale je určena pouze k použití v editoru (mimo běh projektu). Protože uživatelské rozhraní našeho projektu je určeno pouze pro editor, jedná se o vyhovující možnost parent blueprint třídy pro naše použití. Při vytváření uživatelského rozhraní máme zpřístupněné grafické prvky jako tlačítko, text, textové pole a tak dále, ke kterým můžeme přiřazovat funkcionalitu za pomocí blueprint uzlů.

2.17.1 Log

Všechna následující uživatelská rozhraní jsou navržena tak, aby měla zakomponovaný log. Do logu se budou vypisovat výsledky probíhajících operací. Log se bude skládat z:

- Textové pole pro zobrazování zpráv

- Posuvník pro procházení zpráv
- Tlačítko pro vymazání logu

Aby byl Log stále přítomný a nemusely se přenášet zprávy mezi jednotlivými částmi uživatelského rozhraní, bude zbytek uživatelského rozhraní sestrojen za pomoci takzvaného Widget Switcher. Widget Switcher umožňuje existenci více uživatelských rozhraní na jednom místě a pouze mezi nimi přepínat.

2.17.2 Přehled Importovaných modelů

Uživatelské rozhraní pro import nového modelu můžeme vidět na obrázku 2.7. Tato část umožní uživateli import nového IFC modelu za pomoci zadané cesty a obsahuje:

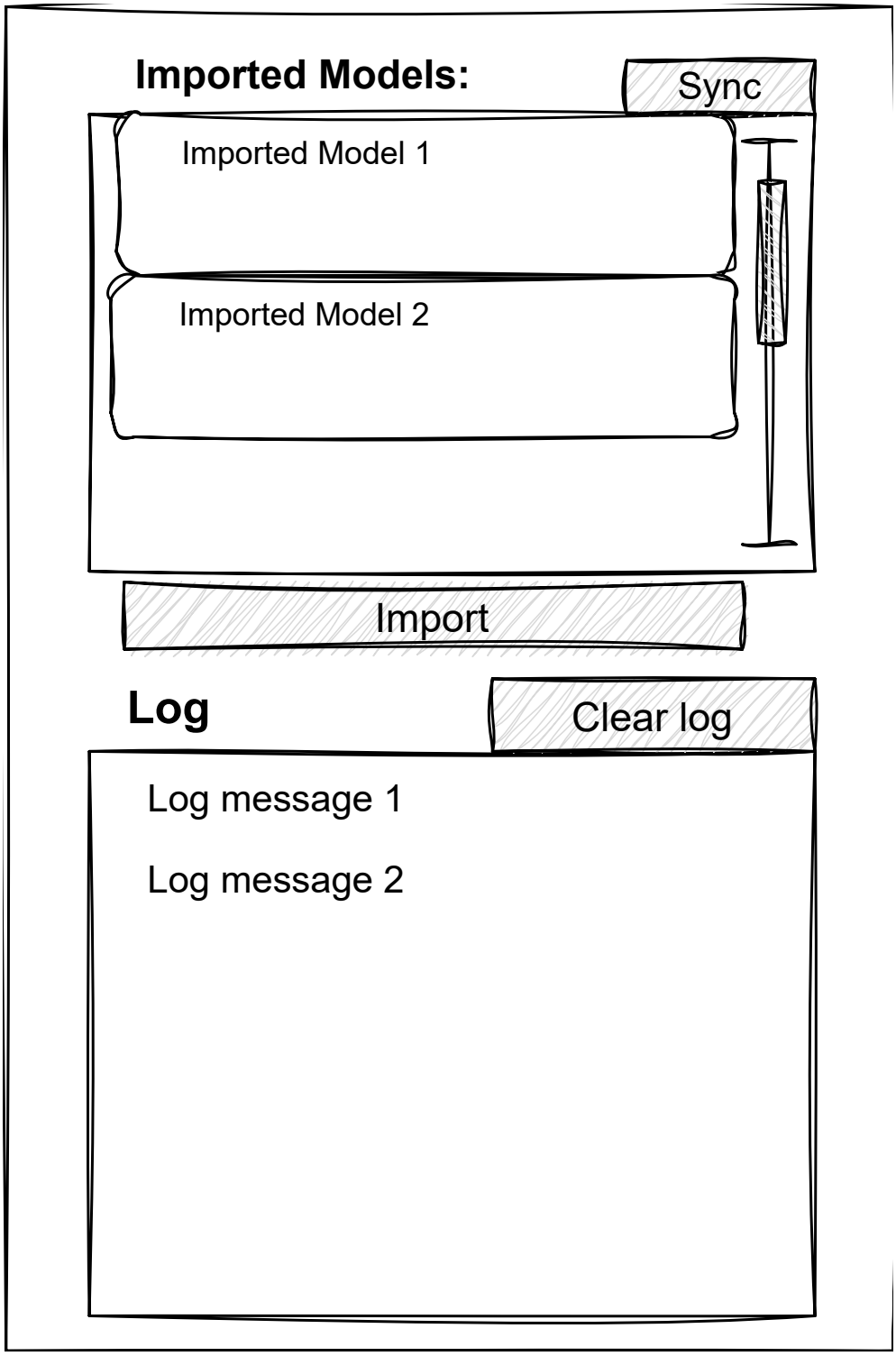
- Kontejner pro reprezentaci modelů
- Reprezentace modelů
- Posuvník k procházení modelů
- Tlačítko pro aktualizaci zobrazených modelů
- Tlačítko pro dodatečný import
- Log zmíněný v podsekci 2.17.1

Po zadání cesty k IFC souboru a zvolení typu cesty má uživatel na výběr, zdali chce zadat parametry importu sám (zaškrtne show options) nebo je nechá vyplnit projektem (odškrtne show options). Po kliknutí na tlačítko Import se podle hodnoty zaškrtávátka zobrazí možnosti či ne a začne probíhat import. Po kliknutí na tlačítko zpět se uživatel navrátí do uživatelského rozhraní přehledu importovaných modelů.

2.17.3 Import IFC modelu

Uživatelské rozhraní pro import nového modelu můžeme vidět na obrázku 2.8. Tato část umožní uživateli import nového IFC modelu za pomoci zadané cesty a obsahuje:

- Textové pole pro cestu k IFC souboru
- Zaškrtávátko pro zobrazení možností při importu
- Rozevírací seznam pro výběr typu zadané cesty
- Tlačítko importu pro spuštění importu
- Tlačítko zpět pro návrat do přehledu modelů



Obrázek 2.7 Návrh uživatelského rozhraní pro úvodní menu s přehledem modelů

- Log zmíněný v podsekci

Po zadání cesty k IFC souboru a zvolení typu cesty má uživatel na výběr, zdali chce zadat parametry importu sám (zaškrtne “show options”) nebo je nechá vyplnit projektem (odškrtne “show options”). Po kliknutí na tlačítko „Import“ se podle hodnoty zaškrtávátka zobrazí možnosti či ne a začne probíhat import. Po kliknutí na tlačítko zpět se uživatel navrátí do uživatelského rozhraní přehledu importovaných modelů.

2.17.4 Možnosti vybraného modelu

Pokud uživatel otevře z přehledu modelů některý ze zobrazených modelů, otevře se část uživatelského rozhraní nabízející operace s daným modelem. Tuto část lze vidět na obrázku 2.9 a obsahuje:

- Název vybraného modelu
- Tlačítko pro doimportování informací o materiálech
- Tlačítko pro přiřazení materiálů
- Rozevírací seznam pro preferenci pravidel pro přiřazení materiálů
- Tlačítko pro nastavení kolizí
- Tlačítko pro generování světel
- Tlačítko zpět pro návrat do přehledu modelů
- Log zmíněný v podsekci 2.17.1

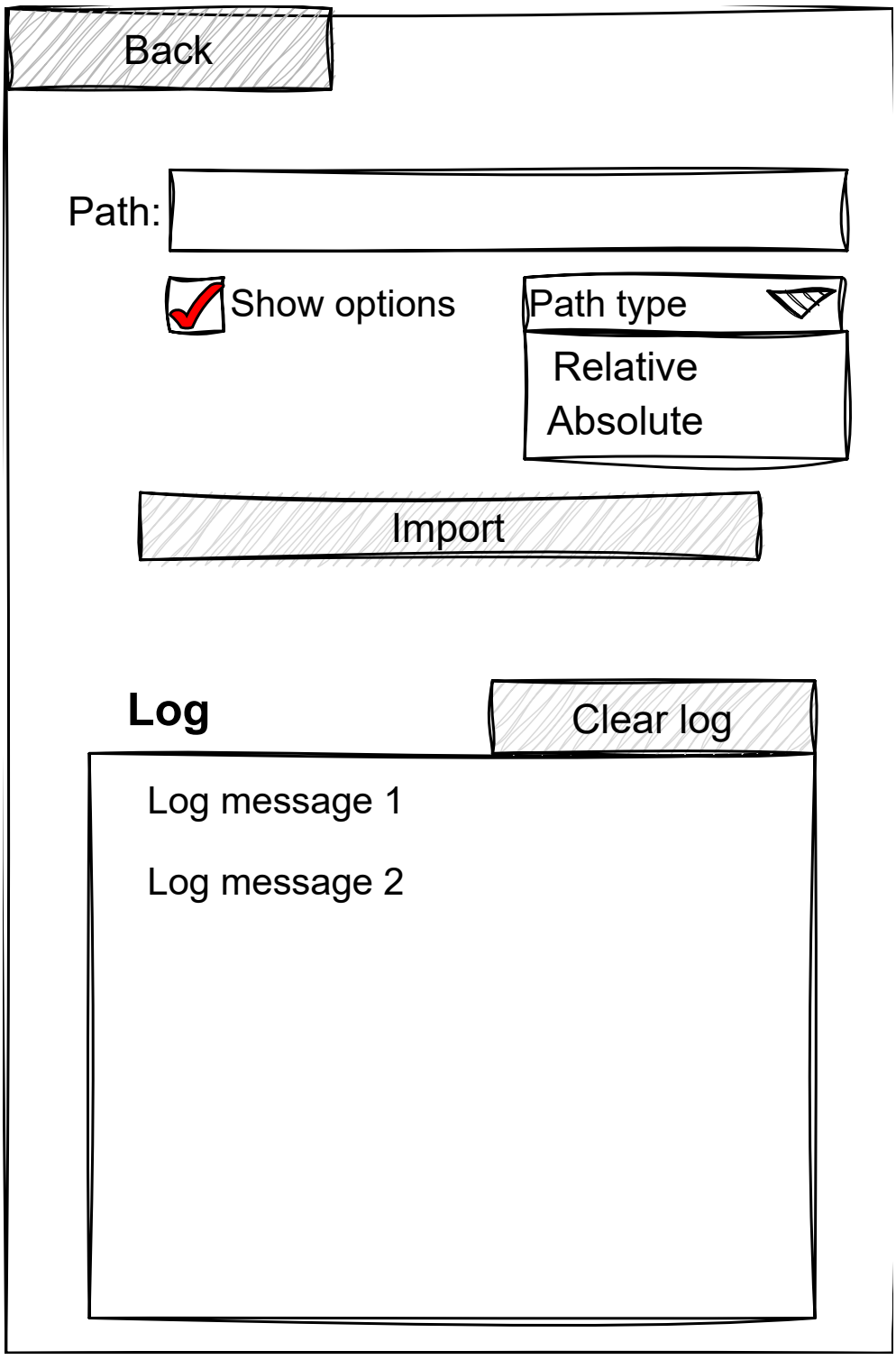
Pokud uživatel zvolí import materiálů, bude přenesen do další části uživatelského rozhraní. Tato další část se zabývá importem materiálů a je popsaná v podsekci 2.17.6. Jinou částí uživatelského rozhraní, kterou může uživatel zvolit a do které může být následně přesměrován, je část pro generování světla, která pokrývá generování a odstraňování světel. Tato část je popsaná v podsekci 2.17.5.

Zbývající tlačítka (přiřazení materiálu a nastavení kolizí) nemají žádné další uživatelské rozhraní a přímo spustí danou operaci. Pro návrat do uživatelského rozhraní přehledu importovaných modelů musí uživatel zvolit tlačítko zpět.

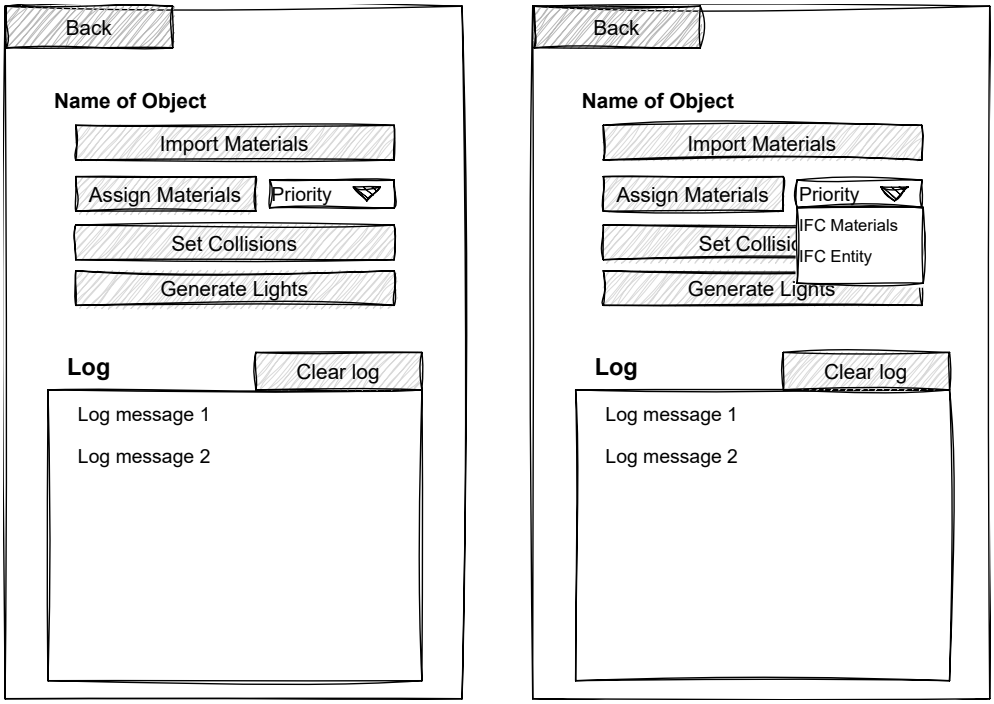
2.17.5 Generování světel

Uživatelské rozhraní pro generování a odstraňování světel můžeme vidět na obrázku 2.10. Obsahuje:

- Posuvník pro výběr intenzity jasu



Obrázek 2.8 Návrh uživatelského rozhraní pro import modelů



Obrázek 2.9 Návrh uživatelského rozhraní ukazující možné operace s importovaným modelem

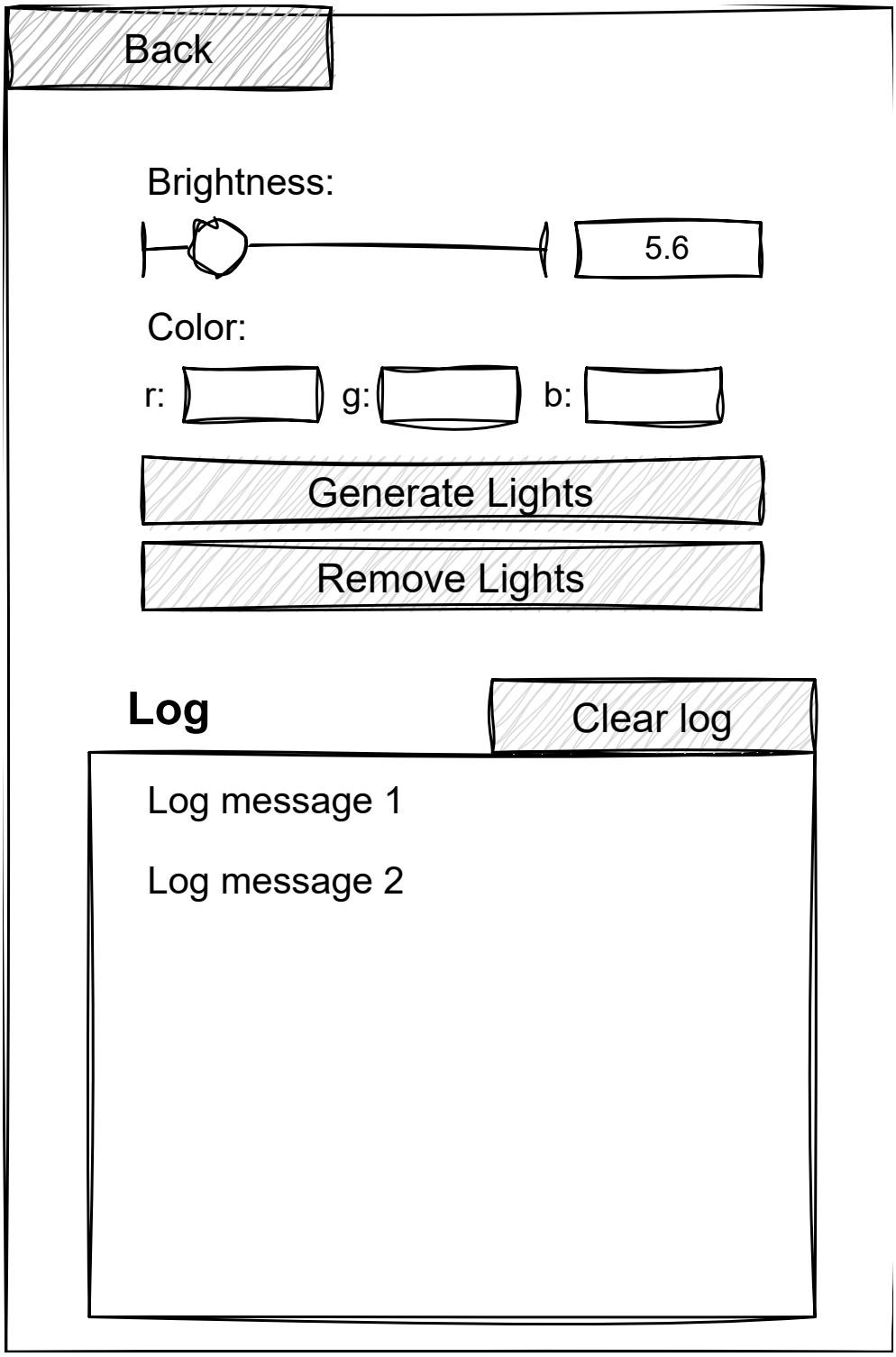
- Políčka pro zadání barvy ve formátu RGB
- Tlačítko pro generování světel
- Tlačítko pro odstranění vygenerovaných světel
- Tlačítko zpět pro návrat do možností modelu
- Log zmíněný v podsekci 2.17.1

Uživatel může v tomto uživatelském rozhraní vybrat parametry (jas a barvu) pro generování světel. Ta se vygenerují či odstraní po stisku jednoho z tlačítek. Podrobnější funkcionalita je popsaná v sekci 2.11. Po kliknutí na tlačítko zpět se uživatel navrátí do uživatelského rozhraní možnosti vybraného modelu.

2.17.6 Import Materiálů

Poté, co uživatel zvolí možnost importu materiálů, dostaneme se do části uživatelského rozhraní, kterou můžeme vidět na obrázku 2.11. Tato část obsahuje:

- Textové pole pro cestu k IFC souboru



Obrázek 2.10 Návrh uživatelského rozhraní pro generování světla a jeho případné odstranění

- Tlačítko pro zobrazení dostupných materiálů (získaných ze souboru IFC)
- Tlačítko pro import vybraných materiálů
- Kontejner pro reprezentace materiálů
- Posuvník k procházení reprezentací materiálů
- Tlačítko zpět pro návrat do možností modelu
- Log zmíněný v podsekci 2.17.1

Reprezentace materiálu bude více rozebrána v podsekci 2.17.8. Cesta k IFC souboru bude navržena tak, aby byla předvyplněná automaticky. Pokud se však soubor přemístil, či cesta nebyla z nějakého důvodu předvyplněná, uživatel bude muset zadat korektní cestu k IFC souboru. Po zadání korektní cesty bude možné zmáčknout tlačítko pro zobrazení dostupných materiálů, které v kontejneru zobrazí všechny materiály dostupné v modelu. Uživatel si z daných materiálů může vybrat materiály, jejichž informace se následně importují po stisknutí tlačítka pro import. Výběr materiálu je podrobněji popsán v sekci 2.8. Po kliknutí na tlačítko zpět se uživatel navrátí do uživatelského rozhraní možnosti vybraného modelu.

2.17.7 Reprezentace importovaného modelu

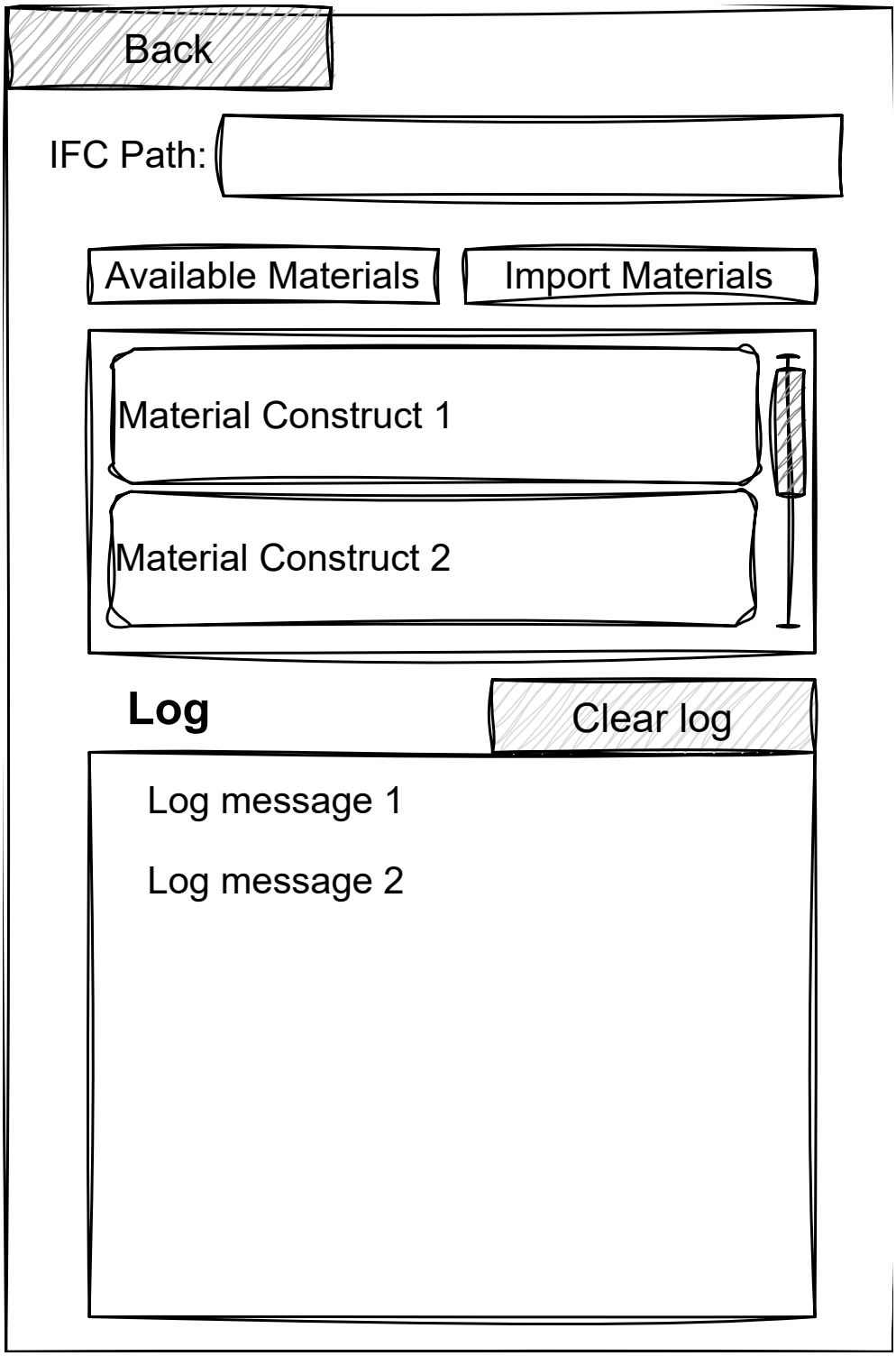
Reprezentace importovaného modelu je velmi jednoduchá a je možné ji vidět na obrázku 2.12. Bude se jednat o tlačítko, které bude mít na sobě napsané pouze jméno pro daný model.

2.17.8 Reprezentace konstruktu materiálu

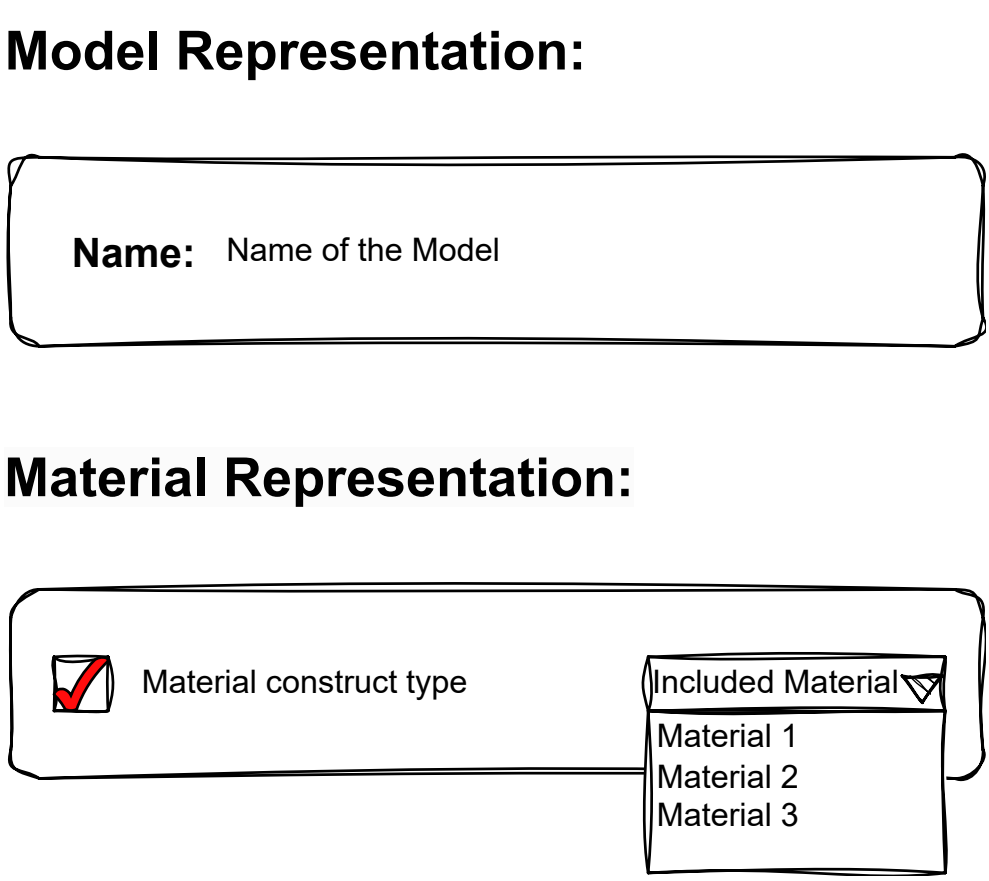
Tuto reprezentaci je možné vidět na obrázku 2.12 a obsahuje:

- Zaškrtávátko pro výběr k importu
- Text pro typ konstruktu
- Rozevírací seznam pro všechny obsažené materiály v konstruktu

Tento konstrukt reprezentuje jak samotné IFC materiály, tak i jejich množiny (například IfcMaterialLayerSet, IfcMaterialConstituentSet a IfcMaterialProfileSet). Pro konstrukty, které obsahují více než jeden materiál, bude uživatel donucen si za pomoci rozevíracího seznamu vybrat maximálně jeden materiál k importu. Zaškrtávátko pak bude sloužit k zahrnutí konstruktu do importu či jeho vynechání. Jako příklad si můžeme uvést konstrukt IfcMaterialConstituentSet, který bude obsahovat materiály sklo a dřevo. Z rozevíracího seznamu si uživatel bude muset vybrat buď pouze sklo, či pouze dřevo. Pokud si uživatel



Obrázek 2.11 Návrh uživatelského rozhraní pro dodatečný import informací o IFC materiálech



Obrázek 2.12 Návrh uživatelského rozhraní ukazující grafické elemnty reprezentující importovaný model a IFC materiál konstrukt

vybere sklo a zaškrtne zahrnutí konstruktu v importu, ke všem elementům obsahujícím daný IfcMaterialConstituentSet se přiřadí pouze vybrané sklo. Bez zaškrtnutí se k elementům nepřiřadí nic.

Kapitola 3
Realizace

3.1 Příručka pro uživatele

V této sekci se budeme věnovat popisu implementovaných funkcionalit a návodu, jak projekt zprovoznit a používat.

3.1.1 Předpoklady pro úspěšné spuštění projektu

Praktická část je vytvořena jako projekt pro Unreal Enine 5. Doporučená verze Unreal Engine pro spuštění je verze 5.1.1, která je dostupná zdarma ke stažení na stránkách Unreal Engine [15]. Doporučený operační systém je Windows. Po úspěšném nainstalování editoru Unreal Engine se projekt spustí otevřením souboru Unreal Engine Project File, k dosažení bezproblémové funkcionality je však nutné splnit určité předpoklady, popsané v následujících podsekcích.

3.1.1.1 Knihovny třetích stran

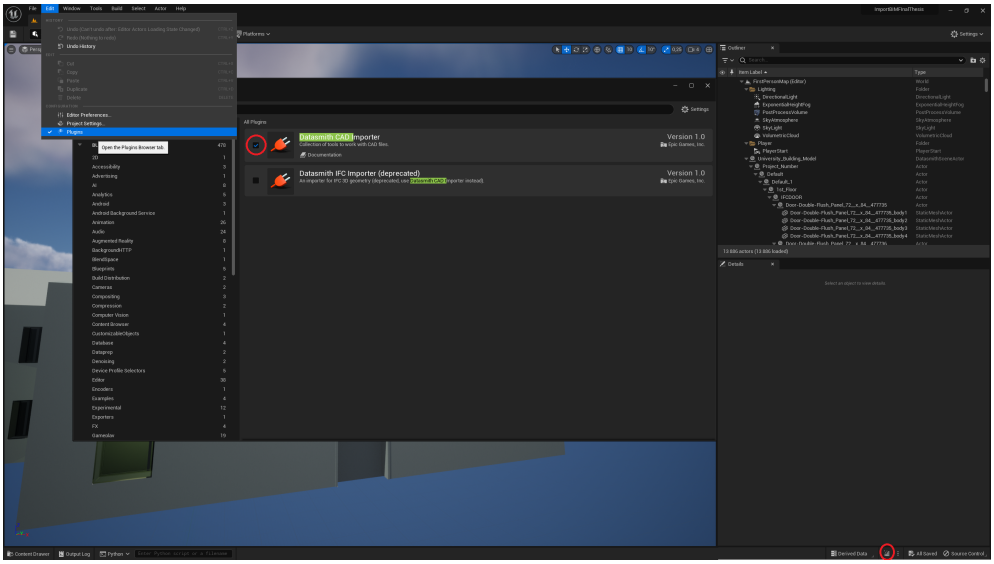
Projekt ke své funkcionalitě využívá knihovnu IfcOpenShell – Python. Tu je nutné mít nainstalovanou přímo v Python instalaci, kterou využívá Unreal Engine. Unreal Engine instalovaný na Windows využívá Python nacházející se zde: `\UE_5.1\Engine\Binaries\ThirdParty\Python3\Win64`.

Dvě nejjednodušší cesty, jak zahrnout potřebné knihovny, je využití k instalaci pip nebo knihovnu přímo přetáhnout do `Lib\site-packages`.

Pro instalaci za pomoci pip je nutné získat přesnou cestu k Unreal Engine python a spustit pro tento python pip instalaci skrze příkazový řádek. Příkaz pro instalaci může vypadat následovně:

```
\UE_5.1\Engine\Binaries\ThirdParty\Python3\Win64\
↪ python.exe -m pip install ifcopenshell
```

Pro instalaci za pomoci přetažení je nutné stažení IfcOpenShell python na oficiální stránce [19]. Důležité je stáhnout správnou verzi (Unreal Engine



Obrázek 3.1 Obrázek ukazující uživatelské rozhraní Unreal Engine 5 a označující tlačítka pro kompilaci projektu a přidání zásuvného modulu

používá Python 3.9.7). Po stažení stačí knihovnu nakopírovat do `\UE_5.1\Engine\Binaries\ThirdParty\Python3\Win64\lib\site-package`.

Pro bezproblémovou funkčnost je doporučeno nainstalovat i balíček nástrojů lark–parser, a to pomocí pip (pip install lark–parser), či stažením na github lark–parser [23]

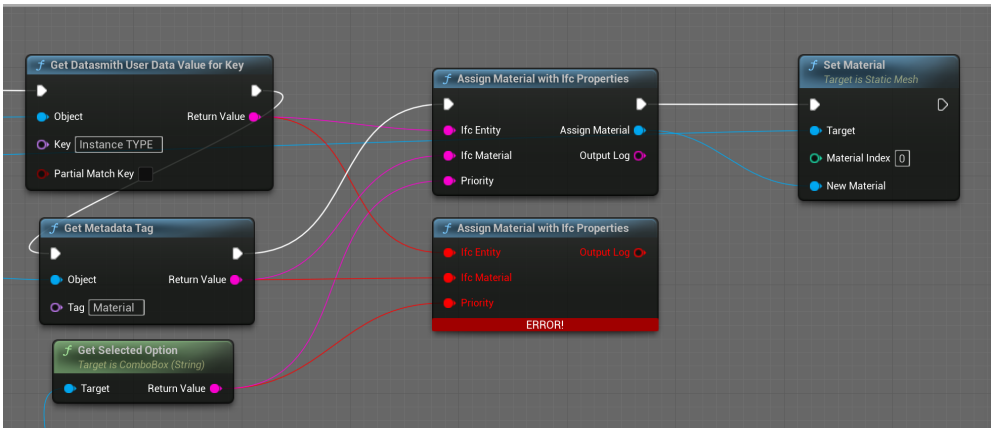
3.1.1.2 Zásuvné moduly

Zásuvné moduly, které projekt využívá, jsou již v základní instalaci Unreal Engine 5, je však nutné je aktivovat. K aktivaci slouží takzvaný Plugin browser. Ten se dá vyvolat za pomoci tlačítka Edit a následně tlačítka Plugins v Unreal Engine menu.

Po otevření Plugin browser je nutné vyhledat zásuvné moduly Datasmith CAD Importer a Python Editor Script Plugin a zkontrolovat, že jsou aktivované (zaškrtávátko po jejich levé straně je zaškrtnuto). Pokud aktivované nejsou, je pro využití všech funkcionalit nutné je aktivovat.

3.1.1.3 Sestavení projektu

Po otevření nového projektu bude uživatel nucen k jeho kompilaci. K tomu slouží tlačítko v pravém dolním rohu 3.1, které znovu načte a zkompiluje všechny C++ zdrojové kódy.



Obrázek 3.2 Obrázek ukazující rozpojený blueprint uzel Assign Material with Ifc Properties a jeho nové znovunapojení

3.1.1.4 Napojení blueprint uzlů

Protože po novém načtení C++ kódů a jejich kompilaci je možné, že Unreal Engine bude mít problémy s napojením blueprint uzlů, je nutné to zkontrolovat. Ve složce s obsahem, takzvaném Content Drawer, je nutné navigovat do složky BimImportFunctionality. V této složce je vidět uživatelské rozhraní se jménem MaterialAssignment, které je možné dvojím kliknutím otevřít. Po otevření se musí v pravém horním rohu zkontrolovat, zdali se nacházíme v takzvaném Graph editoru a následně nalézt sekci blueprint uzlů popsanou jako Assign Material.

Zde je možné najít uzel se jménem Assign Material with Ifc Properties, který je vytvořen jako vlastní C++ blueprint funkce. Po kompilaci je občas nutné uzel znovu přidat a napojit podle obrázku 3.2.

Z důvodu občasné nutnosti přepojování uzlů vytvořených s pomocí C++ byl také nakonec používán primárně Python.

3.1.2 Spuštění

Po splnění všech předpokladů k úspěšnému běhu je možné vyzkoušet funkcionalitu projektu. Ta se ovládá skrze uživatelské rozhraní, nacházející se ve složce BimImportFunctionality. Uživatelské rozhraní je možné spustit klikem pravého tlačítka myši na jeho ikonku a výběrem možnosti Run Editor Utility Widget. Poté se otevře nové okno obsahující uživatelské rozhraní, okno je možné zvětšit či zmenšit, aby byly prvky uživatelského rozhraní dobře viditelné.

3.1.3 Průchod scénou

Jak bylo řečeno v návrhu, pro průchod scénou je nutné umístit aktéra Player Start na požadované místo a následně scénu spustit, což se provádí tlačítkem

Hrát v horní části Unreal Engine menu. Pokud není aktér Player Start ve scéně, je možné ho přidat klikem pravého tlačítka myši a výběru možností `PlaceActor>PlayerStart`.

3.1.4 Možné využití funkcí projektu

V této podsekci si popíšeme možný průběh použití funkcionalit projektu. Jako předpoklad je projekt bez jakéhokoliv importovaného IFC modelu a bez jakéhokoliv IFC modelu ve scéně.

Uživatel si otevře uživatelské rozhraní a za pomocí tlačítka Import importuje nový IFC model. Pro úspěšný import je nutné zadat korektní absolutní cestu a dále uživatel vybere, zdali chce zadat parametry importu sám, či ne.

Po úspěšném importu se model sám vloží do scény, a proto ho uživatel může rovnou začít používat (pokud se model ve scéně nenachází, je nutné vložit Datasmith Scene, reprezentující importovaný model, do scény). Pokud je model ve scéně, uživatel se může navrátit zpět do přehledu modelů tlačítkem Zpět a zvolit nově importovaný model z přehledu.

Po kliknutí na zvolený model v přehledu modelů jsou uživateli nabídnuty možnosti importovat dodatečné vlastnosti (IfcMaterial, HasOpenings), generovat světla, nastavit kolize, či přiřadit materiály. Protože nastavení kolizí a přiřazení materiálů funguje nejlépe po importu dodatečných vlastností, uživatel si nejprve zvolí možnost importovat dodatečné vlastností.

V této části je opět nutné zadat cestu k IFC souboru, ze kterého se budou dodatečné informace importovat. Uživatel tedy zadá tuto cestu a zvolí tlačítko Dostupné materiály. Ze zobrazených materiálů zvolí ty, které chce importovat, za pomocí zaškrtávátka a u zaškrtnutých materiálů zvolí z rozevíracího seznamu jeden materiál k importu.

Aby se nastavily správně i kolize, zaškrtne uživatel i volbu pro zahrnutí vlastností sloužících pro nastavení kolizí (vlastnost HasOpenings) a importuje vše za pomocí tlačítka Import Properties. V logu se následně zobrazí všechny importované materiály, jejichž pravidla musí uživatel nastavit do tabulky pravidel.

Proto si uživatel otevře tabulku String Table s názvem AccessibleMaterialsIfcMaterial (nacházející se v `BimImportFunctionality>MaterialsToAssign`) a zadá importované materiály jako klíče, ke kterým přidá cestu k Unreal Engine materiálům jako hodnotu. Pro jednodušší zadání si může uživatel exportovat tabulku jako soubor csv a zkopírovat importované materiály do csv souboru. K těm pak stačí přidat cestu k Unreal Engine materiálům (případně pouze jméno, pokud se materiály nachází v `BimImportFunctionality>MaterialsToAssign`) a csv soubor znovu importovat do String Table (import csv souboru přepíše celou tabulku).

Po zadání pravidel si může uživatel opět otevřít uživatelské rozhraní a vrátit se tlačítkem Zpět do možností pro zvolený model. Zde uživatel spustí automatické přiřazení materiálů s prioritou podle IfcMaterial, jejichž pravidla

právě nastavil. Uživatel dále spustí automatické nastavení kolizí a následně otevře část pro generování světel.

V části pro generování světel si uživatel může nastavit barvu a jas světla. Uživatel si nastavená světla následně pomocí tlačítka vygeneruje. Nyní je model importovaný, vizualizovaný a připravený k průchodu.

Uživatel přidá do scény aktéra PlayerStart, umístí ho dovnitř vizualizované a importované budovy (důležité je, aby byl umístěn na místo, kde s ničím nekoliduje). Poté spustí projekt a může se procházet vizualizovanou importovanou budovou.

3.2 Implementované funkcionality

Všechny funkcionality, kromě procházení scény, jsou dostupné v uživatelském rozhraní a jsou zastřešeny blueprint třídami dědícími od Editor Utility Widget. Editor Utility Widget třídy jsou v projektu tři. První a hlavní je třída MaterialAssignment (nacházející se v BimImportFunctionality), ta zastřešuje většinu funkcionalit a také slouží jako uživatelské rozhraní. Menší část funkcionalit je v dalších dvou třídách DatasmithSceneUI a IfcMaterialConstruct (nacházející se v `BimImportFunctionality>SubWidgets`). Proto v případě úpravy či rozšíření funkcionalit je nutná manipulace s těmito třídami, v opačném případě se manipulace nedoporučuje.

Z důvodů občasných problémů s blueprint uzly tvořených našim C++ zdrojovým kódem byl preferovaný jazyk Python v kombinaci s blueprint.

3.2.1 Funkcionalita importu

Tato funkcionalita, jak bylo zmíněno v návrhu 2.4, využívá Unreal Engine Python API pro Datasmith v kombinaci se samotným zásuvným modulem Datasmith CAD Importer. Import je dostupný ihned po otevření uživatelského rozhraní po kliknutí na tlačítko import.

Od návrhu se změnila možnost zvolit typ cesty k modelu. Byla vypuštěna možnost zvolit relativní cestu. Přesto, že by bylo jednoduché funkcionalitu implementovat, není pravděpodobné, že by byla uživateli příliš využívaná (import modelů umístěných relativně k poloze blueprint třídy je nepravděpodobný).

Uživatel musí v rozhraní zadat cestu, která je podrobená kontrole o její správnosti za pomocí Python skriptu. Validace cesty probíhá za pomoci Python knihovny OS, zkratka pro operating system interface. Protože import by měl sloužit pouze pro IFC soubory, kontroluje se i zdali cesta obsahuje koncovku ifc.

Samotný import je implementován také Python skriptem. Zde je nejdůležitější Python třída ImportTask, ve které se nastaví například cesta k ifc modelu, kam se model uloží, jak se import bude chovat a s jakými nastaveními bude importován. Dalším důležitým prvkem je třída ImportFactory, která říká, co

bude k importu použito. Vše je vidět ve zdrojovém kódu níže. V této ukázce jsou uvedeny pouze nejdůležitější prvky 3.1.

Výpis kódu 3.1 Nejdůležitější prvky python skriptu pro import modelu

```
# Create an import task instance
import_task = unreal.AssetImportTask()

# Set import task parameters
import_task.filename = normalized_path
import_task.destination_path = "/Game/
    ↪ BimImportFunctionality/DatasmithIfcImports/"

# Create an instance of the Datasmith import factory
import_factory = unreal.DatasmithImportFactory()

# Assign the factory to the task
import_task.factory = import_factory

# Turn off/on Options Dialog
import_task.automated = not options

# Execute the import
unreal.AssetToolsHelpers.get_asset_tools().
    ↪ import_asset_tasks([import_task])
```

Jak je z kódu vidět, byla zahrnuta i funkcionalita z návrhu, která umožní uživateli pro rychlejší import vynechat dialog s nastavením importu.

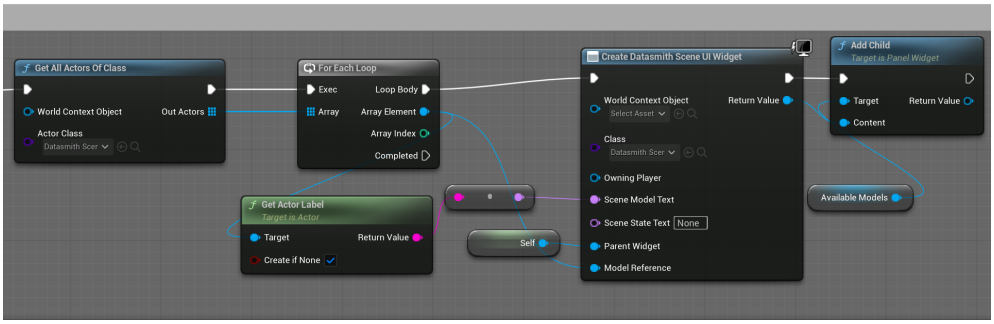
3.2.2 Funkcionalita výběru modelu

Mimo možnost importu nabízí uživatelské rozhraní hned v úvodní obrazovce přehled modelů. Tento přehled se vytváří za pomoci widget třídy DatasmithSceneUI, která je v hlavním uživatelském rozhraní vytvářena jako potomek pro každý model ve scéně. Množina modelů se získává pomocí blueprint uzlu Get All Actors Of Class, kdy se hledají všechny aktéři třídy Datasmith Scene Actor. Vyhledávání modelů a vytváření widget třídy DatasmithSceneUI je zobrazeno na obrázku 3.3

Uživatel si může ze zobrazených modelů jakýkoliv vybrat k následné úpravě. Po kliknutí na vybraný model je uživatel přenesen do další části menu za pomoci widget komponenty Widget Switcher. Přenos je inicializován z widget třídy DatasmithSceneUI.

3.2.3 Funkcionalita importu dodatečných vlastností

V této části jsou tři změny od návrhu. První je přidání uživatelského rozhraní pro import dodatečných vlastností k importu materiálů. Podle návrhu



Obrázek 3.3 Obrázek zobrazující blueprint uzly sloužící k vyhledání všech importovaných modelů ve scéně a k jejich následnému zobrazení v uživatelském rozhraní

měly být dodatečné vlastnosti, přesněji se jedná o vlastnost HasOpenings, importovány společně s importem modelu. Tato změna byla učiněna, protože je velice těžké vstupovat do importu za pomocí Datasmtih a API pro vstup do importu neumožňuje dostatečnou volnost pro provedení změn. Import dalších vlastností byl tedy přidán do uživatelského rozhraní pro import materiálů a celá část byla přejmenována na import dodatečných vlastností.

Z podobného důvodu jako první změna byla provedena i změna druhá. Tato změna se týká vynechání předvyplnění cesty k souboru, aby při dodatečném importu nemusel uživatel zadávat cestu znovu. Cesta v importované Datasmith Scene je, ale dokumentace se nezmiňuje, jak se k cestě dostat, a proto implementace této funkcionality skončila nezdarem.

Poslední změnou oproti návrhu je, že se po importu informací o materiálech názvy materiálů nepřidají automaticky do String Table tabulky. Pro operaci přidání prvku do tabulky String Table byla prohledána dokumentace C++ API i dokumentace Python API, ale bez úspěchu. Proto byla funkcionalita vynechána a nahrazena zobrazením importovaných materiálů v Logu. Z toho si je uživatel může zkopírovat a využít uživatelského rozhraní String Table pro export ve formátu csv a import ve formátu csv k jednoduchému vložení importovaných materiálů do tabulky.

Mimo uvedené změny se realizace řídí návrhem. Uživatel se do části uživatelského rozhraní pro import dodatečných vlastností dostane pomocí příslušného tlačítka. Změna uživatelského rozhraní je implementována za pomoci prvku Widget Switcher. Po vkročení do této části menu je uživatel nucen zadat cestu k IFC souboru, která je stejným způsobem jako u importu podrobena validaci. Pro import materiálů je nejprve nutné zobrazit si materiály dostupné, pro import HasOpenings je potřeba pouze zaškrtnout možnost zahrnutí kolizí.

Při zobrazení materiálů i samotném importu je používán Python skript s knihovnou IfcOpenShell – python. Při zobrazení probíhá práce pouze s IFC souborem. Ten je nahrán a pomocí IFC funkce by_type jsou vytříděny všechny obsažené materiály a jejich konstrukty. Jedná se tedy o IfcMaterial, IfcMaterialLayerSet, IfcMaterialConstituentSet a IfcMaterialProfileSet, přičemž v

testovacích souborech jsou přítomny pouze první tři.

Protože jsou přidány i materiály, které nejsou přiřazeny přímo k nějakému elementu, jsou materiály zredukovány pouze na ty, které obsahují nějaké elementy v atributu AssociatedTo. Množiny materiálů (především IfcMaterialConstituentSet) trpí problémem, že jsou brány jako rozdílné množiny, přestože jsou přiřazeny ke stejnému typu elementu a obsahují stejné materiály. Z tohoto důvodu by byl uživatel nucen vybírat ze stovky materiálů, což je nepřijatelné, a proto jsou množiny, které obsahují stejné materiály, sloučeny. K identifikaci těchto množin materiálů slouží řetězec všech obsažených materiálů, oddělených středníkem.

Takto redukované materiály jsou zobrazeny v uživatelském rozhraní za pomocí widget třídy IfcMaterialConstruct podobným způsobem, jako jsou zobrazeny dostupné modely widget třídou DatasmithSceneUI. Uživatelské rozhraní reprezentace pak kopíruje návrh. Samotný import pracuje jak s IFC souborem, tak s aktéry ve scéně. Funkcionalita pracuje s aktéry, protože v těch se nachází potřebné IFC atributy importované za pomocí Datasmith. Pomocí blueprint uzlu Get All Children jsou získány všechny dostupné materiály a je kontrolováno, zdali jsou zaškrtnuté či nikoliv. Ze zaškrtnutých widget tříd jsou získány vybrané materiály a z těch je vytvořena mapa s identifikací množiny materiálů jako klíčem a s vybraným materiálem jako hodnotou. Mapa je společně s aktéry náležejícími vybranému modelu předána Python skriptu využívajícímu opět knihovny IfcOpenShell – python. V rámci skriptu se získají ze vstupních aktérů Datasmith metadata (ta obsahují importované vlastnosti) a pomocí IFC GUID jsou objekty odpovídající aktérům vyhledány v IFC souboru.

V rámci importu materiálů se z elemetnu získává materiál pomocí funkce ifcopenshell.util.element.get_material(), ze kterého je získána identifikace (spojením všech obsažených materiálů) a podle identifikace je materiál hledán v mapě, zdali je určen k importu, či ne.

Ohledně importu vlastnosti HasOpenings se nejprve zkontroluje, zdali se jedná o stěnu, protože import má smysl pouze pro stěny. Pokud ano, jsou procházeny všechny elementy v množině HasOpenings a kontrolováno, zdali se jedná o dveře či o okna. Pokud je nalezeno okno, je nastaven boolean říkající, že je obsaženo okno a pokud jsou nalezeny dveře, je nastaven boolean říkající, že jsou obsaženy dveře.

Materiály i booleany jsou předány zpět do blueprint grafu společně s aktéry, kterých se informace týkají. V blueprint grafu se dále použije blueprint uzel Set Metadata Tag k nastavení těchto informací. Informace jsou nastaveny pro Static Mesh Actor, protože k těm je připojená geometrie, pro jejíž úpravu jsou informace potřeba. Přiřazení metadat je možno vidět na obrázku 3.4 a výňatky z kódu je možné vidět níže 3.2 3.3.

Výpis kódu 3.2 Nejdůležitější prvky python skriptu pro získání všech dostupných modelů

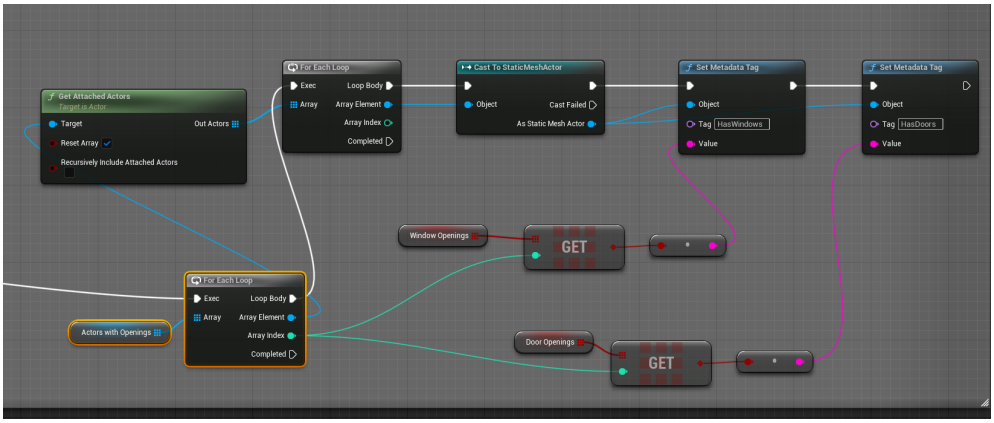
```
# Get all possible materials composites from
material_layer_sets = ifc_file.by_type("
    ↪ IfcMaterialLayerSet")
material_constituent_sets = ifc_file.by_type("
    ↪ IfcMaterialConstituentSet")
material_profile_sets = ifc_file.by_type("
    ↪ IfcMaterialProfileSet")
materials = ifc_file.by_type("IfcMaterial")

# Reduce materials
for material in materials:
    if len(material.AssociatedTo) > 0:
        materials_reduced.append(material.Name)

# Reduce material layer set
for layer in material_layer_sets:
    layers = ""
    for material in layer.MaterialLayers:
        layers = layers + ";" + material.Material.Name
    if layers not in material_layer_sets_reduced:
         material_layer_sets_reduced.append(layers)
```

Výpis kódu 3.3 Část kódu získávající vlastnost HasOpenings z python skriptu pro import dodatečných vlastností

```
# Has Openings import
if has_openings and ifc_element.is_a("IfcWall"):
    openings = ifc_element.HasOpenings

    # Set both types of opening on false
    doors_opening = False
    window_openi
    # Iterate through all the openings
    for opening in openings:
        # Get the opening element
        opening_element = opening.RelatedOpeningElement.
            ↪ HasFillings[0].RelatedBuildingElement
        if opening_element.is_a("IfcDoor"):
            doors_opening = True
        elif opening_element.is_a("IfcWindow"):
            window_open
    actors_output_has_openings.append(actor)
    has_openings_doors_output.append(doors_opening)
    has_openings_windows_output.append(window_opening)
```

3.2.4 Funkcionalita nastavení Kolizí

Nastavení kolizí se řídí podle návrhu a z větší části probíhá pouze za použití blueprint s menší kombinací s Python skriptem. V této části jsme narazili



Obrázek 3.4 Obrázek zobrazující blueprint uzly sloužící k zapsání získaných vlastností jako metadata pro aktéry Static Mesh Actor

na problém. Kolize je ideální řešit v rámci Static Mesh (geometrie), ale Datasmith metadata, říkající, o jakou entitu se jedná, jsou připojena k aktérům. Aktéři pak k ušetření paměti některé geometrie sdílí. Ke geometrii se můžeme s potřebnými informacemi dostat pouze skrze aktéry. A protože jedna geometrie má více aktérů, nastavovali by se kolize pro některé geometrie vícekrát.

Jako řešení se pro každý Static Mesh nastavují dočasná metedata říkající, že kolize již byly pro tento Static Mesh nastaveny, tím je přiřazování kolizí mnohem rychlejší a nenastavují se víckrát. Po dokončení nastavení kolizí jsou metadata zase smazána, tentokrát za pomoci procházení přímo složky s geometrií (k urychlení procesu). Cesta ke složce je získána spojením jména vybraného modelu a složky, do které se modely importují, proto pro správnou funkčnost není doporučeno měnit jména ani umístění modelů.

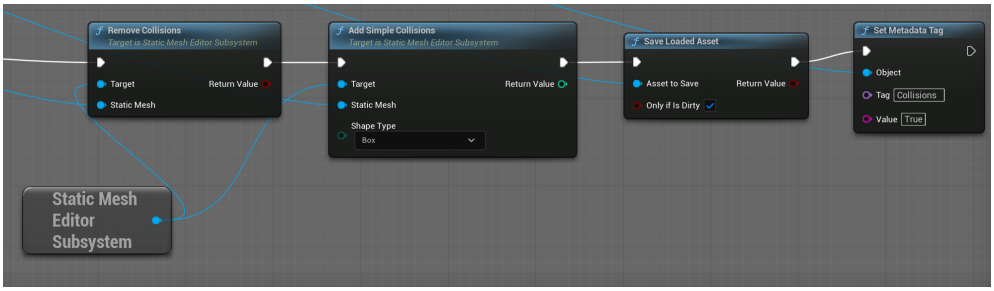
Nastavení kolizí se dělí na tři možné případy. K rozeznání, o jaký případ se jedná, je nutné získat Datasmith metadata určující typ IFC entity a přidaná metadata z minulé podsekce 3.2.3 určující, zdali má element okna či ne.

O první případ se jedná, pokud je IFC entita IFCDOOR, v tomto případě se skrze blueprint uzel Remove Collisions odstraní veškeré kolize a změny jsou uloženy.

V druhém případě se jedná o entitu IFCSLAB reprezentující podlahu nebo o entitu IFCWALL, která obsahuje dveře. V těchto případech je využit Python skript a nastavení vlastnosti collision_trace_flag k použití komplexních kolizí (kolize se budou počítat podle přesné geometrie objektu). Kód 3.4je přiložený níže.

Třetí případ pokrývá všechny ostatní možnosti, kdy se za pomocí blueprint uzlu Add Simple Collisions přidá jednoduché ohraničení celého objektu v podobě kvádru. Nastavení těchto kolizí je možno vidět na obrázku 3.5.

Výpis kódu 3.4 Python skript nastavující kolize na komplexní namísto jednoduchých



Obrázek 3.5 Obrázek zobrazující blueprint uzly nastavující jednoduché kolize v podobě kvádru

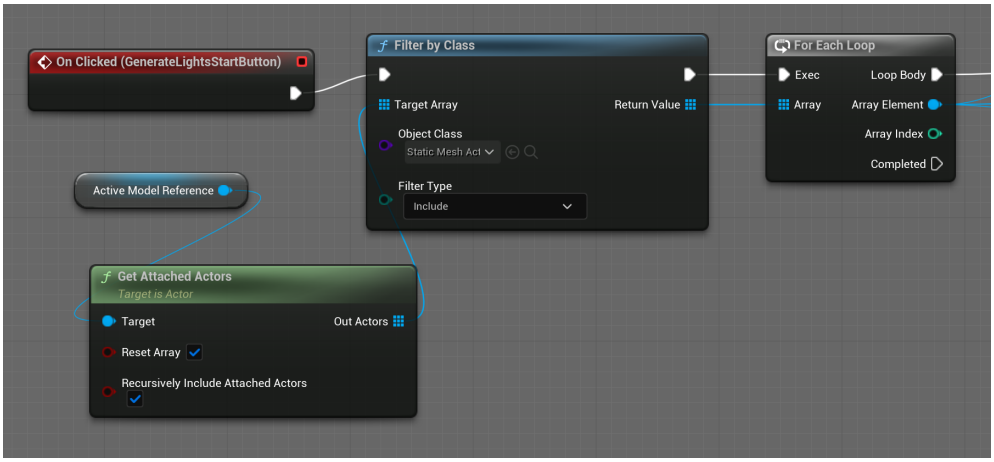
```
# Get the StaticMesh's BodySetup to access collision
    ↪ settings
body_setup = static_mesh.get_editor_property('body_setup'
    ↪ )


# Set the collision complexity
if body_setup:
    body_setup.set_editor_property('collision_trace_flag'
        ↪ , unreal.CollisionTraceFlag.
        ↪ CTF_USE_COMPLEX_AS_SIMPLE)
```

3.2.5 Funkcionalita generování světel

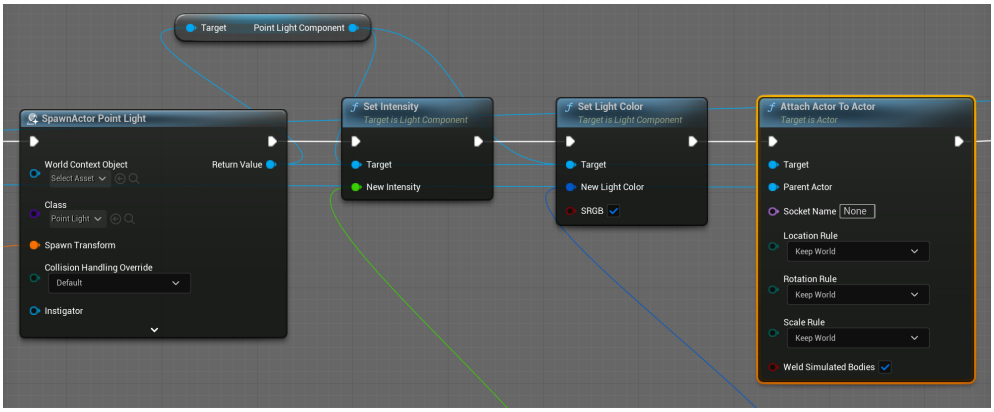
Tato funkcionalita je z většiny inspirovaná návrhem a nachází se v uživatelském rozhraní po zvolení možnosti pro generování světel. Uživatelské rozhraní odráží návrh, a tak obsahuje možnost měnit barvu a svítivost světla společně s tlačítky pro jejich generování a odstranění.

Protože světla se přidávají pouze do scény, pracuje tato funkcionalita s aktéry, které získává z reference na vybraný model (Datasmith Scene Actor). K získání se používá uzel Get Attached Actors, který vyhledá všechny připojené aktéry k vybranému modelu a k filtrování se používá uzel Filter By Class, jak je vidět na obrázku 3.6. Získaní aktéři jsou všichni třídy Static Mesh Actor a pomocí Datasmith metadat se určuje, zdali se jedná o entitu IFCCOVERING (strop).

Aby se negenerovalo světlo již na místě jiného světla (při opakovaném zmáčknutí tlačítka pro generování), jsou k aktérům po vygenerování přiřazena metadata o již vygenerovaném světle. Generování pak využívá uzlu Get Actor Bounds, který určuje přibližné rozměry stropu. Protože jsou stropy ploché, určí se pak nejmenší rozměr a o ten je světlo při vygenerování posunuto dolů, aby se nacházelo pod stropem. Ke generování je pak využit uzel Spawn Actor a k přiřazení ke stropu je využit uzel Attach Actor To Actor. Generování a připojení je pak vidět na následujícím obrázku 3.7



Obrázek 3.6 Obrázek zobrazující jak za pomoci blueprint uzlů získat všechny Static Mesh Actors připojené k vybranému modelu

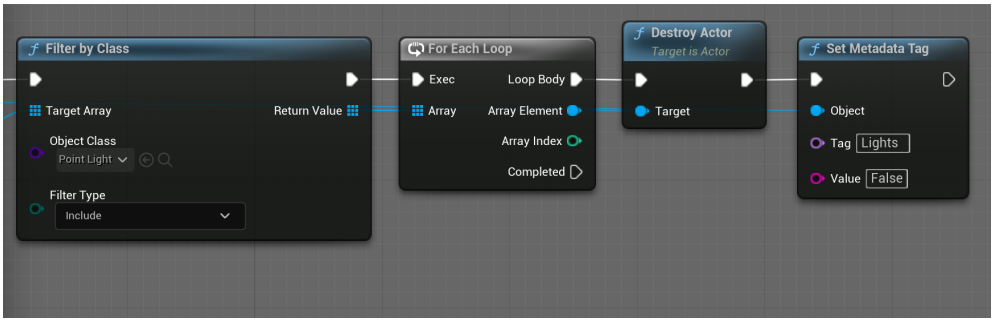


Obrázek 3.7 Obrázek zobrazující blueprint uzly sloužící k vygenerování bodového světla, k následnému nastavení jeho parametrů a k připojení k jinému aktérovi

Opačnou operací je odstranění světel, která opět získá aktéry třídy Static Mesh Actor a pokud obsahují metadata přidaná v operaci generování světel a jsou nastavená na True, jsou pro tohoto aktéra světla odstraněna pomoci uzlu Destroy Actor a následně jsou metadata nastavena na False. Operace je vidět na obrázku 3.8.

3.2.6 Funkcionalita přiřazování materiálů

Tato funkcionalita je implementovaná podle návrhu a jako jediná obsahuje ve své implementaci C++ kód. Jsou zde také využity takzvané String Tables, které uchovávají pravidla pro přiřazení snadno přístupná uživateli. Pro správný průběh funkce musí být splněny dva požadavky. String Table musí obsahovat pravidla pro obsažené IFC entity v modelu a importované materiály a pro-



Obrázek 3.8 Obrázek zobrazující blueprint uzly sloužící k odstranění aktéra bodového světla a změnění metadat k indikaci absence generovaného světla

jekt musí obsahovat vytvořené Unreal Engine materiály referencované cestou v pravidlech. Například pravidla pro IfcEntitu mají formát IFCENTITY jako klíč a referenční cestu k materiálu jako hodnota.

Procházení aktérů a zamezení nastavení stejného materiálu pro jednu geometrii vícekrát funguje na úplně stejném principu jako v podsekci 3.2.4, věnující se nastavení kolizí. K přiřazení materiálu slouží uzel Set Material. Zbytek funkcionality má na starost implementovaná C++ funkce s názvem Assign Material with Ifc Properties.

Tato funkce je vystavená blueprint třídám, a proto je její funkcionalitu možné použít v blueprint grafu. K označení, že je funkce vystavená pro blueprint, slouží řádek `UFUNCTION(BlueprintCallable, Category = "My Assign material")`, umístěný nad její inicializací. Parametry funkce označené jako const jsou Unreal Engine automaticky nastaveny jako vstupní proměnné uzlu. Na druhou stranu parametry, které jsou předány jako reference, jsou automaticky nastavené jako výstupní proměnné uzlu.

Funkce přijímá jako vstupní parametry IFC entitu, IFC materiál a údaj o prioritizaci tabulek. Jako výstupní parametry má Unreal Engine materiál připravený k přiřazení a zprávu o výsledku.

Samotná funkce implementuje načtení tabulek String Table. Tabulka pro IFC entitu je načtena vždy, protože slouží jako základní volba. Pokud je prioritizované přiřazení za pomocí materiálu, je načtena i tabulka pro materiály.

Následně se podle zadané priority funkce pokusí najít materiál či entitu v určené tabulce a pokud pravidlo pro daný klíč neexistuje či je cesta k Unreal Engine neplatná, vyzkouší se defaultní přístup, kdy hierarchie je následující:

1. Přiřazení podle materiálu.
2. Přiřazení podle entity.
3. Přiřazení defaultního materiálu.

Použití implementované funkce jako blueprint uzel, je možné vidět na obrázku 3.2. Části C++ kódu 3.5 3.6 jsou pak níže.

Výpis kódu 3.5 Hlavičkový soubor C++ třídy implementující funkci AssignMaterialWithIfcProperties

```
#pragma once

#include "CoreMinimal.h"
#include "Kismet/BlueprintFunctionLibrary.h"
#include "Engine/DataTable.h"
#include "Internationalization/StringTable.h"
#include "Internationalization/StringTableCore.h"
#include "Internationalization/TextKey.h"

#include "AssignMaterialHelper.generated.h"


/**
 *
 */
UCLASS()
class IMPORTBIMFINALTHESIS_API UAssignMaterialHelper :
    ↪ public UBlueprintFunctionLibrary
{
    GENERATED_BODY()

    public:

        UFUNCTION(BlueprintCallable, Category = "My␣
            ↪ Assign␣material")
            static void AssignMaterialWithIfcProperties(
                ↪ const FString IfcEntity, const FString
                ↪ IfcMaterial, const FString Priority,
                ↪ UMaterial*& AssignMaterial, FString&
                ↪ OutputLog);
};
```

Výpis kódu 3.6 Načtení a použití String Table s pravidly pro přiřazení materiálu dle IFC Materiálu. Část kódu je z funkce AssignMaterialWithIfcProperties

```
// Load the material string table
LoadedStringTableMaterial = LoadObject<UStringTable>(
    ↪ nullptr, TEXT("/Game/BimImportFunctionality/
    ↪ MaterialsToAssign/AccessibleMaterialsIfcMaterial.
    ↪ AccessibleMaterialsIfcMaterial"));

if (LoadedStringTableMaterial)
{
    FStringTableConstRef Table =
        ↪ LoadedStringTableMaterial->GetStringTable();
    FString SourceString = "";
    // Try to retrieve a value wit given key
```

```
    if (Table->GetSourceString(TextKeyMaterial,
       ↪ SourceString))
    {
        // Try to retrieve found material
        AssignMaterial = LoadObject<UMaterial>(nullptr, *
           ↪ SourceString);
        if (AssignMaterial)
        {
            // Return material if sucessful, if not try
               ↪ next approach from hierarchy -> use IFC
               ↪ Entity
            OutputLog = TEXT("Material");
            return;
        }
    }
}
```

3.2.7 Log

Log je implementovaný za pomoci textového pole, jehož text se formátuje pomocí uzlu Format Text. Formátování probíhá načtením starého textu a přidáním nové řádky s novým textem. Log je v uživatelském rozhraní všude přítomen a slouží k informování uživatele o probíhajících procesech a chybách.

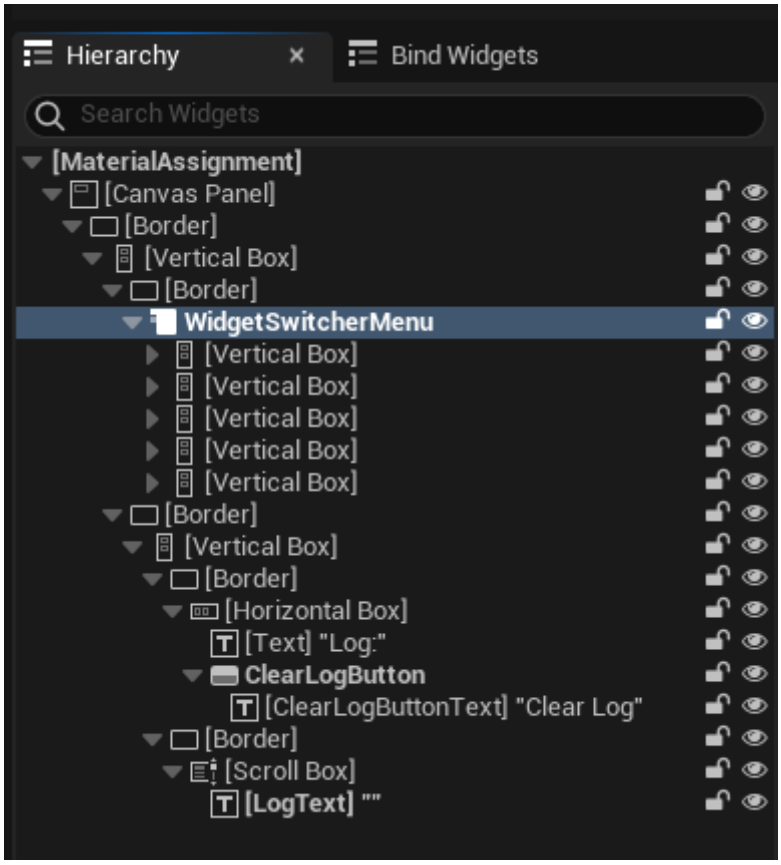
3.2.8 Python skripty

Velká část implementace obsahuje ve své funkcionalitě Python skripty. Ty jsou všechny umístěné v příloze práce a jsou spouštěny za pomoci blueprint uzlu Execute Python Script. V tomto uzlu je možné nastavit vstupní a výstupní proměnné, které je ve skriptu možno podle jména přímo použít. Například si nastavím jako vstupní proměnnou input a tím můžu proměnnou input ve skriptu číst a používat bez jejího nastavení. Pro výstupní proměnné to funguje podobně, akorát jsou ve skriptu nastavovány.

3.2.9 Uživatelské rozhraní

Uživatelské rozhraní je vytvořeno za pomoci Editor Utility Widget. V tom jsou kombinací grafických prvků dostupných od Unreal Engine tvořená uživatelská rozhraní podle návrhu. Celé uživatelské rozhraní se skládá z Logu a z takzvaného WidgetSwitcheru, kterým se přepínají všechny ostatní části Uživatelského rozhraní. Log je tak neměnný a mění se pouze horní část obrazovky, jak je vidět na obrázcích 3.9 a 3.10.

Protože se z velké části drželo uživatelské rozhraní návrhu, uvedeme si zde hlavně příklad, kde se uživatelské rozhraní změnilo nejvíce. Tím je menu pro import dodatečných vlastností. To se od návrhu liší obsahem zaškrtávátka pro



Obrázek 3.9 Obrázek zobrazující Unreal Engine hierarchii implementovaného Uživatelského rozhraní. Na obrázku je vidět část Log a Widget Switcher, který pokrývá všechny ostatní části uživatelského rozhraní

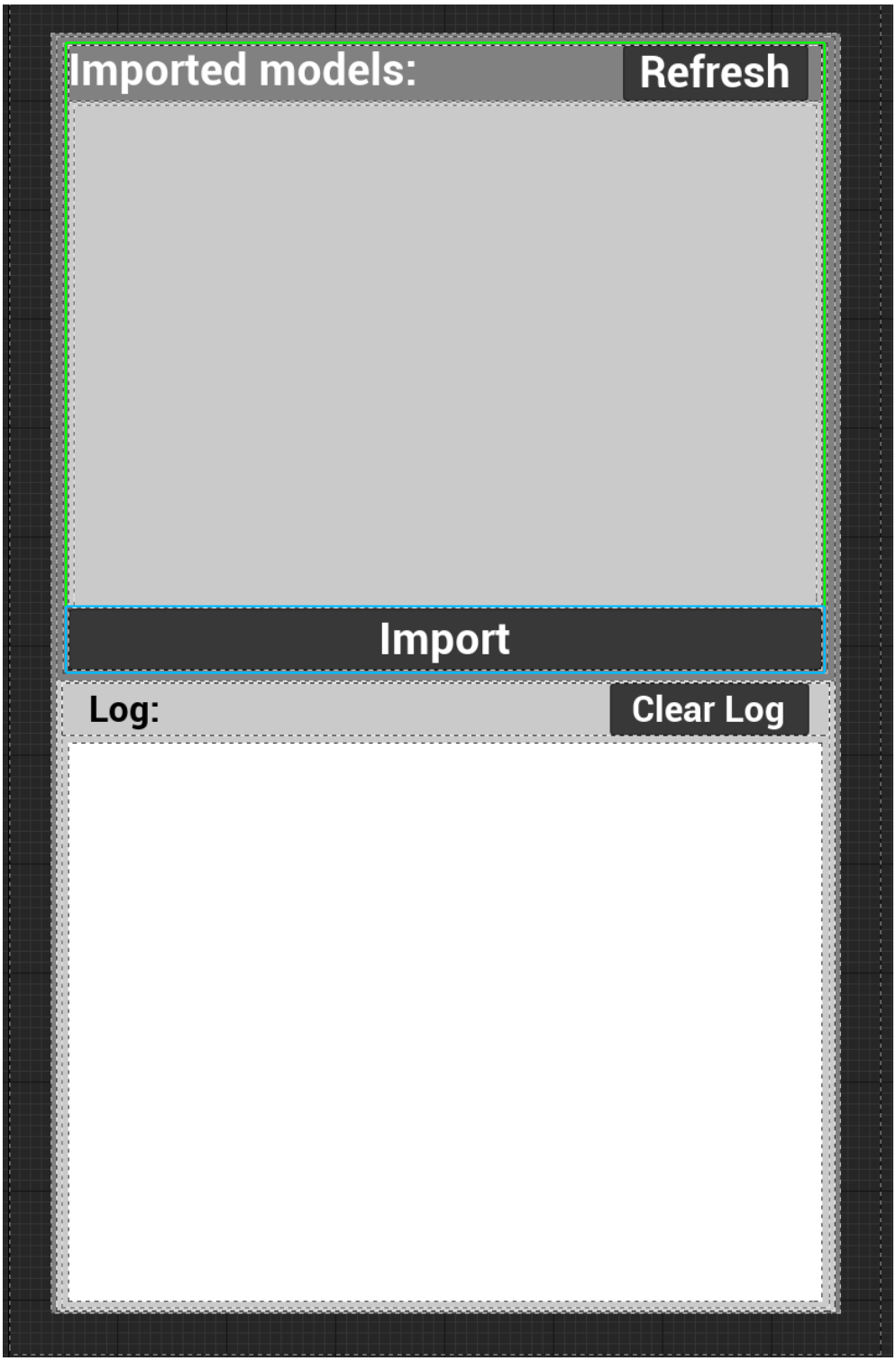
import vlastností sloužících k nastavení kolizí a jiným rozestavením tlačítek, jak je vidět na obrázku 3.11. V obrázku je také vidět reprezentace pro dostupné množiny materiálu.

3.2.10 Blueprint makra

Blueprint graf obsahuje malé množství implementovaných blueprint skupin uzlů uzavřených do takzvaného makra. Mezi tato makra patří využití Python skriptu k rozdělení řetězce písmen do pole. Dělení probíhá za pomoci středníku. Dále sem spadá makro využívající Python k opětovnému slepení řetězce písmen.

K využívání logu existují makra Clear Log a Write Log, sloužící k zápisu a k vymazání celého logu.

Poslední makro slouží k získání cesty k importovanému modelu za pomoci názvu Datasmith Scene Actor.



Obrázek 3.10 Obrázek zobrazující jak vypadá uživatelské rozhraní pro přehled importovaných modelů v UI editoru

Obrázek 3.11 Obrázek zobrazující jak vypadá uživatelské rozhraní při jeho spuštění. Zobrazená část se týká importu dodatečných vlastností

3.3 Obsah složky BimImportFunctionality

Většina funkcionality je zabalená ve složce BimImportFunctionality, proto se v této sekci zaměříme na obsah této složky.

3.3.1 Editor Utility Widget

Složka obsahuje hlavní Editor Utility Widget se jménem MaterialAssignment, zastřešující většinu funkcionalit.

3.3.2 DatasmithIfcImport

BimImportFunctionality obsahuje složku DatasmithIfcImport, do které se importují veškeré IFC modely. Ve složce jsou modely reprezentovány geometrií, materiály a souborem DatasmithScene.

3.3.3 Subwidgets

Subwidgets složka obsahuje dvě Editor Utility Widget, sloužící jako reprezentace pro dostupné materiály a dostupné modely.

3.3.4 MaterialsToAssign

MaterialsToAssign složka obsahuje jak Unreal Engine materiály, které je možné přiřadit ke geometrii, tak pravidla pro jejich přiřazení. Pravidla se nacházejí ve String Tables s názvy AccessibleMaterialsIfcMaterial pro přiřazení podle materiálu a AccessibleMaterialsIfcEntity pro přiřazení podle entity.

Kapitola 4

Testování

Testování proběhlo na dvou testovacích souborech, které byly poskytnuty od zadavatele. Soubor `University Building Model.ifc` funguje s funkcionalitou bez větších problémů, soubor je bezproblémově importován a obsahuje všechny IFC entity a vlastnosti nutné pro správný běh projektu.

Testování bylo provedeno i na souboru `ZaB_only_floors_doors_doorWalls.ifc`, který je také bezproblémově importován, ale další funkce nemají příliš velké uplatnění. Světla se kvůli absenci stropů nemají k čemu přiřadit. Materiály z důvodu obsahu pouze jedné IFC entity a žádných IFC materiálů nelze přiřadit (přiřadí se pouze jeden materiál podle obsažené entity). Kolize se přiřadí všude jednoduché, což umožňuje průchod scénou, ale pouze její částí. K procházení scény je také nutné model ručně škálovat.

Kapitola 5
Závěr

Zadání této práce bylo vytvořeno společností T-SOFT a.s., která si klade za dlouhodobý cíl importovat BIM modely do Unreal Engine a provádět na nich simulace. Prvním krokem k dosažení tohoto cíle je import těchto BIM modelů do Unreal Engine a jejich vizualizace, což bylo také hlavním cílem této práce. Na začátku byla provedena důkladná analýza BIM modelů a formátu IFC, následovaná analýzou možností pro jejich import do Unreal Engine a možností jejich vizualizace. Součástí analýzy byla také revize aplikací, které již s BIM modely pracují, aby bylo možné načerpat inspiraci v používaných procesech.

Z těchto analýz vyplynuly konkrétní způsoby, jak dosáhnout stanovených cílů. Tyto způsoby byly promítnuty do návrhu řešení, který následně sloužil jako základ pro implementaci prototypu funkcionalit a uživatelského rozhraní k jejich použití. Implementované funkcionality poskytnou firmě T-SOFT solidní základ pro další rozvoj a naplnění jejich vizí.

Celý proces realizace je podrobně popsán v tomto dokumentu. Závěrem mohu říci, že práce splňuje stanovené požadavky, a proto byl cíl práce úspěšně splněn.

Bibliografie

1. EASTMAN, Chuck; AL., et. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*. John Wiley & Sons, 2011.
2. KUBBA, Sam et al. *Handbook of Green Building Design and Construction: LEED, BREEAM, and Green Globes*. 1. vyd. Waltham: Butterworth-Heinemann, 2012. LEED AP.
3. NBS. *NBS BIM Object Standard* [Online]. NBS, 2023. Dostupné také z: `https://www.thenbs.com/our-tools/nbs-bim-object-standard`. Accessed: 2024-04-01.
4. MASTERSPEC. *International BIM Object Standard 2016 2nd Draft* [Online]. 2016. Dostupné také z: `https://masterspec.co.nz/filescust/CMS/International%5C%20BIM%5C%20Object%5C%20Standard%5C%202016%5C%202nd%5C%20Draft.pdf`. Accessed:2024-04-01.
5. NATSPEC CONSTRUCTION INFORMATION. *OBOS Open BIM Object Standard: An International Standard for Object Developers* [Online]. 2018. Ver. 1.0. Dostupné také z: `https://bim.natspec.org/documents/open-bim-object-standard`. Accessed: 2024-04-01.
6. BUILDINGSMART INTERNATIONAL. *IFC 4.3.2.20240423 (IFC4X3_ADD2)* [Online]. [B.r.]. IFC4X3_ADD2. Dostupné také z: `https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/content/scope.htm`. Under development, Accessed: 2024-04-01.
7. NATIONAL INSTITUTE OF BUILDING SCIENCES. *Construction to Operations Building Information Exchange (COBIE) V3* [Online]. 2023. Dostupné také z: `https://www.nibs.org/nbims/cobie/v3`. Available at National BIM Standard United States, Accessed: 2024-04-01.
8. HAMIL, Stephen. *What is COBie?* [Online]. NBS, 2018. Dostupné také z: `https://www.thenbs.com/knowledge/what-is-cobie`. Accessed: 2024-04-09.

9. DATACOMP SP. Z O.O. *BIMvision* [Online]. Accessed: 2024. Dostupné také z: `https://bimvision.eu/about/`. Dostupné z: https://bimvision.eu/about/.
10. *FreeCAD* [Online]. Accessed: 2024. Dostupné také z: `https://www.freecad.org`. Dostupné z: https://www.freecad.org.
11. FREECAD COMMUNITY. *Arch Workbench* [Online]. Accessed: 2024. Dostupné také z: `https://wiki.freecad.org/Arch_Workbench`. Dostupné z: https://wiki.freecad.org/Arch$_W orkbench$.
12. FREECAD COMMUNITY. *BIM Workbench* [Online]. Accessed: 2024. Dostupné také z: `https://wiki.freecad.org/BIM_Workbench`. Dostupné z: https://wiki.freecad.org/BIM$_W orkbench$.
13. *XBIM Toolkit* [Online]. Accessed: 2024. Dostupné také z: `https://docs.xbim.net`. Dostupné z: https://docs.xbim.net.
14. IFCOPENSHELL. *BlenderBIM* [Online]. Accessed: 2024. Dostupné také z: `https://blenderbim.org`. Dostupné z: https://blenderbim.org.
15. EPIC GAMES. *Unreal Engine 5* [Online]. Accessed: 2024. Dostupné také z: `https://www.unrealengine.com/en-US/unreal-engine-5`. Dostupné z: https://www.unrealengine.com/en-US/unreal-engine-5.
16. BUŠEK, Tadeáš. *Věnná města Českých Královen - Import 3D modelů do Unreal Engine 4*. Praha, 2021. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
17. EPIC GAMES. *Datasmith Plugins Overview* [Online]. Accessed: 2024. Dostupné také z: `https://dev.epicgames.com/documentation/en-us/unreal-engine/datasmith-plugins-overview?application_version=5.1`. [cit. 17.4.2024]. Dostupné z: https://dev.epicgames.com/documentation/en-us/unreal-engine/datasmith-plugins-overview?application$_v ersion = 5.1$.
18. IFCQUERY. *IFC++* [Online]. Accessed: 2024. Dostupné také z: `https://github.com/ifcquery/ifcplusplus`. [cit. 1.5.2024]. Dostupné z: https://github.com/ifcquery/ifcplusplus.
19. IFCOPENSHELL TEAM. *IfcOpenShell* [Online]. Accessed: 2024. Dostupné také z: `https://ifcopenshell.org`. [cit. 28.4.2024]. Dostupné z: https://ifcopenshell.org.
20. EPIC GAMES. *Unreal Engine Materials* [Online]. Accessed: 2024. Dostupné také z: `https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-materials?application_version=5.1`. [cit. 2.5.2024]. Dostupné z: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-materials?application$_v ersion = 5.1$.
21. DUPREZ, Hugo. *Texture Lab* [Online]. Accessed: 2024. Dostupné také z: `https://www.texturelab.xyz`. Dostupné z: https://www.texturelab.xyz.

22. POLYCAM. *AI Texture Generator* [Online]. Accessed: 2024. Dostupné také z: `https://poly.cam/tools/ai-texture-generator`. Dostupné z: https://poly.cam/tools/ai-texture-generator.

23. LARK PROJECT CONTRIBUTORS. *Lark Parser* [Online]. Accessed: 2024. Dostupné také z: `https://github.com/lark-parser/lark`. Dostupné z: https://github.com/lark-parser/lark.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `impl` ..... odkaz na gitlab s unreal engine projektem
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF