

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

**Rozšírenia a ukážková scéna pre systém
virtuálnych výučbových prostredí**

Diplomová práca

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

**Rozšírenia a ukážková scéna pre systém
virtuálnych výučbových prostredí**

Diplomová práca

Študijný program: Informatika
Študijný odbor: 9.2.1. Informatika
Školiace pracovisko: Katedra počítačov a informatiky (KPI)
Školiteľ: Ing. Štefan Korečko, PhD.

Košice 2025

Bc. Kristián Zsigó

Abstrakt v SJ

Cieľom tejto diplomovej práce je vytvoriť rozšírenia vo forme nových komponentov a ukázkovú scénu pre systém virtuálnych výučbových prostredí, navrhnutom v rámci A-Frame a riadený Petriho sieťami. Riešenie navrhnuté v rámci tejto práce ponúka širokú sadu nových univerzálnych, ale aj ukázkových komponentov, ktoré sú integrované v ukázkovej scéne, pričom ako celok tvoria akýsi návod pre nových vývojárov projektu, ako čo najrýchlejšie a najefektívnejšie vytvárať nové scény v projekte. Tomuto návrhu predchádzala dôkladná analýza súčasného stavu projektu, ako aj iných výskumov, ktoré sa zaoberajú tvorbou pútavých scenárov a zaujímavých interakcií s prostredím vo virtuálnej realite, hlavne so zameraním na edukačné hry.

Kľúčové slová v SJ

virtuálna realita, A-Frame, edukačné hry, Petriho siete, Ammo.js

Abstrakt v AJ

The goal of this thesis is to create extensions in the form of new components and a showcase room for a virtual learning environments system, designed using A-Frame framework and controlled by Petri nets. The solution designed in this thesis offers a wide range of new reusable components, which are integrated in the showcase room, forming a kind of guide for new developers of the project on how to create new scenes in the project as quickly and efficiently as possible. This design was preceded by a thorough analysis of the current state of the project, as well as other research that deals with creating engaging scenarios and interesting interactions with the environment in virtual reality, mainly focusing on educational games.

Kľúčové slová v AJ

virtual reality, A-Frame, education games, Petri nets, Ammo.js

Bibliografická citácia

ZSIGÓ, Kristián. *Rozšírenia a ukázková scéna pre systém virtuálnych výučbových prostredí*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2025. 75s. Vedúci práce: Ing. Štefan Korečko, PhD.

TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
Katedra počítačov a informatiky

Z A D A N I E
D I P L O M O V E J P R Á C E

Študijný odbor: **Informatika**

Študijný program: **Informatika**

Názov práce:

Rozšírenia a ukážková scéna pre systém virtuálnych výučbových prostredí

Enhancements and Demo Scene for Virtual Educational Environment System

Študent: **Bc. Kristián Zsigó**

Školiteľ: **Ing. Štefan Korečko, PhD.**

Školiace pracovisko: **Katedra počítačov a informatiky**

Konzultant práce:

Pracovisko konzultanta:

Pokyny na vypracovanie diplomovej práce:

1. Oboznámiť sa so systémom virtuálnych výučbových prostredí so scenármi vo forme Petriho sietí, vytvorenom na domovskom pracovisku.
2. Analyzovať súčasný stav systému a podobné existujúce riešenia.
3. Na základe analýzy navrhnúť a implementovať vybrané rozšírenia systému virtuálnych výučbových prostredí.
4. Navrhnúť a implementovať ukážkové prostredie demonštrujúce dostupnú funkcionality.
5. Návrhové a implementačné práce koordinovať s ďalšími riešiteľmi prác na systéme virtuálnych výučbových prostredí.
6. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský

Termín pre odovzdanie práce: 17.04.2025

Dátum zadania diplomovej práce: 31.10.2024



prof. Ing. Liberios Vokorokos, PhD.

dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 17.4.2025

.....

Vlastnoručný podpis

Podakovanie

Na tomto mieste by som rád poďakoval Ing. Štefanovi Korečkovi, PhD ako svojmu vedúcemu práce za jeho čas a odborné vedenie a návrhy niektorých riešení počas tvorby tejto záverečnej práce.

Rovnako by som sa rád poďakoval svojim rodičom a priateľom za ich podporu, povzbudzovanie a hlavne trpezlivosť v náročných chvíľach počas celého trvania môjho štúdia.

Obsah

Úvod	1
1 Edukačné hry vo virtuálnej realite	4
2 Analýza aktuálneho riešenia	6
2.1 Ciele analytickej časti	6
2.2 Opis aplikácie	6
2.3 Aktuálny stav aplikácie	7
2.4 Limitácie riešenia	8
2.5 Analýza najnovších prác na aktuálnom riešení	8
2.5.1 Scenárom riadené virtuálne výučbové prostredie pre geo- grafiu	8
2.5.2 Systém virtuálnych výučbových prostredí riadených pro- cesnými grafmi	9
3 Systematická analýza a prehľad podobných riešení	10
3.1 Metodika získavania dát a ich spracovanie	10
3.2 Analýza získaných výsledkov	12
3.3 Analýza podobných riešení	15
3.3.1 A Review on Augmented Reality Authoring Toolkits for Edu- cation	15
3.3.2 Virtual reality and augmented reality in social learning spa- ces a literature review	16
3.3.3 Gamified Evaluation in STEAM for Higher Education: A Case Study	16
3.3.4 Virtual Worlds for Learning in Metaverse: A Narrative Review	17
3.3.5 Education in Collaborative Immersive Virtual Reality	18
3.3.6 Exploring Mixed Reality Level Design Workflows	18
3.3.7 Virtual Reality STEM Education from a Teacher's Perspective	18

3.3.8	Virtual Museum Gamification for Discovery-Based Online Learning in the Metaverse	19
3.3.9	On the role of interaction mode and story structure in virtual reality serious games	19
4	Záverý vyplývajúce z analýzy	20
5	Návrh nových riešení	22
6	Dizajn ukázkovej Scény	23
6.1	Model scény	23
7	Scenár úloh	26
7.1	Miestnosť 1: Hádzanie lopty do koša	28
7.1.1	Popis miestnosti	28
7.1.2	Riadiaca Petriho podsieť	30
7.2	Miestnosť 2: Usporiadanie predmetov	31
7.2.1	Popis miestnosti	32
7.2.2	Riadiaca Petriho podsieť	33
7.3	Miestnosť 3: Triedenie predmetov	34
7.3.1	Popis miestnosti	34
7.3.2	Riadiaca Petriho podsieť	37
8	Implementácia ukázkovej scény	40
8.1	Oprava a vylepšenie existujúcich komponentov	43
8.1.1	Fyzikálny engine Ammo.js	44
8.1.2	Komponent animation-mixer	44
8.1.3	Komponent autofit-gltf	44
8.1.4	Úpravy komponentu Blink controls	46
8.1.5	Nová verzia komponentu Super Hands	48
8.1.6	Prepracovaný komponent aframe-keyboard	49
8.1.7	Nový komponent pn-transition-handler	52
8.1.8	Zmena komponentu petri-net-sim	54
8.1.9	Zjednodušenie a úprava skriptu scény	57
8.2	Nové funkcionality a rozšírenia	58
8.2.1	Komponent ammo-collision-transition	58
8.2.2	Komponent button-control	60
8.2.3	Komponent lever-control	61
8.2.4	Komponent enable-text-on-marking	62

8.2.5	Komponent <i>text-marking-update</i>	63
8.2.6	Komponent <i>timer</i>	64
8.2.7	Komponent <i>show-finish-info</i>	65
8.2.8	Ďalšie komponenty v scéne	66
8.2.9	Zhrnutie	67
9	Vyhodnotenie	68
9.1	Výsledky prieskumu	69
9.1.1	Výsledky SUS	69
9.1.2	Výsledky kontrolných otázok	70
10	Záver	72
	Literatúra	73
	Slovník	76
	Zoznam príloh	77
A	Používateľská príručka	78
A.1	Funkcia programu	78
A.2	Spustenie a inštalácia programu pre administrátorov	78
A.2.1	Požiadavky na technické prostriedky	78
A.2.2	Spustenie projektu	79
A.2.3	Spustenie a konfigurácia systému	82
A.2.4	Prístup k aplikácii	82
A.2.5	Manuálna konfigurácia administrátora	82
A.2.6	Pridanie administrátorského účtu	84
A.3	Použitie programu	85
A.3.1	Prihlasovacia obrazovka	85
A.3.2	Hlavná obrazovka	86
A.3.3	Dashboard pre učiteľov a adminov	87
A.3.4	Virtuálna scéna	89
A.3.5	Manipulácia s objektami	90
A.3.6	Otváranie dverí	90
A.3.7	Písanie na klávesnici	92
B	Systémová príručka	94
B.1	Funkcia programu	94
B.2	Popis programu	94

B.2.1	Popis riešenia	94
B.3	Popis komponentov	97
B.3.1	Aframe komponenty	97
B.3.2	Typescript objekty	98
B.4	Využíte technológie	98
B.5	Spustenie programu	98
B.5.1	Stiahnutie zdrojových kódov	99
B.5.2	Otvorenie v IDE	99
B.5.3	Príprava projektu na spustenie	100
B.5.4	Spustenie projektu	101
B.6	Nasadenie na server	102
B.6.1	Nahratie potrebných obrazov kontajnerov do Dockerhubu	103
B.6.2	Stiahnutie obrazov na server	103
B.6.3	Spustenie systému	104

Zoznam obrázkov

3.1	Graf zobrazujúci rok vydania publikácii	12
3.2	Graf zobrazujúci krajinu vydania publikácii	13
3.3	PRISMA 2020 vývojový diagram pre nové systematické prehľady, ktoré zahrňujú vyhľadávanie iba v databázach	14
6.1	Pohľad na pôvodný model garáže zvonku	24
6.2	Pohľad na upravenú verziu garáže zvonku	24
6.3	Pohľad na pôvodný model garáže zvnútra	25
6.4	Pohľad na upravenú verziu garáže zvnútra	25
7.1	Kompletná Petriho sieť riadiaca priebeh úloh v ukážkovej scéne . .	27
7.2	Dvere do prvej miestnosti	28
7.3	Prvá miestnosť – časť so skenerom	29
7.4	Prvá miestnosť – časť s košmi	29
7.5	Bonus v prvej miestnosti	30
7.6	Petriho sieť riadiaca priebeh úloh v prvej miestnosti	31
7.7	Dvere do druhej miestnosti	32
7.8	Druhá miestnosť - kôš s predmetmi so stolom na ich usporiadanie .	33
7.9	Petriho sieť riadiaca priebeh úloh v druhej miestnosti	34
7.10	Nápoveda k tajným dverám	35
7.11	Otvorenie tajných dverí pomocou klávesnice	35
7.12	Informačná tabuľa nad košmi pred a po uložení predmetov	36
7.13	Tretia miestnosť - stôl s predmetmi a košmi	37
7.14	Podsieť riadiaca zobrazenie klávesnice a textového vstup do tretej miestnosti	38
7.15	Petriho sieť riadiaca priebeh úloh v tretej miestnosti	39
8.1	Štruktúra projektového adresára	42
8.2	Kolízna mapa miestnosti	46
8.3	Príklad úspešnej teleportácie	47

8.4	Príklad neúspešnej teleportácie	48
8.5	Ukážka použitia komponentu Super Hands na interakciu s objektami	49
8.6	Ukážka manipulácie s klávesnicou vo VR	51
8.7	Zdrojový kód nových rozhraní skriptu scény	57
8.8	Fungovanie komponentu páčky	62
8.9	Ukážka komponentu <i>enable-text-on-marking</i>	63
8.10	Ukážka komponentu <i>show-finish-info</i>	66
9.1	Štruktúra projektového adresára	70
A.1	Ukážka súboru <code>docker-compose.yaml</code>	81
A.2	Ukážka súboru <code>servers.json</code>	82
A.3	Okno so servermi v pgAdmin	83
A.4	Databáza v pgAdmin	84
A.5	Prihlasovacie okno	85
A.6	Hlavná obrazovka	86
A.7	Výber úlohy	87
A.8	Používateľský dashboard	88
A.9	Prehľad histórie úloh	89
A.10	Manipulácia s objektami v scéne	90
A.11	Dvere s tlačidlom	91
A.12	Dvere s páčkou	91
A.13	Tajné dvere	92
A.14	Klávesnica	93
B.1	Štruktúra komponentov	96
B.2	Otvorenie projektu	99
B.3	Príklad súboru <code>application.properties</code>	100
B.4	Maven závislosti	101
B.5	Spustenie servera	102
B.6	Úvodná obrazovka aplikácie	102
B.7	Vylistovanie obrazov	103
B.8	Tagovanie obrazu	103
B.9	Push do Dockerhubu	103

Zoznam tabuliek

2.1	Výskumné otázky a ich motivácia	6
3.1	Dopyty použité na prehľadávanie databáz	11

Úvod

Tak, ako téma spojenia virtuálnej reality a výučby školských predmetov bola aktuálna na prelome tisícročia [1],[2], tak aj dnes je stále aktuálna a žiadaná [3]. Pre tvorbu takýchto edukačných hier je nutné vytvoriť dostatočne interaktívne prostredie, pútavý príbeh, ale aj správny návrh aplikácie, aby sa študenti čo najviac naučili. Vďaka väčšej miere vtiahnutia hráča do deja, lepšej voľnosti a možnostiam, ktoré ponúka virtuálna realita, je možné dosiahnuť lepšie výsledky v učení [4]. Taktiež pridanie viacerých možností, ako hru dokončiť, tvorba virtuálnych avatarov a možnosť pracovať na úlohách skupinovo prispieva k zvýšenému záujmu študentov o tému obsiahnutú v hre [5]. Aj keď existuje niekoľko aplikácií na vytváranie edukačných hier a taktiež projektov, ktoré sú dostatočne interaktívne a pútavé, stále ich rozšírenie v praxi je značne obmedzené. Dôvodom môže byť náročnosť vytvárania takýchto aplikácií, ale aj nedostatočná znalosť programovacích jazykov a nástrojov na tvorbu hier. Totiž návrh stratégie vývoja hry vo virtuálnej realite je omnoho zložitejší, ako v prípade klasických hier, pretože dostupnosť vhodných návodov a nástrojov, ako aj vývojárov je značne obmedzená, čo vyplýva z výskumu [6]. Taktiež rozšírenie takýchto aplikácií brzdia náklady na hardvér, ktorý je potrebný na ich spustenie, ako sú napríklad virtuálne okuliare (aj keď sa ich cena stále znižuje). S tým súvisí aj slabá pripravenosť pedagógov a žiakov, ktorý nemajú dostatočné skúsenosti s používaním virtuálnej reality vo výučbe. Ako riešenie týchto problémov vznikol na školiacom pracovisku Technickej Univerzity v Košiciach v roku 2021 systém virtuálnych výučbových prostredí nazvaný LirkisEduVePn, kde konkrétne výučbové scenáre sú dané procesnými grafmi v podobe Petriho siete. Cieľom tejto práce je preto rozšíriť tento existujúci systém a potrebné nástroje na jednoduchší vývoj aj zložitejších aplikácií pre výučbu školských predmetov vo virtuálnej realite, pomocou vytvorenia ukážkovej scény. V tejto miestnosti sú využité všetky nové komponenty a slúži, ako praktická ukážka ich využitia. Rozšírenia opísané v tejto záverečnej práci sa týkajú širokého spektra vylepšení v rámci jednoduchšieho vývoja nových scén a komponentov. Taktiež text tejto záverečnej práce má slúžiť, ako návod pre budúcich

vývojárov ako vytvárať nové a použiť existujúce komponenty a scény a spolu s implementáciou ukážkovej scény má prispieť k lepšej miere pochopenia procesu vývoja, ako aj jednoduchšie vytváranie nových scenárov. Toto riešenie nadväzuje na záverečné práce Adama Kašelu [7], ktorý implementoval prvú verziu systému, Dmytra Damienka [8] a Lukáš Pisarčíka [9], ktorý pridali rôzne rozšírenia týkajúce sa používateľského rozhrania, zbierania údajov o priebehu scenárov, ale aj rôzne rozšírenia existujúcich scén. Ďalej práca nadväzuje na Mariána Cibulú [10], ktorý pridal do projektu nový scenár a komponenty a Juraja Rudyho [11], ktorý urobil rôzne úpravy vo všetkých aspektoch aplikácie.

Formulácia úlohy

V analýze tejto záverečnej práce je prostredníctvom systematického prehľadu vedeckých článkov (kapitoly 3, 3.3) nastolená problematika nedostatočnej dostupnosti vhodných nástrojov a znalostí pre tvorbu interaktívnych edukačných hier vo virtuálnej realite, čo obmedzuje rýchly a efektívny vývoj nových scenárov a aplikácii. Časť analýzy sa venuje aj zhodnoteniu najnovšej verzie projektu (kapitola 2), ako aj záverečných prác študentov, ktorý na projekte pracovali predtým (kapitola 2.5). Kapitola 2 sa venuje podrobnej analýze súčasného stavu riešenia, kde je rozobratá architektúra aplikácie a sú identifikované hlavné obmedzenia pri tvorbe nových scén. V tejto časti sú predstavené použité technológie - od backendovej časti a rámca Angular pre frontend až po A-Frame, ktorý umožňuje tvorbu 3D prostredí. Veľký dôraz je kladený na vývoj nových komponentov, ktoré boli implementované pomocou moderných knižníc ako Ammo.js a riadené pomocou Petriho sietí. Odporúčania uvedené v záveroch týchto záverečných prác pomohli k lepšej formulácii návrhovej časti práce.

Návrh riešenia záverečnej práce sa dá rozdeliť na dve časti: oprava a nahradenie existujúcich komponentov za nové, prípadne vylepšené (kapitola 8) a návrh ukážkovej scény spolu s predstavením nových, univerzálnych komponentov, pripravených na použitie v nových scenároch (kapitoly 6, 7, 8.2). Výsledkom tejto práce je teda nová verzia projektu, ktorá obsahuje nové komponenty a scény, ale aj návod, ako ich používať a vytvárať nové.

V rámci vyhodnotenia riešenia (kapitola 9) bola skúmaná efektivita hlavne písomnej časti tutoriálu pre vývojárov, ktorý sa so systémom zoznamujú. Získané dáta poukazujú na silné stránky navrhnutých rozšírení, ale zároveň identifikujú

oblasti, kde je ešte priestor na optimalizáciu. Výsledky experimentov a prieskumov tak poskytujú cenné poznatky pre ďalší vývoj a integráciu systémov virtuálnej reality vo výučbe.

V závere práce je nakoniec zhodnotené riešenie a jeho efektivita a použiteľnosť v praxi na základe výsledkov prieskumu.

1 Edukačné hry vo virtuálnej realite

Spojenie virtuálnej reality a interaktívnej výučby je predmetom rôznych výskumov už viac ako 30 rokov, čiže dlho pred tým, ako zariadenia umožňujúce vstúpiť do virtuálnej reality boli širokej verejnosti dostupné a cenovo prijateľné [1],[2]. Zo štúdie [1] vyplýva, že vhodným použitím správneho používateľského rozhrania a vlastností virtuálnej reality je možné dosiahnuť efektívnu výučbu študentov. V článku [2] je navrhnutý univerzálny model rámca pre tvorbu prostredí vo virtuálnej realite pre ľubovoľnú edukačnú tému. Tento potenciál virtuálnej reality ako nástroja, či pomôcky na zefektívnenie výučby potvrdzuje aj výskum [3]. V ňom sa uvádza, že virtuálna realita v súčasnom podaní ponúka vysokú mieru interaktivity a zábavy, ktoré sú v dnešnej dobe v rámci zábavno-poučného obsahu bežné, ba až nevyhnutné. Najväčší problém predstavuje malá dostupnosť vhodných implementácií, či rámcov na návrh takýchto učebných pomôcok. Tieto tvrdenia potvrdzuje aj množstvo výskumov, ktoré sa tejto téme už venovali [4]. Pre základný prístup do virtuálnej reality postačujú v dnešnej dobe inteligentné telefóny, alebo pomerne lacné okuliare, ktoré nevyžadujú pripojenie ku externému počítaču. Tieto predpoklady zabezpečujú jednoduchší prístup ku edukačným hrám, ktoré sa odohrávajú vo virtuálnej realite, ako tomu bolo v minulosti [12]. Po pedagogickej stránke výučba vo virtuálnej realite prináša lepšiu mieru zapojenia študentov do procesu výučby, lepšiu mieru samovzdelávania, ale aj efektívnejší proces nadobudnutia vedomostí vďaka zapojeniu viacerých zmyslov v rámci interakcií s virtuálnym prostredím [13]. Virtuálna realita postavená na interakciách a zapojení používateľa do deja a prostredia ponúka vysoký vzdelávací potenciál tým, že robí proces učenia pútavejší a zábavnejší [14]. Ďalším dôležitým aspektom pri vytváraní edukačných hier vo VR je dobrý príbeh. Vďaka nemu si študenti dokážu lepšie zapamätať danú tému. Príbeh ďalej transformuje za bežných okolností nezáživný obsah na pútavý a môže prispieť ku väčšej miere zapamätateľnosti danej školskej látky [15]. Ďalším motivátorom pre študentov je miera zábavy a pozitívneho zážitku z edukačnej hry. Toto tvrdenie podporuje aj štúdia [16], kde boli tieto vplyvy skúmané efektívnosťou učenia sa u dospelých jedin-

cov. Závěry z tejto štúdie naznačujú, že z rastúcou mierou zábavy a pozitívneho zážitku z hrania edukačnej hry rastie podiel naučených vedomostí z učebného materiálu. V skratke, čím viac zábavy študent zažíva pri hraní seriózných hier, tým viac sa naučí. Ak takýto vzťah je potvrdený u dospelých ľudí, existuje vysoká šanca, že to bude fungovať aspoň v malej miere aj na základníckych deti. Efektivita seriózných hier je porovnateľná, ba až vyššia, ako pri tradičných hrách, čo podporuje aj výskum [17]. Mimo základných výhod učenia sa prostredníctvom seriózných hier, ako sú zvýšená motivácia a angažovanosť vzdelávať sa v rámci danej tematiky, je zaujímavou výhodou aj možnosť robiť chyby bez reálnych následkov [18]. Totiž učenie sa na chybách je jedným z najdôležitejších aspektov vzdelávania sa a vedie ku lepšiemu zapamätaniu študovanej látky. Čo sa týka konkrétnych výskumov, ktoré by preverovali edukačné hry pre deti v základných školách s vysokou mierou interaktivity, momentálne ich existuje len veľmi málo. Aj napriek vysokej miere preskúmania virtuálnej reality ako takej, alebo použitia seriózných hier v inej výučbe, ako základníckej, existuje niekoľko dôvodov, ktoré môžu napovedať, prečo tieto hry nie sú stále masovo využívané a z toho vyplývajúci nedostatok výskumov na vplyv týchto hier na edukáciu detí. Ku najčastejším dôvodom podľa všetkých doteraz spomínaných výskumov patrí:

- **Réžia** - Použitie nástrojov virtuálnej reality si vyžaduje značnú prípravu, či už prostredia, softvéru, kúpa hardvéru, alebo zaškolenie žiakov aj učiteľov.
- **Náklady** - Aj keď sa v dnešnej dobe VR okuliare dajú kúpiť relatívne lacno, náklady na podporu, údržbu a zaškolenie pedagógov ostávajú stále veľkým problémom.
- **Zaškolenie** - S malou popularitou virtuálnej reality prichádza aj fakt, že učitelia a študenti nie sú s touto technológiou oboznámení a potrebujú sa naučiť, ako ju používať. To môže spôsobovať odvrátenie pozornosti od pôvodného cieľu seriózných hier vo virtuálnej realite, čiže vzdelávanie sa v rámci učebných osnov.

Medzi menej časté dôvody ďalej patrili nedostatok realizmu a malá miera interakcie. Keďže prvé tri problémy s použitím virtuálnej reality v rámci výučby sú spojené najmä s jej popularitou a navyše problém réžie rieši už sčasti rámec *LirkisEduVePn*, na ktorý táto záverečná práca nadväzuje, zvyšok analýzy sa práve zameriava na posledné dva dôvody, a to zvýšenie interaktivity a autentickejší zážitok z virtuálneho prostredia.

2 Analýza aktuálneho riešenia

2.1 Ciele analytickej časti

V rámci tejto záverečnej práce sa dajú definovať dve hlavné oblasti otázok, ktoré je potrebné preskúmať. Prvá oblasť je zameraná na integrovanie väčšej miery interaktivity do existujúceho riešenia. Druhá oblasť sa týka zlepšenia používateľského zážitku pre vývojárov nových virtuálnych prostredí, hlavne pri ich prvom kontakte s riešením. Z týchto dvoch oblastí teda vyplývajú základné otázky zobrazované v tabuľke 2.1.

Otázka	Motivácia
O1: Na základe doposiaľ vykonaného vývoji rôznych komponentov existujúceho projektu, ktoré časti boli najviac problematické pri vývoji týchto komponentov ?	Motivácia tejto otázky je identifikovať problematické oblasti pri vývoji nových virtuálnych scén do aplikácie.
O2: Aké druhy interaktívnych prvkov virtuálnych prostredí (zahrnujúc klasické, aj VR video hry) sú vhodné pre výučbu detí na základných školách ?	Motiváciou tejto otázky je zistiť na základe predošlých výskumov, ktoré interaktívne prvky sú pre cieľovú skupinu najzábavnejšie / najúčinnnejšie pre efektívnu výučbu.

Tabuľka 2.1: Výskumné otázky a ich motivácia

Nasledujúca časť sa zaoberá analýzou aktuálneho riešenia, pričom závery odvodené z nej poskytujú odpoveď na prvú formulovanú otázku.

2.2 Opis aplikácie

LirkisEduVePn je softvérový systém virtuálnych výučbových 3D prostredí, ktorý umožňuje vytváranie a plnenie interaktívnych úloh. Cieľ tejto aplikácie je tvorba

výučbových prostredí vo forme takzvaných „Seriózných hier“, primárne zameraných na učivo základnej školy. Na to, aby žiaci úspešne vyriešili úlohy v takomto prostredí, potrebujú mať znalosti o danej školskej látke. Priebeh týchto úloh je riadený Petriho sieťami. Aplikácia beží v prehliadači a je možné ju používať aj za pomoci okuliarov na virtuálnu realitu. Samotná aplikácia je rozdelená nasledovne: backend, ktorý je napísaný v Jave sa stará o rôzne služby, ako sú napríklad registrovanie a uloženie používateľa, uloženie a načítanie scény a scenárov, priradovanie úloh používateľom, alebo rozdeľovanie používateľov do skupín. Frontend je napísaný v Javascripte a Typescripte pomocou rámca Angular a stará sa o zobrazovanie stránky, chod virtuálnych prostredí a posielanie rôznych požiadaviek na backend. Samotné výučbové prostredia sú tvorené v rámci A-Fame, ktorý slúži na vytváranie 3D a AR/VR prostredí prostredníctvom knižnice Three.js a jazyka HTML. Riadenie tohto prostredia je realizované pomocou Petriho sietí, ktoré poskytujú jednoduchú tvorbu interaktívnych scenárov s viacerými spôsobmi prechodu. Aplikácia taktiež umožňuje sledovanie plnenia úloh z role učiteľa, kde je možné nájsť základné štatistiky o poslednom pokuse vyriešenia jednotlivých úloh študentmi. Učiteľ taktiež môže do aplikácie vkladať nové scenáre, či prostredia a vytvárať úlohy, pre ktoré potom priradí skupiny žiakov, ktoré tieto úlohy majú vyriešiť.

2.3 Aktuálny stav aplikácie

Po prvom otvorení aplikácie v prehliadači je nutné sa prihlásiť, poprípade sa zaregistrovať. Pri registrácii príde používateľovi potvrdzujúci email. Následne, ak je používateľ v roli študenta, môže si spustiť a riešiť jednotlivé interaktívne úlohy. Študent si taktiež môže pozrieť údaje o poslednom pokuse riešenia úlohy. Priebeh úlohy je možné si uložiť a pokračovať v nej neskôr, ako aj predčasne úlohu ukončiť. V roli učiteľ je možné okrem spomenutých akcií vytvárať úlohy pre žiakov vložením scény, scenára a priradením úlohy žiakom rozdelených do skupín. Ďalej si učiteľ môže pozrieť priebeh riešených úloh a rôzne štatistiky o úspešnosti žiaka na konkrétnej úlohe. Samotné scenáre sú tvorené .cpn súborom s Petriho sieťou vytvorené v nástroji CPN Tools a .json súbormi pre text ku popisom úloh a objektov na scéne. Samotná scéna je tvorená pomocou rámca A-Frame, jazyka HTML a JavaScript-u, pre ktorú je ďalej potrebné naprogramovať jednotlivé komponenty a umiestniť do nej objekty a entity. Modely scény sú vytvárané v ľubovoľnom grafickom editore, ako je napríklad Blender.

2.4 Limitácie riešenia

Za najväčšie obmedzenie aplikácie sa dá považovať proces vytvárania novej scény a jej integrácia do architektúry riešenia. Momentálne, ak sa vytvára nová scéna pre konkrétny scenár reprezentovaný v Petriho sieti, je na autorovi scenára, ako bude udalosti a prechody v scenári definovať, bez existencie typizovaných tried, alebo metód. Vo výsledku je potom súbor so skriptom pre scénu veľmi dlhý a neprehľadný. Taktiež nie je na prvý pohľad zrejmá forma týchto prechodov, čiže aké parametre majú obsahovať a aké metódy majú implementovať. Čo sa týka metód, ktoré sa priamo netýkajú prechodov, ako je napríklad prehranie zvuku pri úspešnom dokončení úlohy, zviditeľnenie informačného panelu, alebo hocijako inak vizuálne, alebo zvukovo aktualizovať scénu, je nutné takéto metódy definovať pre každú scénu, aj keď sa často opakujú. Ďalšie obmedzenie je nedostatok typizovaných udalostí, ktoré riadia prechody prostredím. V aktuálnej verzii riešenia sú definované iba udalosti reagujúce na kliknutie myšou, alebo presun predmetu na určené miesto. To značne limituje rôznorodosť úloh, ktoré je možné pre aplikáciu vytvoriť. V neposlednom rade, samotný proces vytvárania vlastných úloh nie je nijako zdokumentovaný, čiže je nutné vynaložiť veľké úsilie na pochopenie architektúry komponentov a procesu vytvárania scén, aby bolo vôbec možné začať takéto scény pre úlohy vytvárať. Všetky tieto limitácie teda znamenajú, že proces vytvárania nových scén a úloh je náročný, keďže pre vytvorenie nových typov scén je nutné vytvárať vlastné komponenty, či udalosti a zapisovať skripty zdĺhavo a bez pomoci nejakého rozhrania, alebo knižnice.

2.5 Analýza najnovších prác na aktuálnom riešení

V tejto časti sa nachádza opis posledných dvoch prác, ktoré nejakým spôsobom rozširovali existujúce riešenie, či už pridaním nového scenára a scény, alebo rôznymi úpravami webovej stránky a služieb. Postrehy získané z týchto prác spolu s predošlou časťou 2 poskytujú kompletnú odpoveď na prvú otázku analýzy.

2.5.1 Scenárom riadené virtuálne výučbové prostredie pre geografiu

Táto bakalárska práca, ktorej autorom je kolega Bc. Marián Cibula [10], pridáva do virtuálneho prostredia novú scénu v rámci predmetu Geografia na tému Národné parky na Slovensku. Scenár pre toto virtuálne prostredie bol, podobne

ako iné, vytvorený pomocou Petriho sietí, ktoré riadia interaktívne prvky scény. Ďalej 3D prostredie a modely boli vytvorené v aplikácii Blender a importované do nástroja A-Frame. Hlavným cieľom scenára bolo umiestniť modely zvierat na správne miesta, načo bol vytvorený aj špeciálny komponent. V závere tejto práce sa ako odporúčania na ďalšie vylepšenia, alebo rozšírenia aplikácie odporúča pridať viac scén pre nové školské predmety, rozšíriť možnosti interakcie s prostredím, pridať prvky gamifikácie a nakoniec pridať možnosť úpravy vzhľadu scén používateľmi.

2.5.2 Systém virtuálnych výučbových prostredí riadených procesnými grafmi

Cieľom tejto diplomovej práce napísanej autorom Ing. Juraj Rudy [11] boli rôzne úpravy väčšiny častí celého projektu. Tieto vylepšenia sa týkali hlavne úpravy vzhľadu webu, odstránenia chýb v scénach, nasadenie projektu do TUKE cloudu, ale aj zavedenie rôznych HTTPS bezpečnostných protokolov. Do projektu boli taktiež pridané aj nové funkcionality, ako sú ovládanie mobilnými zariadeniami a implementácia virtuálnych rúk, ktoré umožňujú lepšiu manipuláciu s virtuálnym prostredím. V rámci praktickej časti tejto záverečnej práce bol uskutočnený aj prieskum miery pochopenia Petriho sietí ľuďmi s IKT vzdelaním a ľuďmi bez IKT vzdelania. Výsledky tohto prieskumu priniesli očakávané výsledky, a to, že respondenti s IKT vzdelaním, ktorí sa predtým nestretli s Petriho sieťami boli schopní lepšie túto problematiku pochopiť, ako tí bez IKT vzdelania. Podobné výsledky vykazujú aj ďalšie prieskumy bližšie popísané v nasledujúcich kapitolách. Podobne ako v predošlej spomínanej záverečnej práci, aj v tejto sú uvedené rôzne možné rozšírenia, alebo vylepšenia projektu. Medzi už spomínané patrí pridanie nových scén a lepšie možnosti prispôsobenia si scény. Ako ďalšie odporúčania boli uvedené: úprava kódu do čitateľnejšej a efektívnejšej podoby, umožniť riešenie scény viacerými hráčmi, rozšíriť vykonaný prieskum, alebo vytvorenie „čakacej miestnosti“, kde by si hráči vedeli vyberať scény priamo vo virtuálnom prostredí.

3 Systematická analýza a prehľad podobných riešení

V tejto kapitole je opísaný systematický prehľad vedeckých článkov a literatúry, ktoré sa zaoberajú Petriho sieťami, serióznymi hrami a virtuálnou realitou. Keďže cieľom tejto práce je hlavne rozšíriť možnosti interaktivity, ktoré aplikácia ponúka a celkovo vylepšiť spôsob vytvárania nových interaktívnych prostredí, do prehľadu literatúry sú zahrnuté aj výskumy, ktoré sa zaoberajú rôznymi spôsobmi, ako používatelia (hlavne deti) vnímajú takéto virtuálne prostredia a ako zlepšiť ich používateľský zážitok a mieru zábavy. Táto kapitola zároveň prináša odpoveď na druhú otázku analýzy uvedenú v časti 2.1, ktorej znenie bolo nasledovné:

Aké druhy interaktívnych prvkov virtuálnych prostredí (zahrnujúc klasické, aj VR video hry) sú vhodné pre výučbu detí na základných školách ?

3.1 Metodika získavania dát a ich spracovanie

Potom, ako bola sformulovaná druhá výskumná otázka je potrebné definovať, akým spôsobom sa budú získavať dáta pre skúmanú oblasť. Pre druhú výskumnú otázku sa ako zdroj relevantných štúdií, použili nasledujúce databázy:

- **Scopus**¹ - Scopus je rozsiahla multidisciplinárna databáza, ktorá spája bibliografické a scientometrické údaje a radí sa medzi svetového lídra v tejto oblasti.
- **Google Scholar**² - V súčasnosti najrozšírenejší nástroj na vyhľadávanie vedeckých prác od spoločnosti Google. Jednoducho sa používa a vďaka technológiám, ktoré Google používa aj pre svoj webový prehliadač je účinnosť nájdenia toho správneho zdroja veľmi vysoká. Prehľadávanie v tejto data-

¹<https://www.scopus.com/>

²<https://scholar.google.com/>

báze síce nepodporuje priamo použitie zložitejších dopytov, ale nástroj “pokročilé hľadanie,, ponúka dostatočnú mieru filtrovania hľadaných dát.

- **Web of Science³** - Web of Science Core Collection zahŕňa články z odborných časopisov, knižné kapitoly a konferenčné príspevky z rôznych vedeckých odborov.

Následne bolo nad každou databázou spustené vyhľadávanie pomocou dopytov zobrazených v tabuľke 3.1. Vďaka použitiu takýchto dopytov bol proces získavania všetkých relevantných dát rýchly a efektívny.

Databáza	Dopyt
Scopus	TITLE-ABS-KEY (education AND ('Virtual reality' OR 'video game' OR 'serious game' OR 'augmented reality') AND (interaction OR interactivity OR immersion OR 'game design' OR interactive OR engagement)) AND ALL ((webxr OR aframe) AND children AND (school OR learning OR learn)) AND PUBYEAR > 2010
Google Scholar	WebXR study learn school educational virtual reality serious game interact children interactive OR interaction OR immersive OR interactivity 'game design' since:2010
Web of Science	TS=(education AND ('Virtual reality' OR 'video game' OR 'serious game' OR 'augmented reality') AND (interaction OR interactivity OR immersion) and ('game design' OR interactive OR engagement) AND children AND (primary school OR learning OR learn or education) and (implementation or framework) not (disability or autism)) AND PY=(2010-2024)

Tabuľka 3.1: Dopyty použité na prehľadávanie databáz

Keďže tento výskum je pokračovaním niekoľkých ďalších záverečných prác, ktoré rozširovali *LirkisEduVePn* projekt, neobsahujú dopyty na databázu úplne všetky kľúčové slová, ktoré boli v minulých prácach použité. Na druhú stranu dopyty boli doplnené o kľúčové slová relevantné pre tému tejto záverečnej práce, ktoré sa v iných prácach nevyskytovali. V prípade Web of Science bolo nutné explicitne v dopyte vynechať všetky výsledky, ktoré sa zaoberali edukačnými hrami

³<https://www.webofscience.com>

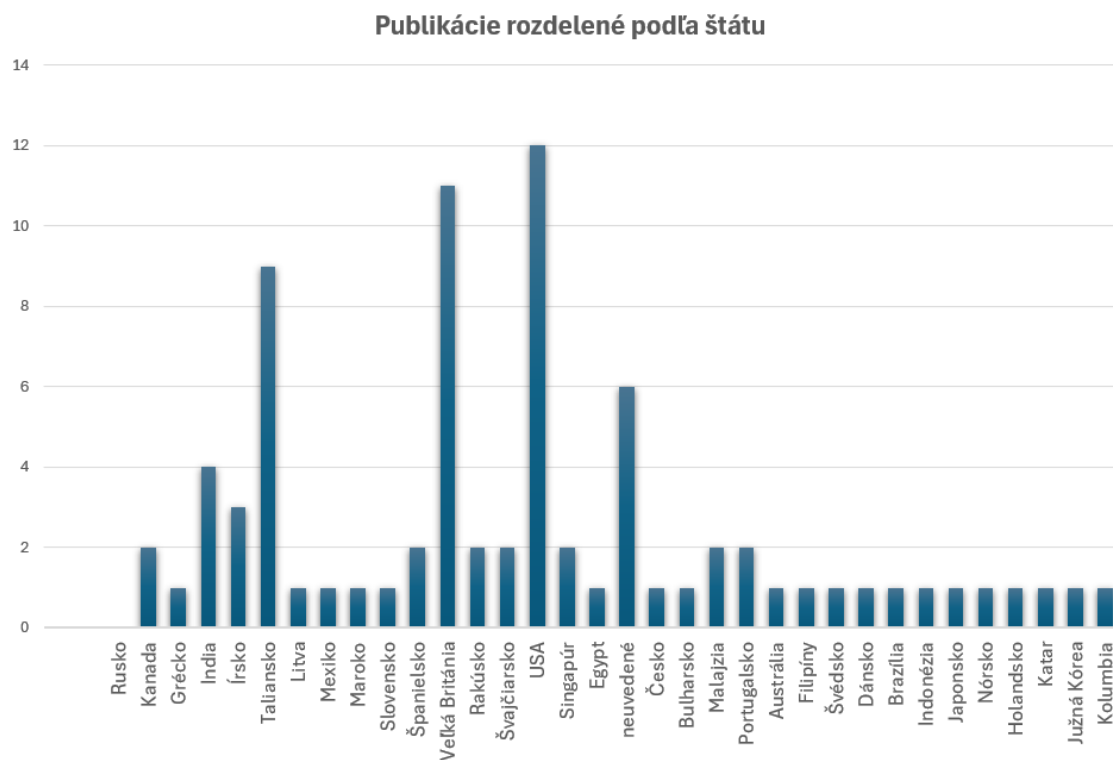
pre zdravotne znevýhodnených, keďže to nie je cieľom tejto záverečnej práce. Po vykonaní dopytov nad všetkými databázami boli unikátne dáta zozbierané a výsledky zanalyzované, čo je opísané v nasledujúcej kapitole.

3.2 Analýza získaných výsledkov

Obsahom tejto kapitoly je systematické zhodnotenie nazbieraných výsledkov získaných z vedeckých databáz. Zo všetkých troch databáz bolo nájdených spolu 79 výskumov, ktoré spĺňali podmienky vo vyhľadávacích dopytoch. Na grafe 3.2 je možné vidieť, z ktorých krajín publikácie pochádzali a na Obr. 3.1 v ktorom roku boli vydané. Keďže virtuálna realita začala byť populárna a dostupnejšia širokej verejnosti až v posledných približne 15-ich rokoch, rozsah dátumu vydania pre hľadané publikácie sa nastavil od roku 2010. Tomu nasvedčuje aj graf 3.1, kde väčšina výskumov boli publikované v období od roku 2020. Rastúcu popularitu hier vo virtuálnej realite a serióznych hier je možné vidieť aj na Obr. 3.2, ktorý zobrazuje veľké množstvo krajín, ktoré riešia danú problematiku.

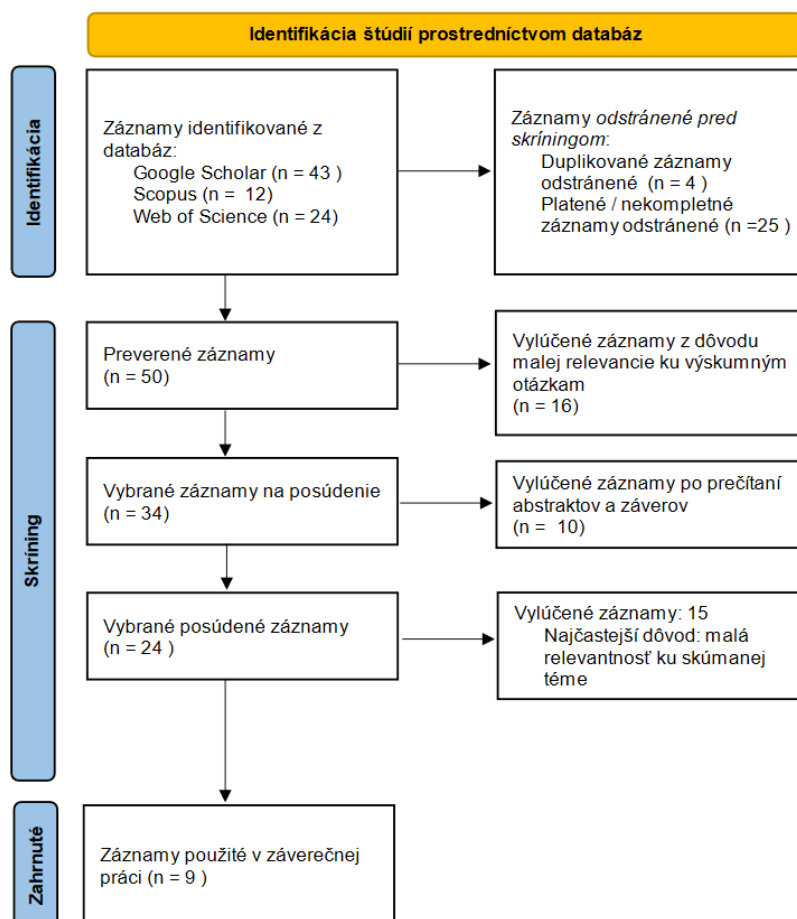


Obr. 3.1: Graf zobrazujúci rok vydania publikácií



Obr. 3.2: Graf zobrazujúci krajinu vydania publikácií

Po nájdení všetkých relevantných zdrojov nastala fáza systematického filtrovania literatúry podľa PRISMA vývojového diagramu, zobrazenom na Obr. 3.3, v ktorom sa nachádzajú jednotlivé kroky analýzy dát spolu s relevantnými údajmi získanými z vedeckých databáz.



Zdroj: Page MJ, McKenzie JE, Bossuyt PM, Boutron I, Hoffmann TC, Mulrow CD, et al. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. BMJ 2021;372:n71. doi: 10.1136/bmj.n71

Obr. 3.3: PRISMA 2020 vývojový diagram pre nové systematické prehľady, ktoré zahŕňujú vyhľadávanie iba v databázach

V prvom kroku sa odstránili všetky duplicitné záznamy, ktoré sa objavovali vo viacerých databázach, ale aj nekompletné, alebo neprístupné záznamy. V druhom kroku systematickej analýzy boli odstránené záznamy, ktoré na prvý pohľad nevyhovovali prvej výskumnej otázke. Boli to napríklad výskumy týkajúce sa vytvárania edukačného materiálu pomocou umelej inteligencie, zmiešanej reality, alebo určenej pre zdravotne znevýhodnených. V treťom kroku sa po prečítaní abstraktov a záverov zvyšných 34-och záznamov vylúčili ďalej tie, ktoré neboli vhodné ako zdroj výskumu. Boli to napríklad záznamy, ktoré sa nedostatočne

venovali interakcií vo virtuálnej realite, alebo boli určené pre starších žiakov a profesionálov. V poslednom kroku sa 24 publikácii skúmalo hlbkovo prečítaním celého ich textu. V tomto kroku sa vybrali finálne práce, ktoré budú podrobne rozobraté v nasledujúcich kapitolách a ktoré boli použité na spätné prehľadávanie zdrojov, ktoré sa nachádzali v bibliografii týchto záznamov a na základe ktorých bola vytvorená prvá kapitola analýzy.

3.3 Analýza podobných riešení

Všetky práce v tejto kapitole predstavovali najrelevantnejšie závery a poznatky, ktoré majú potenciál pozitívne ovplyvniť, alebo inšpirovať syntézu riešenia tejto záverečnej práce. Hlavnou témou vybraných vedeckých článkov, ako už vyplýva z druhej výskumnej otázky, je interaktivita vo virtuálnej realite a jej efektívne využitie v edukačných hrách pre študentov základných škôl.

3.3.1 A Review on Augmented Reality Authoring Toolkits for Education

Podľa analýzy publikovanej v článku [6] je momentálne na trhu nedostatok nástrojov na rýchle a jednoduché vytváranie virtuálneho obsahu, čo je jedným zo základných predpokladov na rozšírenie používania VR, alebo AR edukačných hier na základných školách. Ďalej úzka spolupráca medzi tvorcami edukačných hier a ich používateľmi, či učiteľmi je veľmi dôležitá pre tvorbu úspešnej a použiteľnej hry. Vytváranie týchto hier však prináša 3 výzvy: ovládateľnosť gestami, správna simulácia kamery (dôležitá na minimalizovanie kinetózy) a v neposlednom rade vhodný proces návrhu samotnej hry pomocou návrhových diagramov. Tento výskum ďalej zdôrazňuje nutnosť vytvárať rámce na produkciu VR hier dostupné aj pre učiteľov a osoby bez technických a programátorských znalostí a na možnosť vytvárať interaktívny obsah. Tieto aspekty v existujúcich riešeniach zväčša chýbajú. Čo sa týka analýzy konkrétnych nástrojov na vytváranie VR hier a ich poskytované formy interakcie, za zmienku stojí nástroj **BlippAR's toolkit Blipp-builder**[19]. Tento nástroj umožňuje používateľom pridávať napríklad YouTube videá do virtuálneho prostredia a otvárať internetové odkazy na základe interakcie. Okrem toho je do scény možné pridávať nahrávky, ktoré slúžia ako inštrukcie pre študentov v rámci postupu hrou.

3.3.2 Virtual reality and augmented reality in social learning spaces a literature review

Autori článku [20] sa zamerali na prehľad literatúry, ktorá skúma použitie VR a AR v edukačných priestoroch, ako sú školy, múzeá, ale aj rôzne ďalšie neformálne edukačné priestory. Z analýzy tohto výskumu vyplývajú dva spôsoby, ako kategorizovať interakciu s virtuálnym prostredím.

- **Bowman a Hodge**[21] - rozdeľujú interakcie vo virtuálnom prostredí do troch skupín: ovládanie zorného poľa, čiže navigácia, výber a manipulácia
- **LaViola a kol.**[22] - rozdeľujú interakciu podľa jej spôsobu vykonania na: uchopenie (napríklad pomocou virtuálnej ruky), ukazovanie (napríklad pomocou smerového lúča), použitie ovládacích prvkov na ovládači, nepriame (napríklad výber pomocou lúča a následná manipulácia objektu), obojručná a nakoniec hybridná (kombinácia predošlých spôsobov podľa potreby)

Narozdiel od kategorizácie spôsobov interakcie s virtuálnym prostredím, ktorá je pomerne priamočiara, odpoveď na otázku: „Čo je najlepšia stratégia pri vytváraní edukačných VR hier?“ je často konfliktná a v podstate z každého výskumu vychádzajú iné výsledky. Dôvodom tohto nesúladu môžu byť aj nedostatok štandardizácie v tejto oblasti, alebo málo snahy reprodukovať výsledky predošlých štúdií.

3.3.3 Gamified Evaluation in STEAM for Higher Education: A Case Study

V článku [23] zameranom na hodnotenie techník gamifikácie predmetov základnej školy vyplynulo, podobne ako v mnohých ďalších výskumoch, že tieto techniky napomáhajú ku lepšiemu pochopeniu školskej látky. Ku zaujímavým technikám, ako transformovať látku na hru patria napríklad systém odmiern a pedagogický dizajn. Systém odmiern, ako napríklad odznaky, body, ale aj odmeny mimo hry, ako sú tradičné známky a pochvaly, môžu pozitívne motivovať žiakov ku dosiahnutiu lepších výsledkov. Pedagogický dizajn hovorí o štyroch vlastnostiach, ktoré by kvalitná seriózna hra mala obsahovať:

- hra by mala obsahovať dostatok nápodvedí a pomôcok pre študentov, ale zároveň im nechať aj priestor na samostatné riešenie úloh, čo podporuje schopnosť rozhodovať sa a riešiť problémy rôznymi spôsobmi

- hra by mala byť generovaná z určitou mierou náhodnosti pre dosiahnutie individuálneho a jedinečného zážitku pre každého študenta, ale mala by tiež zachovať jasnú a konzistentnú štruktúru a úroveň obtiažnosti
- hra má poskytovať údaje o postupu a výsledkov študentov, koľko času v hre strávili, a ktoré sekcie im robili najväčšie problémy
- hra by mala byť použiteľná na hodnotenie študentov a ich nadobudnutých vedomostí v rámci vyučovanej témy

3.3.4 Virtual Worlds for Learning in Metaverse: A Narrative Review

Vo vedeckej práci [5] zameranej na recenziu a rolu príbehu v edukačných hrách sú hry rozdelené do niekoľkých kategórii: Dobrodružné svety, Simulačné svety, Kreatívne svety, Rolová (RPG) hra a Korporatívne svety. Každý z týchto svetov ponúka svoje výhody a má svoje uplatnenia v rôznych odvetviach výučby. Pre základškolské vzdelávanie a s pohľadu najväčšej šance preraziť na trhu sú najvhodnejšie žánre Dobrodružné a Rolové hry. Dali by sa tam zaradiť aj Kreatívne hry, ale tých je momentálne na trhu dostatok a konkurencia, ako napríklad Minecraft, Roblox a Lego hry ponúkajú prakticky všetko, čo by takáto hra mala obsahovať. Na druhú stranu, simulačné hry sú skôr vhodné do vyššieho vzdelávania, kde presnosť a vysoká podobnosť s realitou je vysoko žiadaná a na rôzne prehliadky, ako sú napríklad múzeá a galérie. Taktiež kolaboračné hry majú množstvo výhod a sú veľmi dôležité pre budovanie sociálnych a komunikačných zručností detí, ich tvorba nie je cieľom tejto záverečnej práce, a preto z porovnávania budú vynechané. V kontexte učenia študentov schopností a vedomostí relevantných pre život v 21. storočí ponúkajú Dobrodružné a Rolové a hry nasledujúce výhody:

- **Dobrodružné hry** - podporujú kritické myslenie prostredníctvom riešenia úloh, podporujú lepšie pochopenie výrazov a príbehu a stimulujú lepšiu predstavivosť
- **RPG hry** - rozvíjajú schopnosť rozhodovania v komplexných scenároch, umožňujú vývin verbálnej a neverbálnej komunikácie v rámci hraných rolí, podporujú stratégie spolupráce a podporujú kreatívne rozvíjanie a vytváranie rolí

Čo sa týka interakcie, tak priama manipulácia objektov vo virtuálnom priestore a pohyb po nich podporujú priamejší a pohlcujúci zážitok z hry a učenia. Čiže

spojenie príbehu, ktorý vtiahne študentov do deja a zaujme ich so správnou mierou interakcie a hernými mechanikami podľa tohoto výskumu vedie ku lepšej pozornosti študentov, ako tradičné formy výučby.

3.3.5 Education in Collaborative Immersive Virtual Reality

V záverečnej práci [12] sa spomína tzv. „Teória duálneho kódovania“, ktorá hovorí o lepšom učení študentov kombináciou vizuálnych a verbálnych informácií. Ideálne médium na aplikovanie tejto teórie sa považuje práve virtuálna realita. Komplexné prostredie virtuálnej reality môže pomôcť podať látku efektívnejšie, ako tradičné formy výučby, čo vedie ku pútavému viac zapamätateľnému zážitku zo štúdia.

3.3.6 Exploring Mixed Reality Level Design Workflows

Medzi zaujímavé poznatky prezentované v záverečnej práci [24] patria spôsob ako vytvárať používateľské rozhranie vo virtuálnej realite a spôsob interakcie s prostredím. Pri tvorbe používateľských rozhraní je nutné tieto rozhrania vložiť priamo do prostredia a nie „nalepiť na šošovku“ ako tomu je v tradičných videohrách. Dôvodom je neschopnosť zaostriť na objekty, ktorým chýba hĺbka, čo má za následok nečitateľné rozhranie. Keďže táto práca sa zameriavala na tvorbu rámca pre modelovanie 3D prostredí a hier priamo vo virtuálnej realite, na samotné umiestňovanie objektov do scény bol využitý svetelný lúč, ako určitá forma ukazovadla. Toto riešenie umožňovalo používateľom využiť priestorovú náтуру VR a lepšiu flexibilitu, v porovnaní s priamym ovládaním virtuálnou rukou.

3.3.7 Virtual Reality STEM Education from a Teacher's Perspective

Z pohľadu učiteľov, ktorému sa práca [25] venovala, patrí medzi dôležité súčasti edukačných hier pomôcky a nápovedy, ako už bolo spomínané vo výskume 3.3.1. Konkrétne bola spomenutá pomôcka vo forma audio záznamu. Ďalej učitelia navrhovali pridanie zvukových efektov, ako kulisa ku scénam pre lepšiu a viac pohlucujúci zážitok.

3.3.8 Virtual Museum Gamification for Discovery-Based Online Learning in the Metaverse

Zdigitalizovanie výstavy v múzeu určenej pre študentov základných škôl taktiež prinieslo pozitívne výsledky v oblasti učenia a zapamätania si vedomostí [26]. Žiakom napríklad pri lokalizovaní skrytej miestnosti pomohli zvukové nápovedy oveľa viac, ako nápovedy vyplývajúce z textu. Dosiahnutím rovnováhy medzi obťažnosťou a mierou pomôcok v rámci obsahu vo virtuálnej realite má za následok dlhšie udržanie detskej pozornosti, čo znova vedie ku lepšiemu nadobudnutiu vedomostí.

3.3.9 On the role of interaction mode and story structure in virtual reality serious games

V tejto publikácii [27] sa znova zdôrazňuje dôležitosť nájsť rovnováhu medzi obťažnosťou a mierou nápoved v edukačných hrách. V ideálnom prípade by hra mala vedieť rozpoznať, kedy je vhodná chvíľa žiakom poskytnúť pomôcku a kedy ich nechať riešiť problém samostatne. Ďalej čím hra umožňuje väčšiu mieru rozhodovania a slobody, tým sú študenti viac zainteresovaný a motivovaný hru hrať. Je to ale často krát na úkor získaných vedomostí. Preto sa v edukačných hrách odporúča používať aspoň prvky linearity a štruktúry, pre čo najlepší zážitok z hrania a zároveň vysokú mieru naučenia sa danej látky.

4 Závery vyplývajúce z analýzy

Po vykonaní analýzy aktuálneho riešenia a záverečných prác opisujúcich posledné zmeny v ňom, ako aj systematickej analýzy literatúry z vedeckých databáz, boli získané odpovede na dve výskumné otázky, ktoré boli definované v úvode tejto záverečnej práce. Prvá výskumná otázka týkajúca sa aktuálneho riešenia znela:

O1: Na základe doposiaľ vykonanom vývoji rôznych komponentov existujúceho projektu, ktoré časti boli najviac problematické pri vývoji týchto komponentov ?

Najviac problematické časti vývoja, vyplývajúce z vykonaného výskumu vybraných vedeckých článkov zahŕňali:

- Keďže vytváranie nových scén v aktuálnom riešení komplikuje nedostatok pripravených komponentov, ktoré by tvorili základ pre nové scény, je nutné vytvoriť nové komponenty a univerzálne interakcie, ktoré by umožňovali jednoduché a rýchle vytváranie nových scén, ktorých je aktuálne nedostatok.
- Prílišná voľnosť tvorby scenárov ponechaná na autorov jednotlivých scén spôsobuje problémy pri úpravách a rozširovaní projektu. Preto je nutné zjednotiť štýl písania kódu a zvýšiť jeho čitateľnosť, prípadne ho skrátiť a zefektívniť.
- Fyzikálny rámec CANON.js, ktorý poháňa všetky interakcie vo virtuálnom prostredí, je príliš jednoduchý a zastaraný na vytváranie zložitejších a hlavne zaujímavejších interakcií. Preto je nutné nahradiť tento rámec za moderný a výkonnejší, ktorý by umožňoval vytvárať zložitejšie a pútavejšie interakcie.

Vďaka týmto vylepšeniam by tvorba nových scén, ako aj interakcií mala byť v rámci tohto, ale aj budúcich vývojov rýchlejšia, jednoduchšia a mala by viesť k vytváraniu scén, ktoré spĺňajú väčšinu požiadaviek na interaktívne prostredie pre

edukačné hry.

Systematický přehled literatury vykonaný v druhé části analýzy skúmal vedecké práce zamerané na interakciu vo virtuálnej realite, seriózne hry a využitie VR a AR v edukačných hrách pre základné školy. V rámci tejto analýzy boli preskúmané rôzne výskumy z vedeckých databáz, ako Scopus, Google Scholar a Web of Science. Z týchto výskumov boli vybrané tie najrelevantnejšie, ktoré boli následne podrobne analyzované a získané poznatky boli použité na nájdenie odpovede pre druhú výskumnú otázku:

O2: Aké druhy interaktívnych prvkov virtuálnych prostredí (zahrnujúc klasické, aj VR video hry) sú vhodné pre výučbu detí na základných školách ?

Z analýzy vyplýva, že najvhodnejšie interaktívne prvky pre výučbu detí zahŕňajú:

- Manipuláciu objektov vo virtuálnom prostredí, ako sú uchopenie, presúvanie, hádzanie predmetov, alebo akákoľvek aktívna forma interakcie nad rámec 2D počítačových hier. Tieto prvky podporujú motorické schopnosti a priame zapojenie do virtuálneho prostredia.
- Ukazovanie/ovládanie pomocou smerového lúča, ktoré umožňuje jednoduché a intuitívne interakcie, pričom rešpektuje priestorovú náтуру VR. Je to vhodné najmä pre mladších študentov, ktorí môžu mať problém s jemnou motorikou, ale aj pre jednoduchšiu navigáciu, alebo manipuláciu objektov.
- Kombinácia komplexnej fyzikálnej simulácie správania predmetov a prvkov prostredia (tlačidlá, páčky, hádzanie, kolízie, gravitácia) a interaktívnych herných mechaník (napr. skladačky, časovo obmedzené úlohy, manipulácia s predmetmi) spájajú zábavu so vzdelávaním a rozvíjajú schopnosť riešiť problémy.
- Prispôbené rozhranie: Používateľské rozhranie umiestnené priamo v prostredí namiesto „nalepenia na šošovku“ zvyšuje čitateľnosť a informovanosť hráčov o aktuálnom dianí v scéne.
- Zvýšená miera interaktivity, napríklad pomocou implementovaných Petriho sietí, ponúka určitú mieru „voľnosti“, akým spôsobom hráč scénu dokončí.

Tieto prvky vytvárajú pohlcujúce prostredie, ktoré spája učenie a hru, podporuje schopnosť rozhodovania, riešenia praktických úloh a logického myslenia.

5 Návrh nových riešení

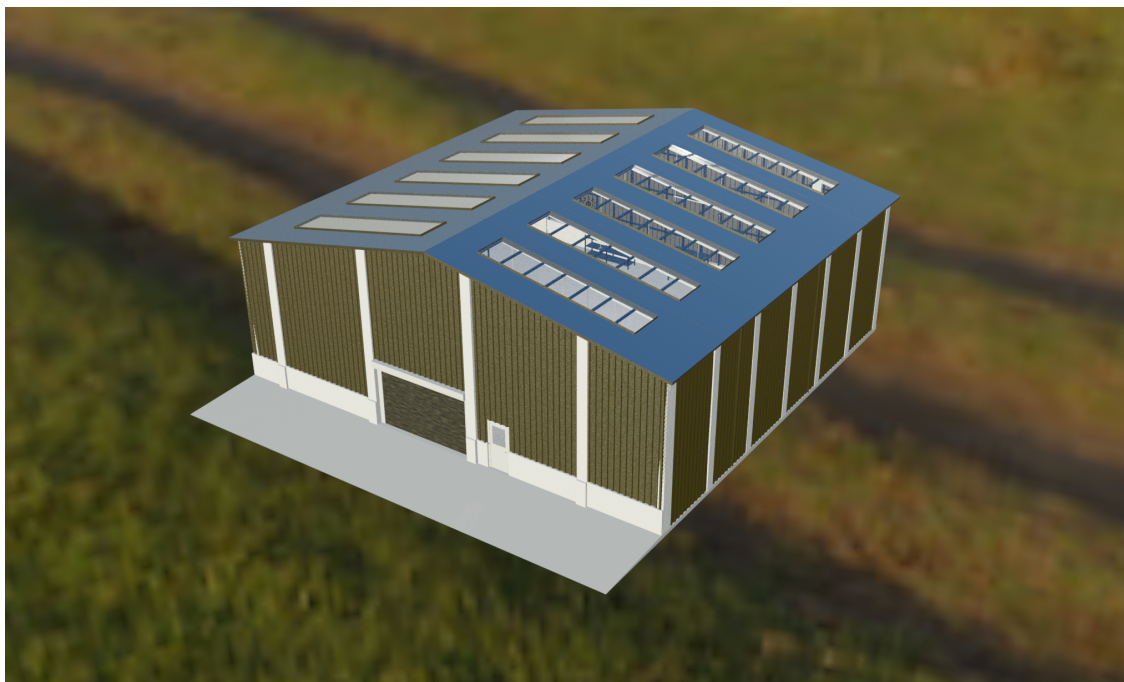
Ako vyplynulo z analýzy aktuálneho stavu projektu, ako aj zo skúmaných vedeckých prác, pre tvorbu dostatočne kvalitných a interaktívnych scén je potrebné implementovať sadu nových komponentov, ktoré by umožnili jednoduchšiu tvorbu nových scén a zároveň by zabezpečili dostatočnú mieru interaktivity s prostredím. Taktiež je potrebné zjednodušiť proces tvorby nových scén a zvýšiť prístupnosť projektu pre širšiu verejnosť. Z tohto dôvodu bola do projektu pridaná nová, takzvaná *Ukážková Scéna*, ktorá v sebe implementuje všetky novo pridané komponenty a zmeny v rámci vytvárania nových virtuálnych scén. Veľká časť týchto komponentov je navrhnutých univerzálne pre budúce použitie pri navrhovaní nových scén, iné slúžia, ako ukážka možností rámcov A-Frame a Three.js.

6 Dizajn ukážkovej Scény

Predtým, než je možné popísať jednotlivé komponenty, ktoré boli implementované v rámci ukážkovej scény, je potrebné popísať proces grafického návrhu scény, ktorá reprezentuje samotnú ukážkovú scénu, ako aj úlohy, ktoré sú v nej obsiahnuté. Tieto úlohy nezahŕňajú žiadnu konkrétnu tému v rámci výučbových osnov, narozdiel od predošlých riešení, ale sú skôr zamerané na demonštráciu možností rámca A-Frame a Three.js, ako aj nových komponentov, ktoré boli implementované.

6.1 Model scény

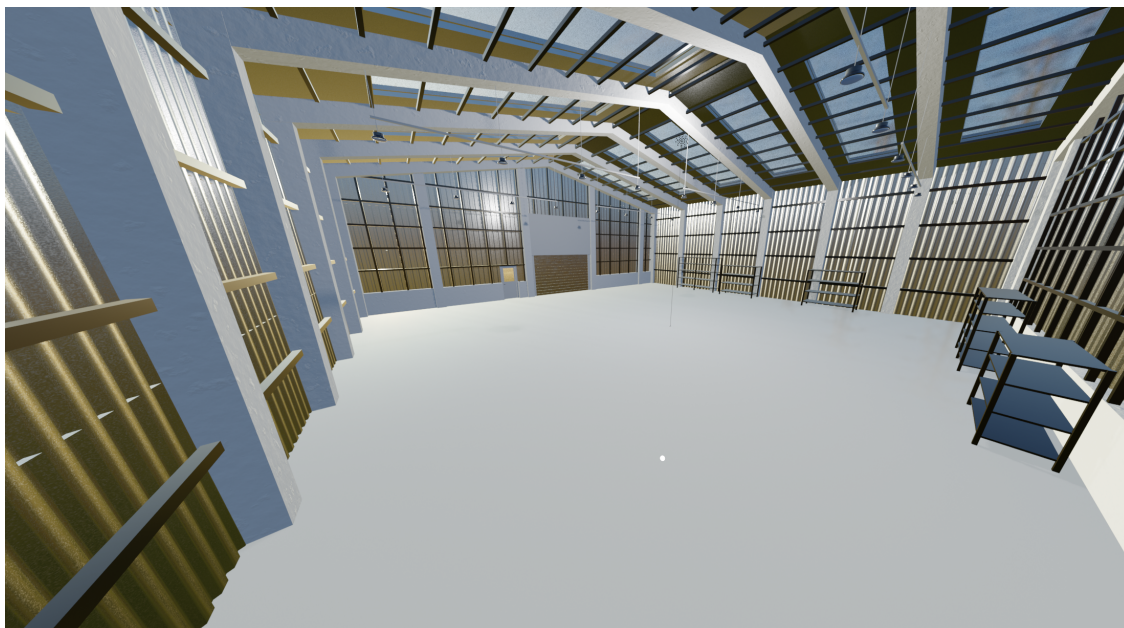
Tak ako predchádzajúce scény vytvorené pre tento projekt, aj na tvorbu ukážkovej scény bol použitý program Blender, ktorý umožňuje jednoduchú tvorbu a úpravu 3D modelov, ktoré je možné spracovať rámcom A-Frame. Ako základ navrhutej scény bol použitý model garáže, ktorý bol značne upravený a rozšírený pre potreby scenárov, ktoré sa v nej odohrávajú. Celá scéna je rozdelená do troch hlavných častí, ktoré sú navzájom oddelené stenami a dverami. Namiesto pôvodných stien boli pridané presklenné steny, pričom scéna bola umiestnená do lesného prostredia, čo navodzuje príjemnú atmosféru pre hráčov. Z hľadiska ušetrenia výkonu boli podobne, ako v predošlých riešeniach, použité voľne dostupné textúry, na ktoré bola aplikovaná technika pečenie osvetlenia, ktorá do textúr „vypáli“ osvetlenie a tieňovanie, čím sa odstráni potreba počítania tieňov a osvetlenia v reálnom čase. Na obrázkoch 6.1, až 6.4 je možné vidieť porovnanie pôvodného modelu garáže a novovytvorenej scény.



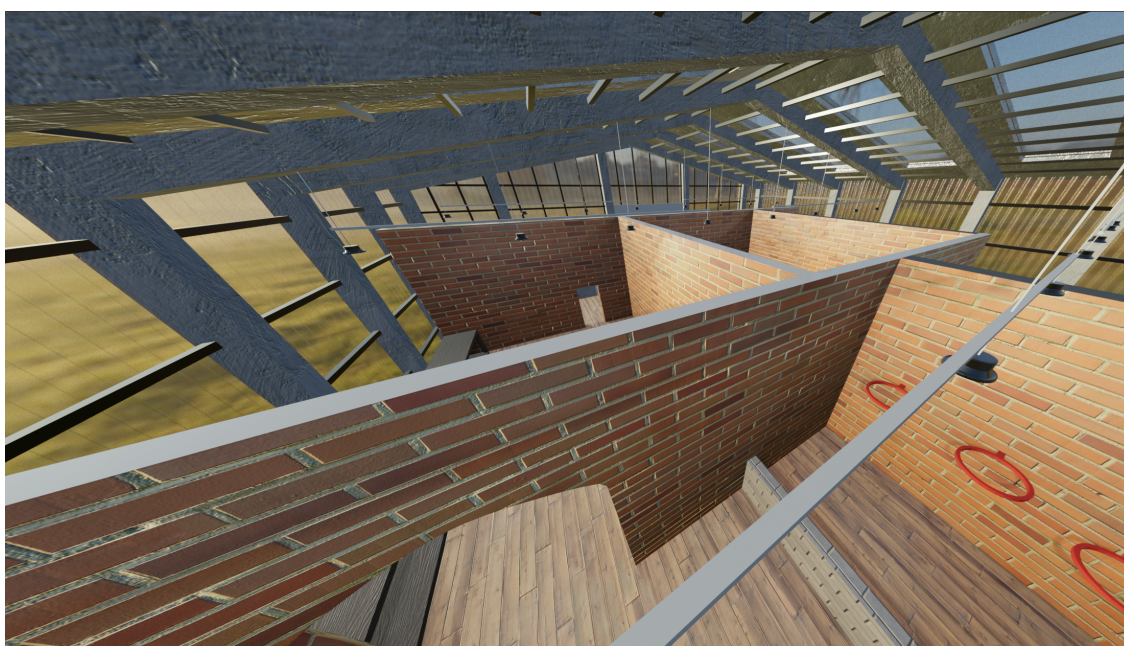
Obr. 6.1: Pohľad na pôvodný model garáže zvonku



Obr. 6.2: Pohľad na upravenú verziu garáže zvonku



Obr. 6.3: Pohľad na pôvodný model garáže zvnútra

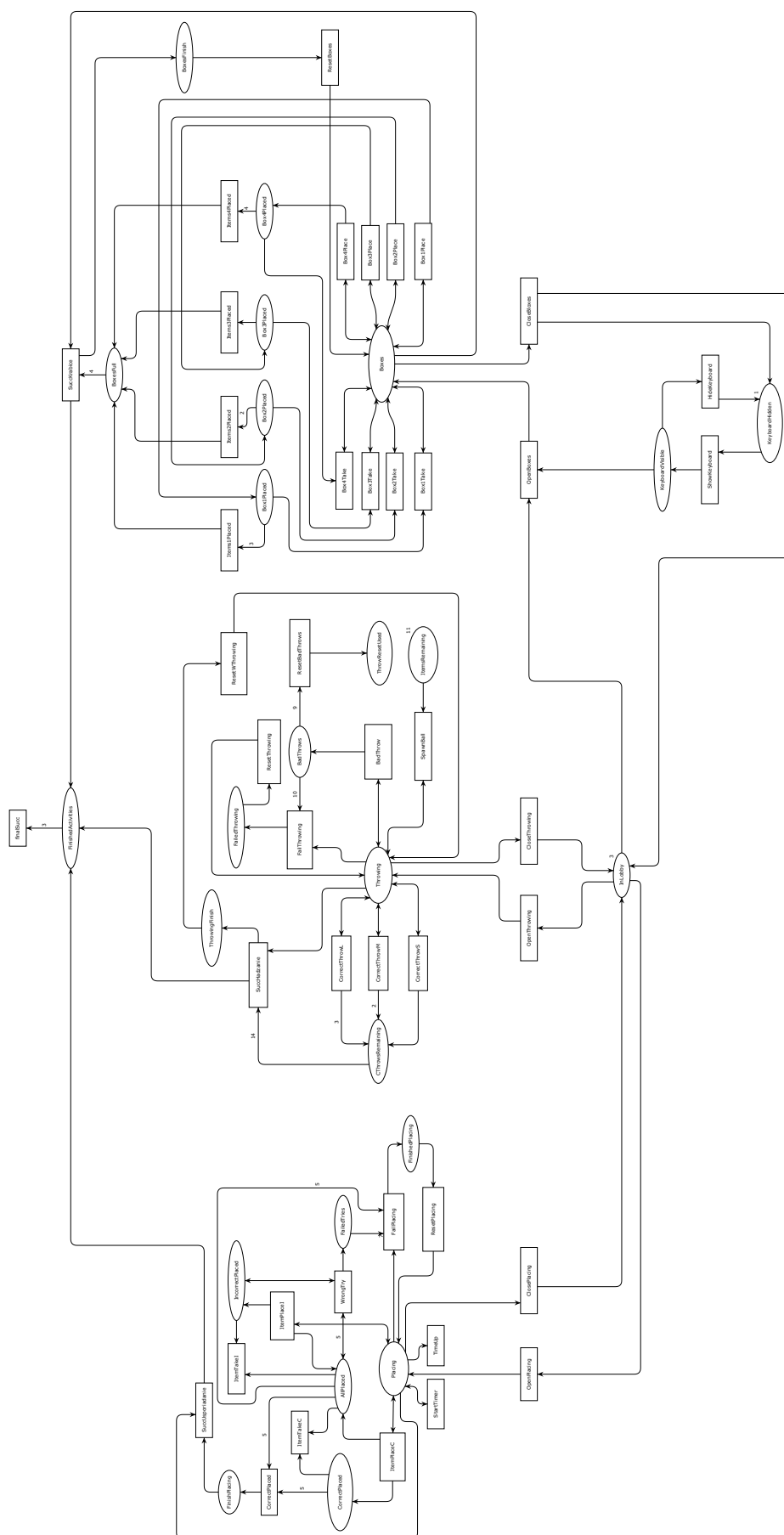


Obr. 6.4: Pohľad na upravenú verziu garáže zvnútra

Takto vytvorená scéna bola ďalej vložená do A-Frame rámca, kde tvorí základ pre všetky ďalšie úpravy a implementácie nových komponentov.

7 Scenár úloh

Podobne, ako v predošlých riešeniach je scenár, ktorý riadi priebeh plnenia úloh v ukázkovej scéne, implementovaný pomocou Petriho sietí. Táto sieť je rozdelená na 3 podsiete, predstavujúce 3 miestnosti v scéne, pričom každá z nich riadi iný typ úlohy. Vstup do každej miestnosti je možný iba po úspešnom otvorení jedným z nových typov dverí, ktoré slúžia, ako ukážka možností nových komponentov. Na konci každej úlohy je hráč informovaný o úspešnom, či neúspešnom splnení úlohy a je možné pokračovať do ďalšej miestnosti. V nasledujúcich častiach sú podrobne popísané jednotlivé úlohy, ktoré sú v miestnostiach obsiahnuté. Celá Petriho sieť je zobrazená na Obr. 7.1.



Obr. 7.1: Kompletná Petriho sieť riadiaca priebeh úloh v ukážkovej scéne

7.1 Miestnosť 1: Hádzanie lopty do koša

Prvá miestnosť sa ako jediná skladá z dvoch častí, ktoré sú navzájom prepojené. Motiváciou na jej vytvorenie bolo využitie fyzikálnych vlastností nového fyzikálneho enginu Ammo.js (časť 8.1.1), ktoré sa uplatňujú hlavne v hádzaní lopty do košov.

7.1.1 Popis miestnosti

Za prvými zatvorenými dverami, ktoré sa otvoria po stlačení tlačidla na stene (Obr. 7.2) sa nachádza miestnosť riadená prvým scenárom. V tejto miestnosti (Obr. 7.3) musí hráč najprv položiť správny typ predmetu (v rámci tejto ukážky to je ružový valec) do skenera, ktorý predmet odstráni z miestnosti a vyhodí z rúry v zadnej časti miestnosti červenú loptu.



Obr. 7.2: Dvere do prvej miestnosti



Obr. 7.3: Prvá miestnosť – časť so skenerom

Tú musí potom hráč v obmedzenom počte pokusov trafiť do jedného z troch košov (Obr. 7.4), pričom každý kôš má priradený určitý počet bodov. Po úspešnom alebo neúspešnom dokončení miestnosti je hráč informovaný o výsledku a môže pokračovať do ďalšej miestnosti. Pokiaľ hráčovi dochádzajú pokusy, zobrazí sa mu bonus (Obr.7.5), ktorý resetuje jeho pokusy a umožní mu pokračovať v hre. V miestnosti sa taktiež nachádzajú informačné tabule, ktoré hráča informujú o jeho postupe, ako aj o pravidlách hry.



Obr. 7.4: Prvá miestnosť – časť s košmi



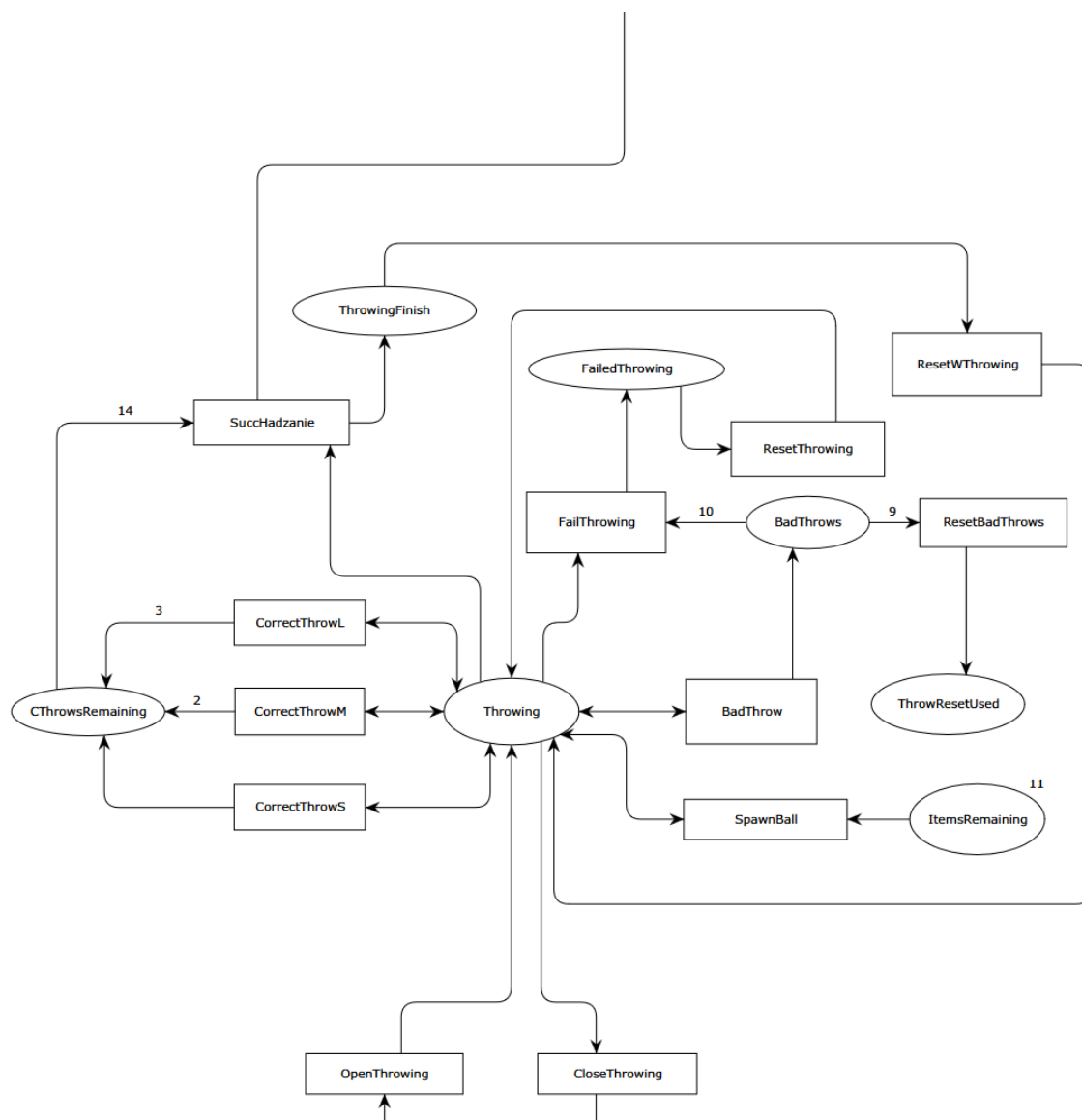
Obr. 7.5: Bonus v prvej miestnosti

7.1.2 Riadiaca Petriho podsieť

Petriho sieť riadiaca priebeh úloh v prvej miestnosti je zobrazená na Obr. 7.6. Po odpálení prechodu *OpenThrowing* sa token presunie¹ do miesta *Hádzanie*, ktoré je napojené na všetky hlavné prechody riadiace akcie v miestnosti. Po vložení správneho predmetu do skenera sa odpáli prechod *SpawnBall*, ktorý v scéne vytvorí loptu. Tento prechod zároveň odčíta token z miesta *ItemsRemaining*, ktorý slúži na uchovávanie stavu zostávajúcich správnych predmetov v prvej miestnosti, zobrazený na informačnej tabuli v miestnosti so skenerom (Obr. 7.3). Pomocou novovytvorenej červenej lopty je možné triafať 3 rôzne koše. Tieto koše majú rôzny počet bodov, čo je vyjadrené prechodmi *CorrectThrowS*, *CorrectThrowM* a *CorrectThrowL*. Po každom úspešnom pokuse je do miesta *CThrowsRemaining* vložený token s príslušnými bodmi. Bodový stav je taktiež zobrazovaný na informačnej tabuli v zadnej miestnosti (Obr.7.4). Ihneď po dosiahnutí dostatočného množstva tokenov v mieste *CThrowsRemaining* sa odpáli prechod *SuccThrowing*, ktorý odoberie token z hlavného miesta *Hádzanie* a presunie ho do miesta *FinishedActivities*, ktoré uchováva počet úspešne ukončených miestností. Pokiaľ hráč ale nenahraje dostatočné množstvo bodov, čiže v mieste *BadThrows* naakumuluje dostatočne veľa tokenov pomocou viacnásobného odpálenia prechodu *BadThrow*, odomkne sa prechod *ResetBadThrows*. Ak bude tento prechod odpálený trafením guľôčky do zobrazeného terča (Obr. 7.5), resetujú sa tokeny v mieste *BadThrows* a hráč môže po-

¹odpálením prechodu sa token z miesta A skonzumuje a pridá sa do miesta B

kračovať v hre. Pokiaľ hráč netrafí kôš 10 krát, odomkne sa prechod *FailThrowing*, ktorý presunie token z miesta *Hádzanie*, čím znemožní pokračovanie v scenári, pokiaľ hráč neopakuje úlohu odznova odpálením prechodu *ResetThrowing*.



Obr. 7.6: Petriho sieť riadiaca priebeh úloh v prvej miestnosti

7.2 Miestnosť 2: Usporiadanie predmetov

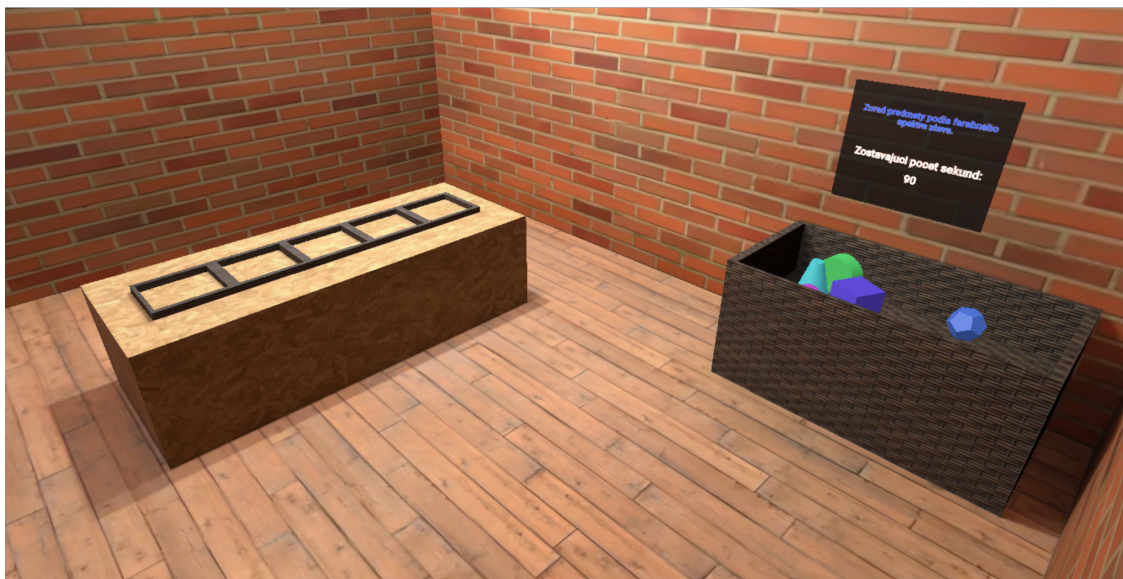
Druhá miestnosť je zameraná na usporiadanie predmetov do správneho poradia, pričom hráč má obmedzený čas na nájdenie správneho riešenia. Táto miestnosť je jedna z dvoch, ktorá slúži ako ukážka kategorizácie a usporiadania predmetov, ktoré sa v nej nachádzajú.

7.2.1 Popis miestnosti

Vstup do druhej miestnosti je podmienený sklopením páčky do správnej polohy (Obr. 7.7). V rámci zvýšenej miery interakcie s prostredím je nutné páčku fyzicky dostať do správnej polohy, či už pomocou virtuálnych rúk v rámci VR módu, alebo myšou mimo neho. Po otvorení dverí do miestnosti má hráč k dispozícii niekoľko predmetov, ktoré musí správnym spôsobom usporiadať na stôl v strede miestnosti (Obr. 37.8). V rámci tejto ukážky je predmety potrebné usporiadať podľa farebného spektra. Na obzvláštnenie úlohy má hráč na usporiadanie predmetov obmedzený čas, ktorý sa začne odpočítavať v momente, keď sa priblíži ku stolu. O časovom zostatku je hráč informovaný informačnou tabuľou, podobne ako v predošlej miestnosti. Keď umiestni všetky predmety do vyhradených miest, prostredie ho oboznámi o tom, či usporiadanie bolo správne. Pri zlom usporiadaní má hráč možnosť predmety premiestniť, pokiaľ mu to dovoľuje časový limit. Správnym usporiadaním predmetov je scenár druhej miestnosti úspešne dokončený.



Obr. 7.7: Dvere do druhej miestnosti



Obr. 7.8: Druhá miestnosť - kôš s predmetmi so stolom na ich usporiadanie

7.2.2 Riadiaca Petriho podsieť

Petriho sieť riadiaca priebeh úloh v druhej miestnosti je zobrazená na Obr. 7.9. Po odpálení prechodu *OpenPlacing* sa token presunie do miesta *Usporiadanie*, ktoré je napojené na všetky hlavné prechody riadiace akcie v miestnosti. Postupným umiestňovaním predmetov sa odpáľujú prechody *ItemPlaceC* a *ItemPlaceI* podľa toho, či bol predmet umiestnený na správnom mieste. Pokiaľ sa hráč rozhodne predmet odstrániť zo stola, odpália sa prechody *ItemTakeC*, alebo *ItemTakeI*, kde záleží na tom, či zdvihnutý predmet bol správne umiestnený, alebo nie. Pokiaľ sú všetky predmety správne umiestnené, sprístupní sa prechod *CorrectPlaced*, ktorý sa odpáli, keď hráč potvrdí usporiadanie predmetov na stole. Touto akciou sa presunie token do miesta *FinishPlacing* z ktorého sa ukončí úloha a pošle sa token do miesta *FinishedActivities*. Pokiaľ hráč neusporiada predmety správne, alebo mu dôjde čas, odpáli sa prechod *WrongTry*, ktorý sa sprístupní po umiestnení aspoň jedného predmetu na zlé miesto odpálením prechodu *ItemPlaceI*. Pokiaľ hráč takto nesprávne uloží predmety sedem krát, úloha sa ukončí neúspešne odpálením prechodu *FailPlacing*, ktorý odstráni token z miesta *Usporiadanie* a znemožní pokračovanie v scenári.

9 1 1

[illegible]

červeného otáznika, ktorý má hráča nabádať preskúmať aj túto časť scény.



Obr. 7.10: Nápoveda k tajným dverám

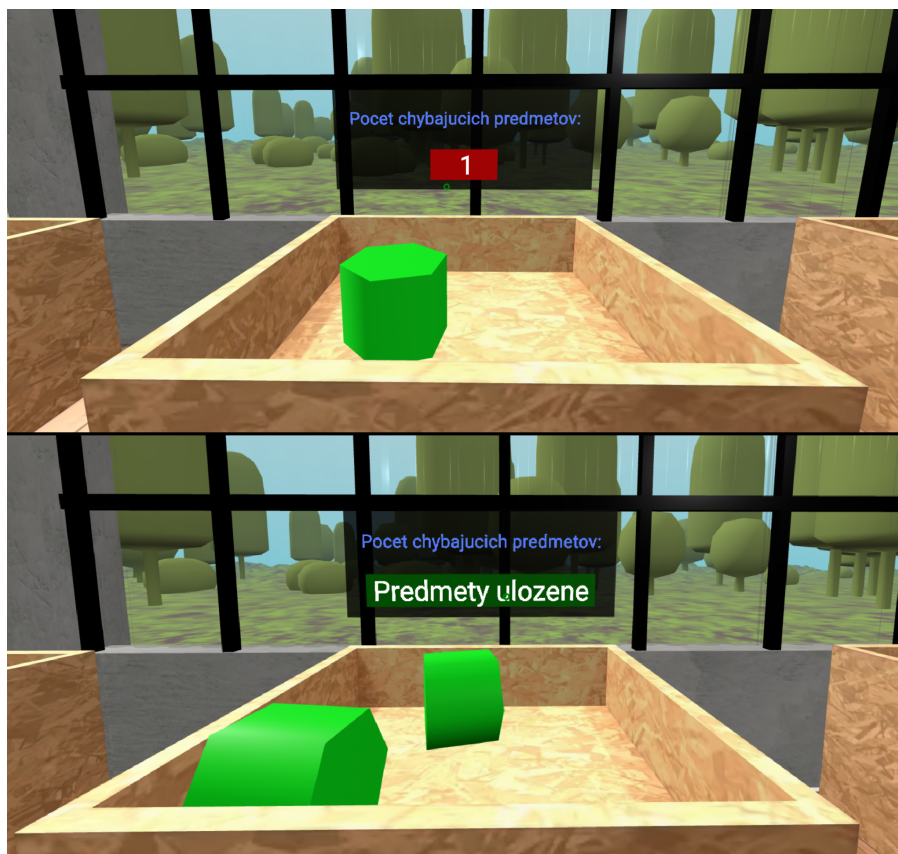
Ako sa hráč približuje, otáznik pomaly mizne a objaví sa pred ním komponent klávesnice (Obr. A.14) a textového vstupu, pričom po vyfukaní správneho hesla sa tajné dvere rozplynú a za nimi sa odhalí posledná miestnosť (Obr. 7.11).



Obr. 7.11: Otvorenie tajných dverí pomocou klávesnice

V nej musí hráč predmety uložené na stole podľa kategórie umiestniť do prí-

slušných boxov. Pre demonštračné účely bola ako kategória zvolená farba a tvar predmetov. Nad každým boxom sa nachádza informačná tabuľa, ktorá hráča informuje, koľko predmetov v boxe ešte chýba do správneho rozdelenia. Keď hráč uloží všetky potrebné predmety do daného boxu, ten sa považuje za dokončený, čo sa znova prejaví na informačnej tabuli (časť 7.12).



Obr. 7.12: Informačná tabuľa nad košmi pred a po uložení predmetov

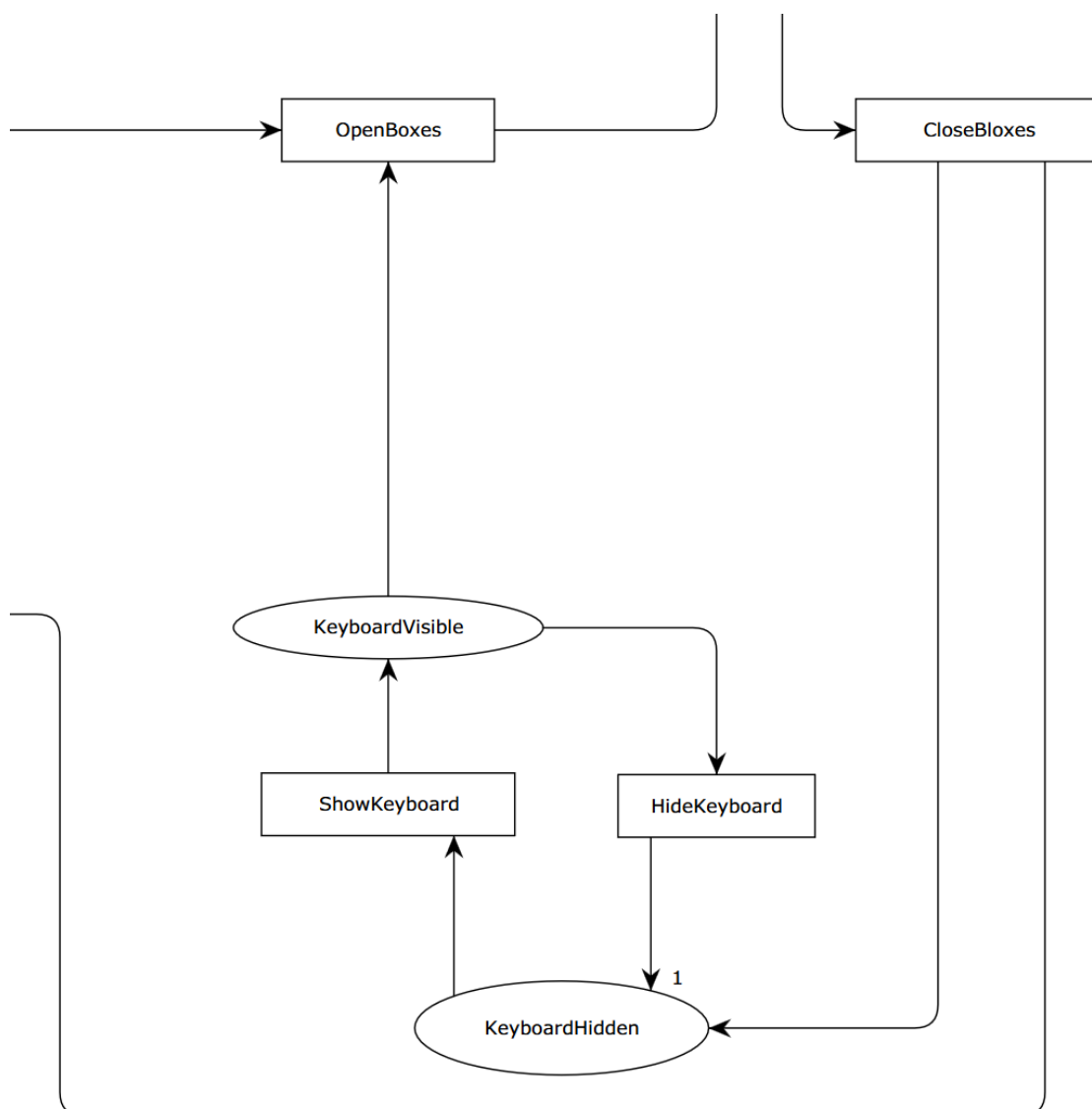
Po správnom usporiadaní predmetov sa hráčovi zobrazí informácia o úspešnom splnení úlohy a môže pokračovať do ďalšej miestnosti, ak mu ešte nejaké ostali.



Obr. 7.13: Tretia miestnosť - stôl s predmetmi a košmi

7.3.2 Riadiaca Petriho podsieť

Petriho sieť riadiaca priebeh úloh v tretej miestnosti je zobrazená na Obr. 7.15. Na rozdiel od zvyšných dvoch miestností, kedy vstup do miestnosti v rámci Petriho sietí bol podmienený iba jednoduchým odpálením prechodu, v tomto prípade to je o niečo zložitejšie, keďže je nutné riadiť viditeľnosť klávesnice podľa vzdialenosti hráča od otáznika. To je zabezpečené podsieťou na Obr. 7.14, kde sa striedavo odpalujú prechody *ShowKeyboard* a *HideKeyboard*, ktoré zobrazujú, alebo skrývajú klávesnicu a zároveň prenášajú token takým spôsobom, aby v daný moment bolo možné odpáliť iba jeden z nich. V rámci Obr. 7.14 sú tiež na porovnanie zobrazené prechody aj do ostatných miestností, ktoré neobsahujú žiadnu doplňujúcu podsieť, mimo jedného prechodu, ktorý posúva token do hlavného miesta v sieti.

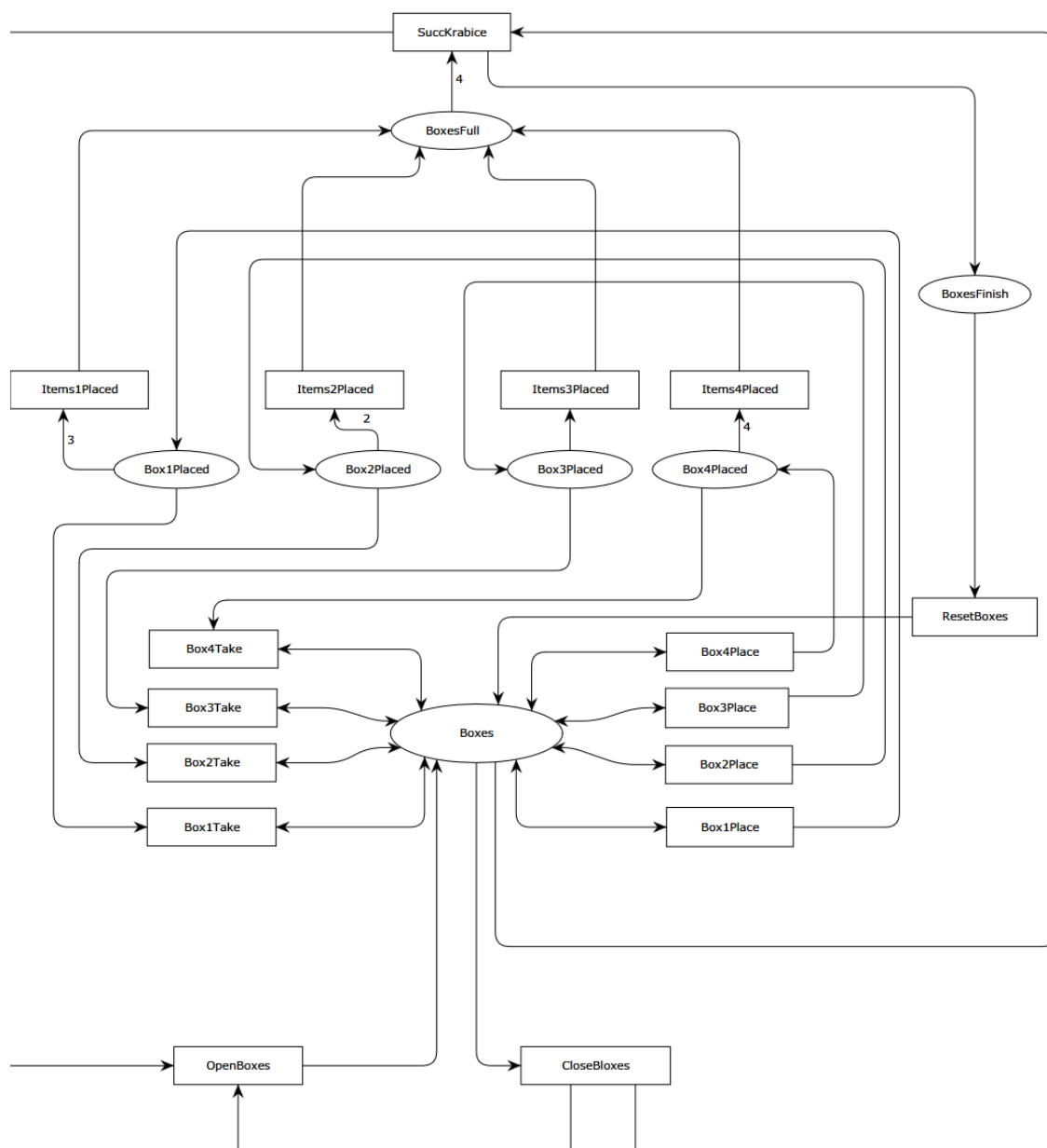


Obr. 7.14: Podsieť riadiaca zobrazenie klávesnice a textového vstup do tretej miestnosti

Petriho sieť miestnosti už tradične obsahuje hlavné miesto *Krabice*, z ktorého sú odpaľované všetky ostatné prechody v rámci scenára. Aj keď v základe je scenár tretej miestnosti veľmi podobný miestnosti *Usporiadanie*, sieť je zkonštruovaná v tomto prípade trochu inak. V rámci tejto siete sa rozlišujú tri typy prechodov:

- *BoxPlace* - prechod, ktorý sa odpáli, keď je predmet umiestnený do správneho koša
- *BoxTake* - prechod, ktorý sa odpáli, keď je predmet z koša odobratý
- *BoxPlaced* - prechod, ktorý sa odpáli, keď sú všetky predmety v koši správne umiestnené

Počet uložených predmetov sa ukladá vo forme tokenov do miest *BoxPlaced*, ktoré po dosiahnutí potrebnej hodnoty umožní odpálenie prechodu *ItemsPlaced*, ktorý posúva token do posledného miesta *BoxesFull*, kde sa koncentrujú tokeny dokončených boxov. Keď všetky boxy obsahujú správne predmety, podobne ako v predošlých sieťach sa odpáli prechod *SuccBoxes*, ktorý posunie token do finálneho miesta *FinishedActivities* a zároveň odoberie token z miesta *Krabice*, čím sa ukončí úloha.



Obr. 7.15: Petriho sieť riadiaca priebeh úloh v tretej miestnosti

8 Implementácia ukážkovej scény

Motiváciou pre spojenie všetkých implementovaných zmien do ukážkovej miestnosti bola okrem praktickej ukážky využitia nových komponentov aj ťažkopádny proces vývoja a navrhovania zložitejších scén v rámci rámca A-Frame. Aj keď je tento rámec známy svojou ľahkou prístupnosťou a jednoduchým použitím, vytváranie zložitejších scén, ktoré by obsahovali viacero interaktívnych prvkov, je často zdĺhavé a vyžaduje si dobrú znalosť rámca a jeho možností. Tento proces oboznamovania sa s rámcom je ale sťažený akútnym nedostatkom komplexných riešení a projektov, ktoré by slúžili ako vzor na plné využitie možností rámca. Taktiež existujúce komunitné, či oficiálne komponenty, ktoré by mali slúžiť na zjednodušenie tvorby scén, sú často zastaralé, nedokončené, či nekompatibilné s inými komponentmi. Ako príklad je možné uviesť knižnicu *Super Hands*¹, ktorá slúži ako platforma pre používanie rôznych ovládačov na manipuláciu s predmetmi vo virtuálnom prostredí. Táto knižnica je ale kompatibilná iba so starším a jednoduchým fyzikálnym enginom CANON.js a nie s enginom Ammo.js, ktorý ponúka podstatne viac možností a lepší výkon. Ako príklad nedokončeného, či neúplného komponentu môže slúžiť implementácia virtuálnej klávesnice *aframe-keyboard*², ktorá nepodporuje jej použitie s ovládačmi a nie je možné s ňou nijako manipulovať v priestore. Podobne komponent *Blink controls*³, ktorý slúži na teleportáciu po virtuálnom prostredí, nepodporuje používanie v uzavretom interieri, keďže v jeho základnej verzii sa hráč vie teleportovať aj cez steny. V neposlednom rade bolo taktiež nutné upraviť a doplniť komponenty na prácu s Petriho sieťami *clk-single-event-handler* a *clk-multi-event-handler* tak, aby boli vhodné na univerzálne použitie a boli kompatibilné s novými komponentmi. Tieto zmeny taktiež viedli k úpravám v objektoch scenára jednotlivých miestností, ktoré slúžia ako medzičlánok medzi Petriho sieťami a akciami v scéne. Všetky tieto problémy viedli k rozdeleniu implementačnej časti navrhovaného riešenia na dva celky, a to oprava a vylepšenie existujúcich komponentov a následná implementácia nových

¹<https://github.com/c-frame/aframe-super-hands-component>

²<https://github.com/WandererOU/aframe-keyboard>

³<https://github.com/jure/aframe-blink-controls>

komponentov. V rámci druhej časti implementácie bola určitá časť komponentov, ktorá slúžila ako základ ku vytvoreniu všetkých troch scenárov, navrhnutá na univerzálne použitie, neviazané na konkrétnu úlohu. Pre demonštrovanie možností týchto univerzálnych komponentov bolo mnoho mechanizmov a prvkov v scéne implementovaných ich viacnásobným použitím. Druhá časť nových komponentov slúži na ukážku, ako je možné využiť rámec A-Frame na tvorbu podobných nových komponentov, pre konkrétne potreby daného scenára.



Obr. 8.1: Štruktúra projektového adresára

Na Obr. 8.1 je zobrazená časť projektového adresára, ktorý obsahuje všetky nové a upravené komponenty, ako aj zdrojové súbory pre jednotlivé miestnosti a Petriho

siete.

8.1 Oprava a vylepšenie existujúcich komponentov

V rámci tejto časti tvorby ukážkovej miestnosti boli implementované nasledujúce zmeny:

- nahradenie fyziálneho enginu CANON.js za Ammo.js, ktorý ponúka podstatne viac možností a lepší výkon
- nový komponent, ktorý vytvorí presnú kolíznú mapu pre akýkoľvek 3D model, ktorý sa nachádza v scéne
- využitie komponentu *animation-mixer* na prehrávanie animácií 3D modelov, v rámci viacerých nových komponentov a objektov
- úprava komponentu *Blink controls*, ktorý znemožní teleportáciu cez steny
- nahradenie komponentu *Super Hands* za nový komponent, ktorý je kompatibilný s enginom Ammo.js
- prepracovaný a opravený komponent klávesnice, ktorý je možné ovládať vo VR pomocou raycastera a je možné s ním manipulovať v priestore ako s fyzickým objektom
- nový komponent *pn-transition-handler*, ktorý spája existujúcu funkcionálnosť komponentov *clk-single-event-handler*, *clk-multi-event-handler* a *collision-detector* a zároveň pridáva nové možnosti, ako riadiť scénu pomocou Petriho sietí
- refaktORIZÁCIA a doplnenie komponentu *petri-net-sim* pre kompatibilitu s novým komponentom *pn-transition-handler* a pridanie nových akcií, ktoré je možné v rámci Petriho siete vykonať
- zefektívnený spôsob písania scenárov pre Petriho siete využitím vzorových tried a rozhraní v súbore `transitions.ts`

V nasledujúcich častiach sú podrobne popísané jednotlivé úpravy a nové komponenty, ktoré boli implementované v rámci ukážkovej scény.

8.1.1 Fyzikálny engine Ammo.js

Keďže fyzikálny engine CANON.js, ktorý bol použitý v pôvodnom riešení, je zastaralý a nepodporuje toľko možností prispôsobenia rozhrania, rozhodli sme sa v rámci ukážkovej scény použiť druhý podporovaný engine knižnicou *aframe-physics-system*⁴ s názvom Ammo.js. Tento engine je priamym portom fyzikálnej knižnice Bullet, ktorá je implementovaná v jazyku Python a ponúka širokú škálu možností na komplexné, ale aj jednoduché fyzikálne simulácie. Ammo.js je teda implementovaný v Javascripte, čo umožňuje jeho použitie vo webových aplikáciách. Medzi jeho najväčšie výhody patrí použitie filtrovacích masiek na presnejšie riadenie kolízií, implementácia rôznych algoritmov na lepšie definovanie kolíznej mriežky objektov, či široká škála typov, ako previazať dva fyzikálne objekty (vhodné na simulovanie kyvadiel, pántov, páčok a pod.).

Na jeho použitie v rámci A-Frame rámca je nutné do koreňového elementu `<a-scene/>` pridať atribút `physics="driver: ammo"` a do každého objektu, ktorý sa má správať ako fyzikálny objekt, pridať komponent `ammo-body` s príslušnými hodnotami, pre inicializovanie objektu v rámci fyzikálneho enginu a komponent `ammo-shape`, ktorý definuje tvar objektu pre fyzikálny engine.

8.1.2 Komponent animation-mixer

Tento komponent⁵, ktorý je súčasťou základnej A-Frame knižnice *aframe-extras* umožňuje prehrávanie animácií 3D modelov vo formáte GLTF/GLB. Taktiež umožňuje nastavenie rýchlosti prehrávania animácie, opakovanie, nastavenie začiatku a konca animácie, či zachytenie udalostí, ktoré sa vyskytnú počas prehrávania animácie. V rámci ukážkovej scény je tento komponent využitý napríklad na prehranie animácie stlačenia tlačidla, otvorenia dverí, alebo cyklického pohybu objektov.

8.1.3 Komponent autofit-gltf

Popis a účel komponentu

A-frame komponent `autofit-gltf` slúži na vytvorenie presnej kolíznej mapy 3D modelov vo formáte GLTF/GLB pre použitie v simuláciách s fyzikálnym enginom Ammo.js. Tento komponent umožňuje jednoduchú integráciu fyzikálnych vlast-

⁴<https://github.com/c-frame/aframe-physics-system/blob/master/AmmoDriver.md>

⁵<https://github.com/c-frame/aframe-extras/blob/master/src/loaders/README.md>

ností modelov do scény a poskytuje flexibilitu pri definovaní fyzikálneho typu (statický alebo dynamický). Komponent tiež zabezpečí, že pre všetky správne označené dynamické objekty v scéne budú fyzikálne vlastnosti nastavené až po načítaní hlavného modelu, čo predíde prepadnutiu objektov do podlahy a dlším problémom s kolíziami.

Schéma a parametre

Komponent obsahuje jednu konfiguračnú vstupnú hodnotu:

- **ammoType** (string): Určuje typ fyzikálneho tela pre model.
 - **static** (predvolená hodnota): Model je pevný, čiže po strete s iným objektom sa nehýbe.
 - **dynamic**: Model je pohyblivý, reaguje na fyzikálne sily a je možné s ním manipulovať.

Implementácia a funkcionality

Komponent nastaví kolíznu mapu po spustení udalosti `model-loaded`, ktorý sa vysieľa po načítaní 3D modelu do scény. Jeho hlavné funkcionality zahŕňajú:

1. Priradenie fyzikálnych vlastností modelu:

- Na základe hodnoty parametra `ammoType` komponent priradí modelu vlastnosť `ammo-body` s príslušným typom:
 - **Statické telo**: Využíva type: `static`.
 - **Dynamické telo**: Využíva type: `dynamic` s nastavením `activationState: disableDeactivation`, čo zaisťuje, že s dynamickými objektami je možné manipulovať bez časového obmedzenia.
- Modelu sa priradí vlastnosť `ammo-shape` s hodnotou type: `mesh`, ktorý vytvorí kolíznu sieť presne podľa sieťovej reprezentácie objektu.

2. Automatická konfigurácia dynamických objektov:

- Komponent identifikuje objekty v scéne s ID obsahujúcim reťazec `late` a priradí im nasledovné vlastnosti:
 - Vlastnosť `ammo-body` a `ammo-shape` pre inicializovanie dynamického objektu v rámci fyzikálneho enginu.
 - Pridá komponenty pre interakcie:

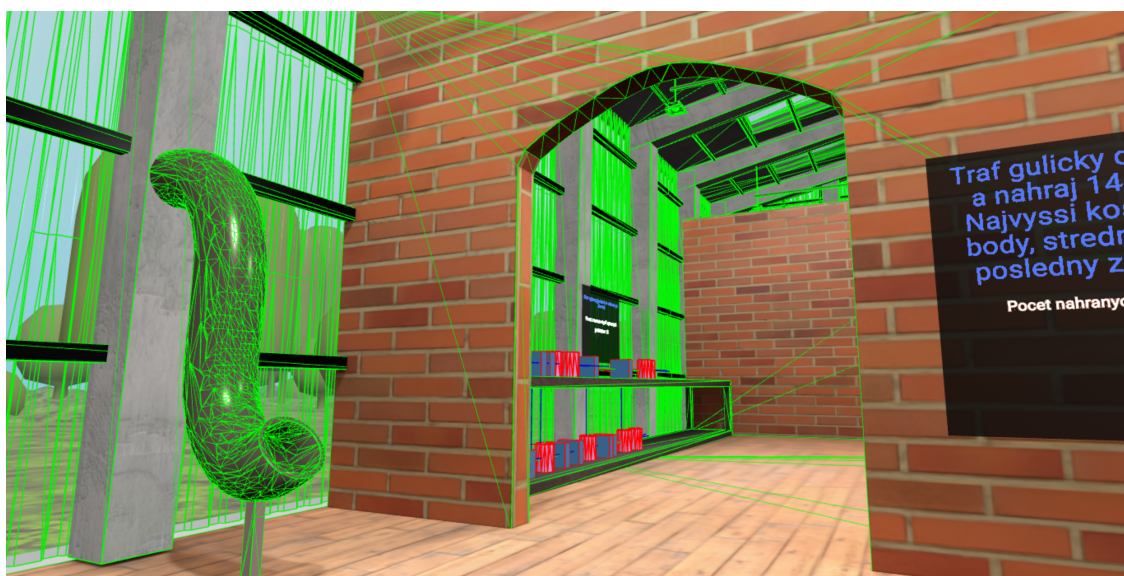
- * **grabbable**: Umožňuje uchopenie objektov - tu je podstatný atribút `constraintComponentName: ammo-constraint`, čo prispôsobí komponent *Super Hands* pre interakciu s Ammo.js fyzikálnym enginom.
- * **draggable**: Umožňuje presúvanie objektov.
- * **droppable**: Umožňuje púšťanie objektov.
- Pridá CSS triedu `interactable` označujúcu predmety, s ktorými je možné interagovať.

Príklad použitia vo VR scéne:

```
<a-entity gltf-model="path/to/model.gltf"
  autofit-gltf="ammoType: dynamic"/>
```

Použitie v rámci ukážkovej scény

Tento komponent sa používa na vytvorenie kolíznej mapy celého modelu scény, pričom po jeho načítaní inicializuje všetky dynamické objekty v jednotlivých miestnostiach. Aplikované kolízne mapy je možné vidieť na Obr. 8.2, zvýraznené zelenou farbou.



Obr. 8.2: Kolízna mapa miestnosti

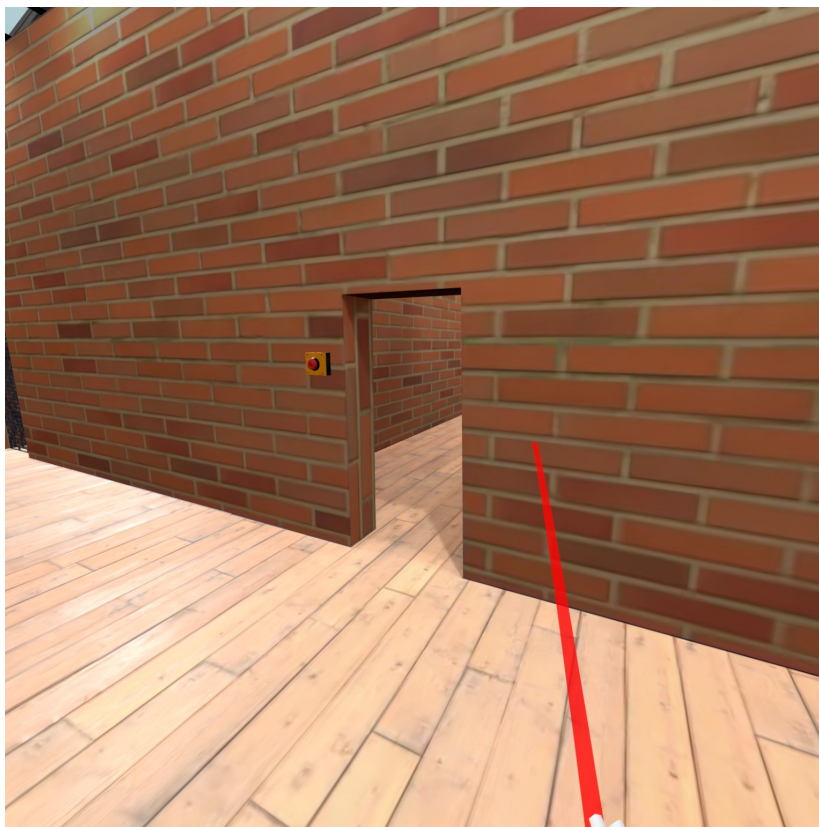
8.1.4 Úpravy komponentu **Blink controls**

Komponent `Blink controls` slúži na teleportáciu po virtuálnom prostredí pomocou určeného tlačidla na ovládači. Jeho princíp spočíva vo vyslaní paraboly

smerujúcej od ovládača do scény smerom na definované kolízne entity, na povrch ktorých sa hráč po uvoľnení tlačidla teleportuje. V pôvodnom riešení komponent prechádzal všetky kolízie s definovanými entitami a určil miesto teleportovania na vodorovný povrch poslednej v poradí. To spôsobovalo, že aj keď medzi kolízne entity boli zaradené steny, komponent vybral ako miesto teleportovania podlahu za stenou, čo narúša princíp oddelených miestností v scéne. V upravenej verzii tohto komponentu sa umožní teleportácia v prípade, že vyslaná parabola pretína iba jeden objekt (podlahu) a daný priesečník je na zemi (kvôli prípadu krajnej steny v miestnosti). Ak lúč pretína viacero objektov, teleportácia sa neuskutoční. Takto upravený komponent umožní jeho použitie iba v rámci miestnosti a eliminuje možnosť teleportácie cez steny. Na Obr. 8.3 je možné vidieť príklad úspešnej teleportácie v miestnosti, pričom hráčovi je povolené teleportovať sa iba na povrchy podlahy (lúč je zvýraznený modrou farbou). Na Obr. 8.4 je zobrazený príklad neúspešnej teleportácie, keď hráč sa snaží teleportovať za stenu (lúč je zvýraznený červenou farbou).



Obr. 8.3: Príklad úspešnej teleportácie



Obr. 8.4: Príklad neúspešnej teleportácie

Pre správne fungovanie komponentu je potrebné do atribútu `collisionEntities` pridať všetky entity, ktoré majú predstavovať hranice scény, čiže steny, stropy, podlahy, ale aj neviditeľné bariéry.

8.1.5 Nová verzia komponentu **Super Hands**

Výmenou hlavného fyzikálneho enginu scény za Ammo.js sa pôvodný komponent *Super Hands* stal nepoužiteľným, keďže od základu nepodporuje použitie iného enginu, ako CANON.js. Riešenie tohoto problému sa naskytlo v neoficiálnej modifikácii komponentu⁶, ktorý Ammo.js už podporuje. V tomto prípade nebolo potrebné vytvárať vlastné riešenie, ale stačilo využiť túto modifikovanú verziu komponentu.

Čo sa týka použitia komponentu, je nutné pri jeho inicializácii na objektoch v scéne pridať nasledujúce atribúty naviac, oproti základnej konfigurácie pre engine CANON.js:

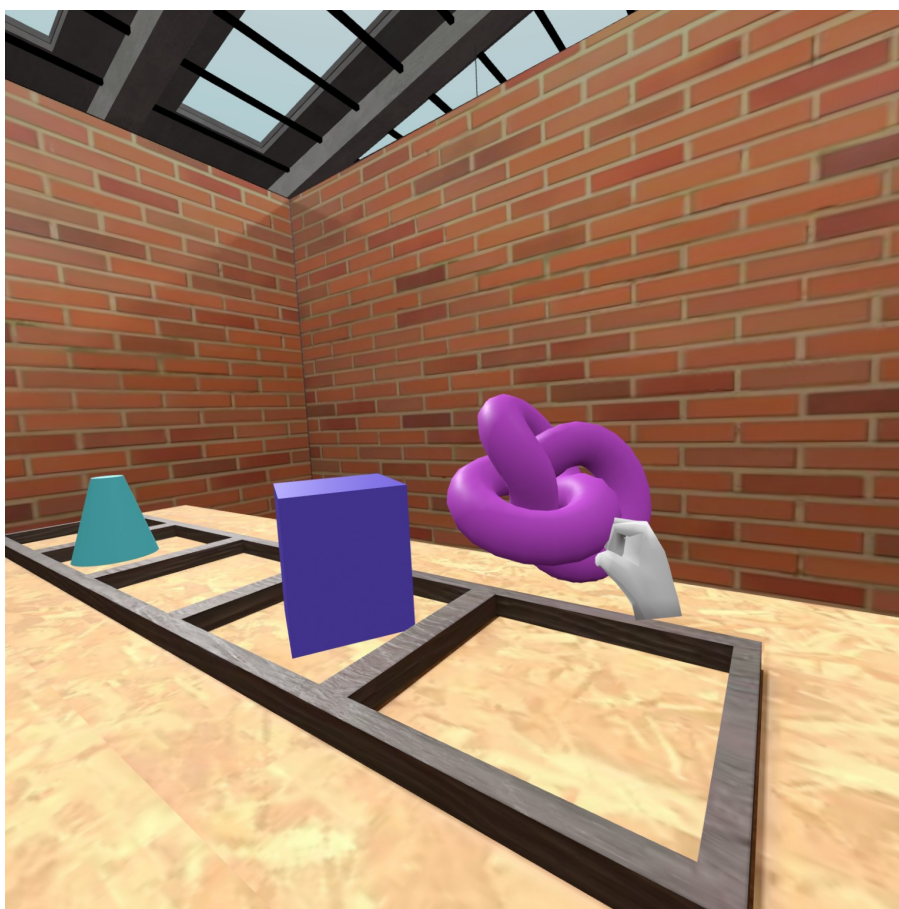
- `constraintComponentName: ammo-constraint`: Tento atribút, ktorý sa na-

⁶<https://github.com/diarmidmackenzie/aframe-super-hands-component>

stavuje v rámci atribútu `grabbable`, zabezpečí, že komponent bude pri vykonávaní interakcií s objektami v scéne využívať fyzikálny engine `Ammo.js`.

- `activationState: disableDeactivation`: Keďže fyzikálny engine `Ammo.js` má nastavené deaktivovanie fyziky dynamických objektov po určitom čase, tento atribút zabezpečí, že s objektami bude možné manipulovať neobmedzene.

Ďalšie atribúty a konfigurácie komponentu zostávajú rovnaké ako v pôvodnom riešení.



Obr. 8.5: Ukážka použitia komponentu `Super Hands` na interakciu s objektami

8.1.6 Prepracovaný komponent `aframe-keyboard`

Ako posledným komponent, ktorý prešiel úpravami je virtuálna klávesnica `aframe-keyboard`, ktorá je taktiež pôvodne vytvorená komunitou. V základnej verzii je klávesnica statická, umožňuje prepínanie medzi veľkými, malými písmenami, či špeciálnymi znakmi. Jej ovládanie je možné buď pomocou myši, alebo hardvérovej klávesnice. Takto navrhnutá klávesnica je ale vo VR s ovládačmi nepoužiteľná, keďže ich nepodporuje. Ďalší nedostatok vyplýva z charakteru ukážkovej scény,

ktorej základom je interaktivita s prostredím aj formou fyzického presúvania objektov. V základnej verzii je klávesnica statická a nie je možné s ňou manipulovať v priestore, čo môže pôsobiť obmedzujúco, hlavne keď väčšina okien v rámci používateľského rozhrania virtuálnych okuliarí umožňuje túto funkcionality. V neposlednom rade klávesnica obsahovala aj zopár chýb s kódovaním znakov, ktoré sa prejavili pri používaní kurzora na písanie. Tie boli ale pred samotnou implementáciou nových riešení odstránené. Z týchto nedostatkov teda vyplynuli dva hlavné vylepšenia, ktoré boli implementované, a to možnosť ovládania klávesnice vo VR pomocou raycastera a možnosť manipulácie s klávesnicou v priestore ako s fyzickým objektom.

Ovládanie klávesnice vo VR

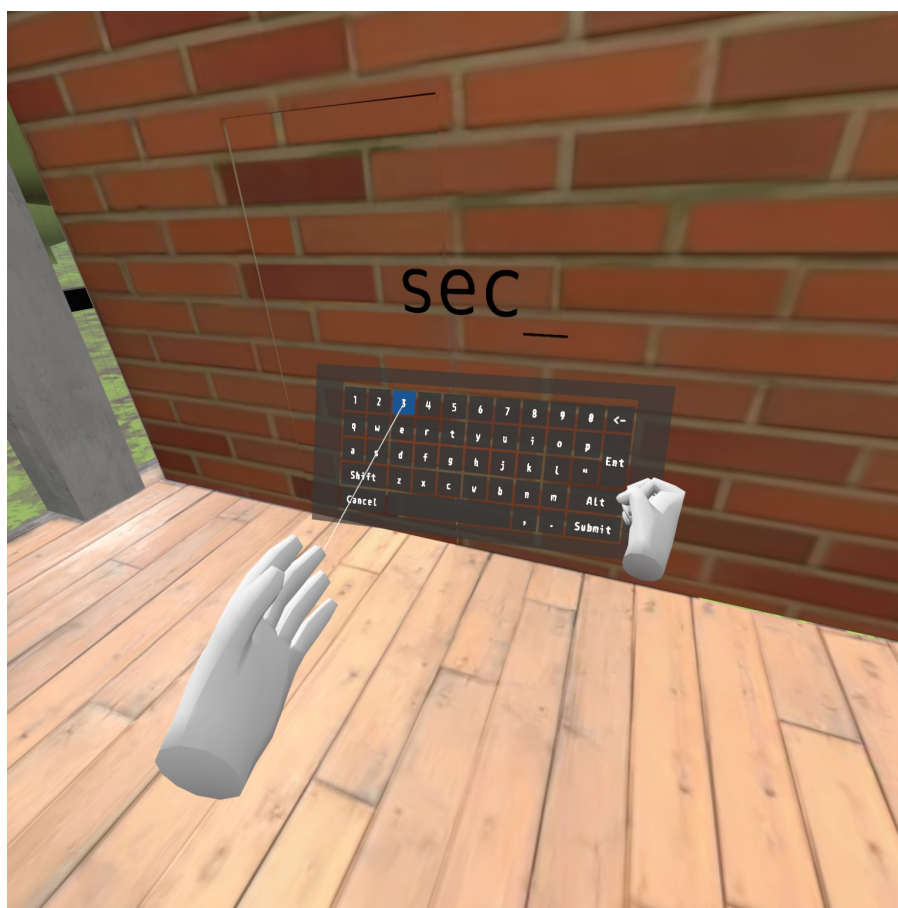
Pre ovládanie klávesnice vo VR bolo pre lepšiu presnosť vybrané riešenie pomocou raycastera, ktorý umožňuje presnejšie určenie pozície a umožňuje použiť kompaktnejšie rozmery klávesnice, oproti riešeniu s fyzickým stláčaním jednotlivých kláves. V rámci tejto úpravy bola taktiež odstránená možnosť ovládať klávesnicu hardvérovou klávesnicou, keďže táto funkcionality nie je vo VR praktická a mimo VR je v konflikte s ovládaním pohybu hráča po scéne.

Implementácia tohto riešenia bola pomerne jednoduchá, v rámci vykresľovania jednotlivých tlačidiel bolo potrebné pridať atribút `class: collidable`, ktorý označuje, že daný objekt interaguje s raycasterom. Potom v rámci inicializácie raycastera je nutné pridať túto triedu medzi entity, ktoré pri kolízii raycaster berie do úvahy nasledovne: `raycaster="objects: .collidable"`. Samotnú klávesnicu je možné pridať jednoducho inicializovaním komponentu `a-keyboard` v rámci entity. Pre zobrazenie výstupu bol vytvorený vzorový komponent `update-input`, ktorý reaguje na udalosti klávesnice a zobrazuje textový výstup v scéne a taktiež riadi otvorenie tajných dverí v miestnosti po zadaní správneho hesla. Na Obr. A.14 je možné vidieť proces písania hľadania hesla pomocou raycastera, ktorý vychádza z ovládača ľavej ruky.

Manipulácia s klávesnicou v priestore

Narozdiel od predošlého vylepšenia si táto zmena vyžadovala značne viac práce a úprav, keďže pôvodný komponent na túto zmenu nebol prispôsobený. Hlavný problém pri tvorbe riešenia predstavoval konflikt, medzi akciou presúvania a písania na klávesnici, hlavne pri používaní raycastera a mimo VR módu. V takom prípade sú tieto akcie naviazané na spoločné tlačidlo, čo môže spôsobiť pri mani-

pulácii s klávesnicou neočakávané správanie. V takom prípade sa naskytá riešenie zmeniť tlačidlo pre presun klávesnice mimo VR mód, alebo ho odstrániť úplne vo VR móde, čím sa znemožní presun raycasterom. Toto riešenie je ale málo univerzálne pre prípadné budúce zmeny. Preto sa ako riešenie tohto problému vybralo oddelenie rámu klávesnice od samotných kláves, čo efektívne oddeľuje akciu písania od akcie presúvania klávesnice. To bolo zabezpečené pridaním rámu okolo kláves, určeného na presun celej klávesnice, pričom tento rám nezasahuje do priestoru jednotlivých kláves. Toto riešenie taktiež ponecháva možnosť presunu klávesnice pomocou tlačidiel myši mimo VR módu, ako aj presun pomocou raycastera vo VR móde. Na Obr. A.14 je taktiež zobrazená manipulácia klávesnice pomocou komponentu *Super Hands* vyjadrená držaním klávesnice ovládačom pravej ruky.



Obr. 8.6: Ukážka manipulácie s klávesnicou vo VR

Pre implementovanie pohyblivej klávesnice sa použilo primitívum `<a-ring/>` so štyrmi segmentami na vytvorenie pravouhlého rámu. Takto vytvorený rám sa potom pridal medzi fyzikálne objekty pomocou atribútov `ammo-body` a `ammo-shape`, ďalej pomocou atribútu `grabbable` sa pridala možnosť presúvania klávesnice vo

VR móde a pomocou atribútu `collidable` sa zabezpečilo, že klávesnica bude reagovať na raycaster. Pre pohyb klávesnice ako celku sa entity rámu a kláves priradili ako potomkovia pod hlavný komponent klávesnice, pričom pozícia a rotácie rámu klávesnice sa prenáša na všetky klávesy. Táto akcia sa opakuje každé volanie funkcie `tick`, ktorá sa volá pri každom vykresľovaní scény.

8.1.7 Nový komponent `pn-transition-handler`

V pôvodnom riešení boli v rámci Petriho sietí definované tri typy udalostí, a to prepínacia, kolízna a jednoduchá. Jednoduchá udalosť reagovala na kliknutie myšou na sledovanom objekte, po ktorom sa odpálil určený prechod v sieti a vyslala signál komponentom, ktoré mali na zmenu stavu Petriho siete reagovať. Podobne prepínacia udalosť sa aktivovala po kliknutí myšou na objekte, len v tejto udalosti sa na základe prepínania vnútorného stavu odpálila striedavo jedna z dvoch definovaných prechodov v sieti. Nakoniec kolízna udalosť reagovala na kolíziu s objektom, čiže odpaľovala prechody pri vstupe a výstupe z kolíznej oblasti. V rámci predošlej implementácie tohto komponentu bol definovaný taktiež špeciálny typ udalosti, a to vstup do a výstup z miestnosti. Motiváciou pre úpravu troch spomenutých udalostí do jednej spoločnej boli nasledujúce okolnosti:

- **Duplicita kódu:** Všetky tri udalosti fungujú na rovnakom základe, čiže je nelogické mať ich implementované ako tri samostatné komponenty.
- **Príliš malá modularita:** V pôvodnej verzii reagujú komponenty iba na klik myšou a kolízne udalosti z `CANON.js` enginu, pričom každý nový typ udalosti by si vyžadoval zásah do kódu týchto komponentov.
- **Nové udalosti:** Súčasťou návrhu ukážkovej scény sú aj nové typy interakcií, ktoré nie je možné implementovať v rámci pôvodných komponentov, čiže je nutné ich rozšírenie o tieto nové akcie.

Základom riešenia prvého a tretieho bodu bolo vytvoriť enumeračný typ pre všetky pôvodné, aj nové akcie. Medzi nové akcie patria takzvané *SomeEnabledFromMany*, ktorá odpáli *n* aktivovaných akcií zo vstupného zoznamu prechodov v sieti a *AllEnabled*, ktorá odpáli všetky aktivované prechody zo vstupného zoznamu prechodov. Pre každú akciu bola pridaná aj vedľajšie akcia *CheckMarkingsFor*, ktorá po odpálení prechodu vráti informáciu o počte tokenov v danom mieste v sieti. Táto podakcia sa vykoná iba vtedy, ak používateľ pre atribút `checkMarkingsFor` zadal názov miesta v sieti. Tieto akcie slúžia hlavne na jednoduché použitie najčastejších akcií v Petriho sieťach, čo zvyšuje modularitu a znižuje duplicitu kódu.

V jadre komponent `pn-transition-handler` pozná v podstate iba jeden druh akcie, ktorý odpáli `n` aktivovaných prechodov zo vstupného zoznamu a spraví to buď v poradí, v akom boli prechody v komponente definované, alebo náhodne. Spôsob, akým sa aktivované prechody vyberajú, je naviazaný na prepínač `fireRandom`. Táto nová funkcionality nepredstavuje len rozšírenie pôvodnej, ale aj zmenu v základnom fungovaní komponentu, ktorý sme implementovali až v rámci tejto záverečnej práci. Pre správne fungovanie komponentu, boli navrhnuté nasledujúce vstupné parametre:

- **transitions** - Zoznam prechodov potrebných na vykonanie akcie.
- **action** - Typ akcie, ktorá sa má vykonať. Hodnota je jeden z prvkov enumeračného typu `ActionType`.
- **n** - Počet aktivovaných prechodov, ktoré sa majú odpáliť
- **fireRandom** - Hodnota tohto atribútu určuje, či sa aktivované prechody majú odpáliť v poradí, alebo náhodne.
- **selected** - Stavová premenná používaná pri prepínacej akcii.
- **affectedElements** - Zoznam objektov v scéne, ktoré sú ovplyvnené prechodom.
- **listenToEvents** - Zoznam udalostí, na ktoré má komponent reagovať.
- **checkMarkingsFor** - Hodnotou tohto atribútu je názov miesta v sieti, pre ktoré sa má vrátiť počet tokenov po odpálení prechodu. Tento atribút je nepovinný.

Všetky tieto parametre sa ďalej posúvajú do hlavného komponentu *petri-net-sim*, ktorý na základe typu akcie vykoná príslušnú udalosť okrem nasledujúcich parametrov, ktoré sa použijú nasledovne:

- **event** - Typ udalosti, ktorý sa vykoná v hlavnom komponente *petri-net-sim*. Pokiaľ ide o akciu reagujúcu na udalosť v scéne, hodnota atribútu je `SceneEvent.firedTransition`. Ďalšie, menej používané hodnoty sú `SceneEvent.enteredPlace`, `SceneEvent.leftPlace` a `SceneEvent.petriNetLoaded`.
- **message** - Interný atribút, ktorý podľa typu akcie obsahuje názov prechodu, ktorý sa má odpáliť. Pochádza z hlavného atribútu `transitions`.

Atribút *listenToEvents* sa ako jediný neposúva do žiadneho komponentu, pretože sa používa iba na naviazanie poslucháčov udalostí na komponente, na základe ktorých sa vykonáva akcia. Aký typ akcie sa má vykonať sa potom podľa týchto všetkých vstupných údajov rozhodne v metóde *init*, kde sa pomocou použitia *switch* príkazu vykoná príslušná akcia. Zvyšné metódy v komponente sa vykonávajú po úspešnom odpálení prechodu. To znamená, že nad objektami z atribútu *affectedElements* sa vyšlú udalosti s informáciou o vykonanej akcii. Inými slovami, odpálenie prechodu vyšle udalosť *transitionFired* s názvom prechodu, ktorý sa odpálil a získanie množstva tokenov v mieste vyšle udalosť *getPlaceMarking* s počtom tokenov v mieste. Akcie, ktoré vyvolajú odpálenie viacerých prechodov v sieti vysielajú udalosti pre každý prechod zvlášť, pričom samotná udalosť nesie meno odpáleného prechodu. V neposlednom rade, pokiaľ je prechod aktivovaný, ale nebol odpálený, vyšle sa udalosť *transitionEnabled* s názvom prechodu.

8.1.8 Zmena komponentu *petri-net-sim*

Ďalším krokom po inicializovaní nového komponentu *pn-transition-handler* bolo prispôbenie komponentu *petri-net-sim* na nové akcie a jeho následná refaktoriácia. Tento komponent je určený pre správu Petriho sietí v scéne, ktorý reaguje na udalosti z *pn-transition-handler* komponentu. V pôvodnom riešení komponent dokázal odpáliť v podstate iba jeden prechod, či už vo forme jednoduchého prechodu, alebo pomocu prepínacej udalosti. Počas procesu hľadania a opdaľovania prechodu boli volané nasledujúce metódy:

- *ifTransitionFound* - volá sa, ak bol prechod nájdený v sieti
- *ifTransitionNotFound* - volá sa, ak prechod nebol nájdený v sieti
- *ifTransitionDisabled* - volá sa, ak prechod bol nájdený, ale nebol aktivovaný
- *ifTransitionEnabled* - volá sa, ak prechod bol nájdený a bol aktivovaný (následne aj odpálený)

Takáto štruktúra definovaná pre každý prechod v poli prechodov definovanom v súbore *transitions.ts* ale nebola dostatočne univerzálna pre nové akcie, ktoré boli pridané do komponentu *pn-transition-handler*. Preto bolo nutné túto štruktúru upraviť a pridať nové metódy, ktoré by reagovali na nové akcie.

Školiteľom práce bola navrhnutá nová zostava metód prechodu a príslušný algoritmus činnosti komponentu, ktoré boli v tejto práci implementované. Nová zostava metód sa od predchádzajúcej líši v nasledujúcom:

- odstránenie metódy `ifTransitionEnabled`
- nová metóda `ifTransitionFired` - volá sa, ak bol prechod odpálený
- nová metóda `ifTransitionEnabledNotFired` - volá sa, ak aktivovaný prechod nebol odpálený

Ostatné metódy ostali nezmenené.

Samotné fungovanie komponentu má novú, všeobecnú funkcionálnu, ktorá je opísaná nasledujúcim algoritmom:

Algorithm 1 Spúšťanie prechodov v Petriho sieťach

Require: Zoznam prechodov tl , parameter n , boolean $fromFirst$, Petriho sieť net

Ensure: Odpálenie nanajvýš n aktivovaných prechodov z tl

1: $tl_{nf} \leftarrow$ prechody zo zoznamu tl , ktoré nie sú v sieti

2: $tl_f \leftarrow$ prechody zo zoznamu tl , ktoré sú v sieti

3: **while** tl_{nf} nie je prázdny **do**

4: $t \leftarrow \text{removeFirst}(tl_{nf})$

5: $t.\text{ifTransitionNotFound}()$

6: **end while**

7: **if** $n = \infty$ **then**

8: $n \leftarrow \text{length}(tl_f)$

9: **end if**

10: **while** tl_f nie je prázdny **do**

11: **if** $fromFirst$ **then**

12: $t \leftarrow \text{removeFirst}(tl_f)$

13: **else**

14: $t \leftarrow \text{removeRandom}(tl_f)$

15: **end if**

16: **if not** $net.\text{isEnabled}(t)$ **then**

17: $t.\text{ifTransitionDisabled}()$

18: **continue**

19: **end if**

20: **if** $n > 0$ **then**

21: $net.\text{fire}(t)$

22: $t.\text{ifTransitionFired}()$

23: $n \leftarrow n - 1$

24: **else**

25: $t.\text{ifTransitionEnabledNotFired}()$

26: **end if**

27: **end while**

Takáto štruktúra umožňuje vykonávať rôzne druhy akcií, aj nad rámec definovaných v súbore `actionTypes.enum.ts`.

8.1.9 Zjednodušenie a úprava skriptu scény

Pre reakciu scény na vykonanie nejakej akcie bol v pôvodnom projekte vytvorený koncept skriptu scény, ktorý obsahoval objekty prechodov obsahujúce ich názov a metódy volané z komponentu *petri-net-sim*. Tieto metódy potom ďalej v komponente *pn-transition-handler* volali príslušné metódy spomenuté na konci sekcie 8.1.7. Problémom ale bola neexistencia rozhraní, alebo základnej triedy, ktorá by určovala štruktúru prechodov a udalostí v scenári. To malo za následok, že tvorba skriptu bola ponechaná vo veľkej miere na autorovi scenára, čo viedlo častokrát ku neprehľadnému a redundantnému kódu. Pre porovnanie, základná verzia scény múzea [7] s 3 jednoduchšími miestami mala skript scény na 430 riadkoch kódu. Ako riešenie týchto problémov boli vytvorené dve základné rozhrania - *BaseTransition* a *BasePlace* (Obr. 8.7), ktoré obsahujú základné metódy pre vykonanie zmeny v scéne na základe vykonanej akcie, zahrnujúc zmeny predstavené v predošlých častiach.

```
export interface BaseTransition{
  transitionName: string;
  index: number;
  ifTransitionFound(targets: string[], replay: boolean): void;
  ifTransitionNotFound(targets: string[], replay: boolean): void;
  ifTransitionDisabled(targets: string[], replay: boolean): void;
  ifTransitionEnabledNotFired(targets: string[], replay: boolean): void;
  ifTransitionFired(targets: string[], replay: boolean): void;
}

export interface BasePlace{
  placeName: string;
  ifPlaceFound(placeName: string): void;
  ifPlaceNotFound(placeName: string): void;
  ifPlaceNotFoundOnStart(placeName: string): void;
  ifMarkingInformEnabled(marking: number, targets: string[]): void;
}
```

Obr. 8.7: Zdrojový kód nových rozhraní skriptu scény

Potom v rámci skriptu scény pre ukážkovú scénu boli pomocou týchto rozhraní vytvorené nové triedy, ktoré dopĺňajú funkcionality základných metód a prijímajú vstupné parametre. Vďaka tomu sa podarilo znížiť počet riadkov kódu na 150, čo predstavuje výrazné zníženie a lepšiu čitateľnosť.

8.2 Nové funkcionality a rozšírenia

Poslednou kategóriou nových komponentov, ktoré tvoria základ ukázkovej scény, sú originálne komponenty týkajúce sa priamo interakcie užívateľa so scénou. Keďže ukázková scéna nemá slúžiť iba ako demonštrácia možností A-Frame rámca a jeho použitie s Petriho sieťami, niektoré nové komponenty opísané v tejto kapitole boli navrhnuté tak, aby ich bolo možné použiť pri tvorbe širokého spektra nových scén implementovaných pre tento projekt. Aby bola dokázaná univerzálnosť týchto komponentov, scenár miestností bol navrhnutý tak, aby obsahoval rôzne typy interakcií, ktoré sú možné vytvoriť pomocou týchto komponentov. Medzi tieto univerzálne komponenty patria:

- *ammo-collision-transition*: spôsobí zmenu v scéne po kolízii s objektom
- *button-control*: umožňuje otváranie dverí po stlačení tlačidla
- *lever-control*: umožňuje otváranie dverí po potiahnutí páky
- *enable-text-on-marking*: zobrazí textové pole potom, ako miesto v sieti dosiahne určitý počet tokenov
- *text-marking-update*: zobrazuje aktuálny stav tokenov v mieste v Petriho sieti
- *timer*: spustí časovač po odpálení prechodu a následne po dosiahnutí limitu odpáli prechod o jeho ukončení
- *show-finish-info*: zobrazí informáciu o ukončení scenára

Zvyšné komponenty navrhnuté pre použitie iba pre túto scénu sú predstavené na konci tejto kapitoly, po detailnom popise univerzálnych komponentov.

8.2.1 Komponent *ammo-collision-transition*

Komponent *ammo-collision-transition* slúži na spracovanie kolízií v A-Frame prostredí s podporou fyzikálneho enginu Ammo.js. Umožňuje filtrovať kolidujúce objekty, vyvolávať prechody v Petriho sieti a tiež ponúka možnosť odstrániť alebo označiť objekty po kolízií.

Schéma komponentu

Komponent definuje nasledujúce parametre:

- **removeCollided** (boolean, default: false) - Ak je nastavený na true, po vyvolaní prechodu sa kolidujúci objekt odstráni zo scény.
- **markCollided** (boolean, default: false) - Označuje objekt po kolízii atribútom `hit` s hodnotou true.
- **filterGroup** (string, default: "correct") - Určuje skupinu objektov, ktoré majú vyvolať prechod v sieti.
- **filterGroupPrefix** (string, default:) - Prefix pre filtrovanie, používané pri filtrovaní viacerých skupín.
- **removeTimeout** (number, default: 1000) - Určuje, po akom čase sa má kolidujúci objekt odstrániť zo scény.
- **removeSelf** (boolean, default: false) - Ak je true, komponent sa po kolízii odstráni.
- **toggleTransition** (boolean, default: false) - Ak je true, opakované kolízne udalosti budú odpalovať vždy iný z dvoch prechodov. Používa sa pri `ActionType.ToggleEvent` v `pn-transition-handler` (časť 8.1.7).
- **fireIfWrong** (boolean, default: false) - Ak je true, vyvolá prechod aj pri kolízii s predmetom so správnym atribútom `filterGroupPrefix`, ale nesprávnym atribútom `filterGroup`.
- **multipleObjects** (boolean, default: false) - Ak je true, komponent vie naraz spracovať prepínacie udalosti, aj odpálenie prechodu po kolízii s nesprávnym objektom.

Spracovanie udalosti

Komponent reaguje na nasledujúce udalosti:

- **collidestart** - Pri detekcii kolízie skontroluje, či objekt patrí do filtrovanej skupiny a vyvolá prechod v Petriho sieti pomocou udalosti `fireTransition`. Pri atribúte `fireIfWrong == true` sa vyvolá prechod aj pri nesprávnom objekte.
- **collideend** - Ak je prepínač `toggleTransition` nastavený na hodnotu true, prechod sa vyvolá aj pri ukončení kolízie.
- **transitionFired** - Po vyvolaní prechodu v sieti odstráni, alebo označí objekt podľa nastavenia atribútov `removeCollided` a `markCollided`.

Použitie v scenári

Tento komponent sa používa v každej z troch ukážkových miestností. V miestnosti *Hádzanie* riadi najprv skenovanie predmetov v skeneri. Pri tejto akcii sa využíva možnosť odstránenie predmetu zo skenera. Udeje sa to iba ak je predmet v správnej skupine. Ďalej pri v druhej časti prvej miestnosti sa komponent používa na vyvolávanie rôznych akcií podľa toho, či a akú obruč hráč trafil. Pri tejto akcii sa znova využíva odstraňovanie kolidujúceho objektu, konkrétne pri páde lopty na zem po jej hodení. Tá sa odstráni po určitom časovom odstupe, vďaka atribútu `removeTimeout`. V druhej miestnosti sa komponent používa na štart časovača po pristúpení ku stolu a pri sledovaní správneho a nesprávneho zoradenia predmetov. Pri usporiadaní predmetov sa využíva prepínacia udalosť na sledovanie polohy a odstránenia predmetu zo stola a zároveň atribút `filterGroupPrefix` na sledovanie nesprávneho usporiadania predmetov. Keďže sa používa naraz prepínacia udalosť a udalosť pri nesprávnom usporiadaní, je nutné nastaviť atribút `multipleObjects` na hodnotu `true`. Nakoniec v tretej miestnosti sa používa pri určovaní korektnosti predmetu uloženého do príslušného boxu. Tu sa využíva iba prepínacia udalosť na vykonanie prechodu pri položení a zobrať predmetu z boxu.

8.2.2 Komponent *button-control*

Komponent *button-control* slúži na spustenie akcií po stlačení tlačidla medzi dverami a modelom navigačnej siete `navMesh`. Po kliknutí, alebo dotyku virtuálnej ruky s tlačidlom sa spustí animáciu stlačenia, vyšle sa udalosť na otvorenie dverí a následne sa vymení navigačná sieť, aby odrážala zmenu v prostredí. Navigačná sieť je 3D model vytvorený v Blenderi, pričom obsahuje niekoľko takzvaných uzlov, kde každý predstavuje jednu kombináciu otvorených a zatvorených dverí. Názvy týchto uzlov sú zostavené podľa dverí, ktoré sú aktuálne v scéne otvorené, pričom 1 znamená otvorené dvere a 0 zatvorené. Číže napríklad označenie 110 znamená, že prvé dvere a druhé dvere sú otvorené a tretie zatvorené.

Parametre komponentu

- **buttonId** - ID tlačidla, ktoré sa má aktivovať.
- **navMeshId** - ID navigačnej siete, ktorá sa má aktualizovať po zmene stavu dverí.
- **doorId** - ID dverí, ktoré majú reagovať na tlačidlo.

- **collisionId** - Číselný identifikátor slúžiaci pre navigačnú sieť na identifikáciu uzlu, ktorý sa má zobrazíť ako nová navigačná sieť.
- **animation** - Názov animácie tlačidla.

Spracovanie udalosti

Komponent reaguje na nasledujúce udalosti:

- **transitionFired** - Po vykonaní prechodu sa spustí metóda, ktorá vykoná animáciu stlačenia tlačidla a vyšle udalosť na otvorenie dverí.
- **animationcomplete** - Po dokončení animácie otvárania dverí sa aktualizuje navigačná sieť, aby reflektovala novú dostupnosť priestoru.

Použitie v scenári

Tento komponent sa používa na vstup do miestnosti *Hádzanie* (časť 7.1) na otvorenie dverí, ktoré do nej vedú.

8.2.3 Komponent *lever-control*

Komponent *lever-control* umožňuje otváranie dverí fyzickým potiahnutím páky v scéne. Pri prekročení určitého uhla náklonu páky sa automaticky spustí prechod, ktorý otvorí dvere a zmení navigačnú sieť *navMesh*, podobne ako pri tlačidle.

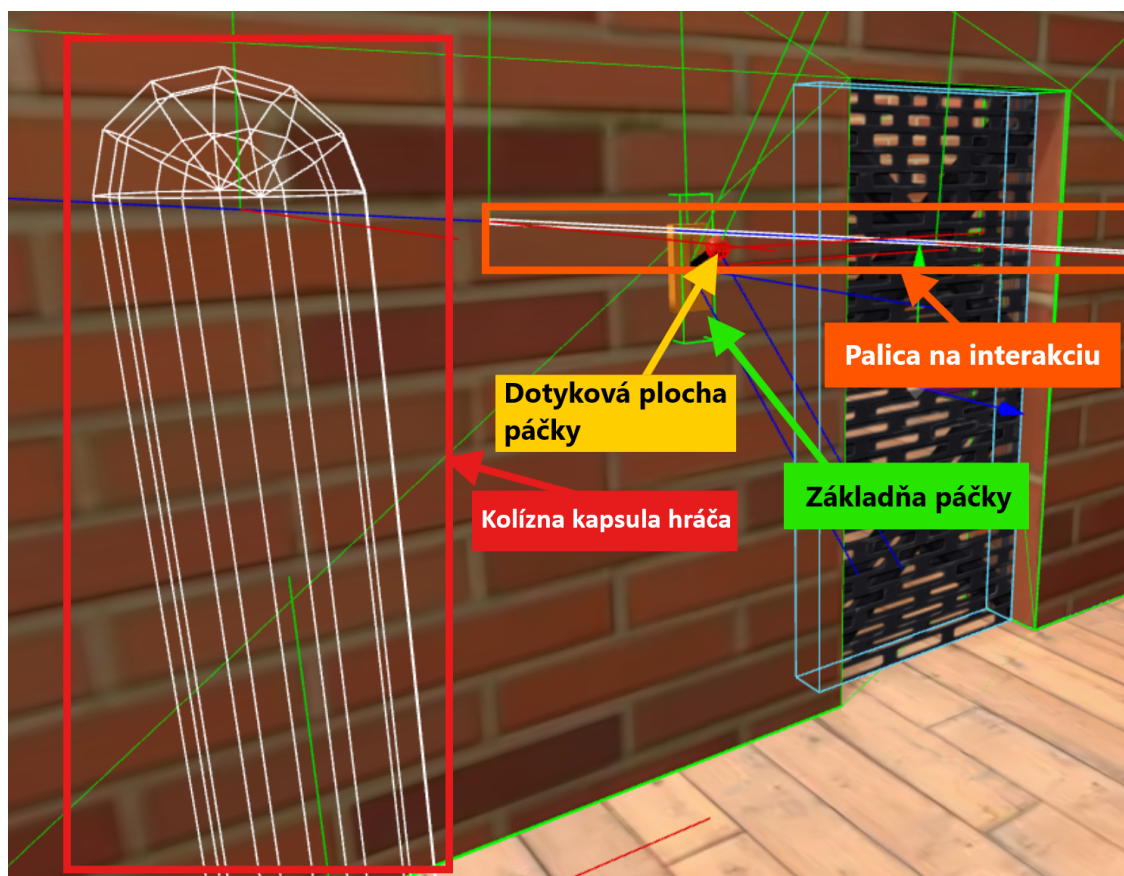
Parametre komponentu

- **navMeshId** – ID navigačnej siete, ktorá sa má po zmene stavu aktualizovať.
- **doorId** – ID dverí, ktoré sa majú otvoriť alebo zavrieť pri aktivácii páky.
- **collisionId** – Číselný identifikátor používaný na identifikáciu uzlu, ktorý sa má zobrazíť ako nová navigačná sieť.

Spracovanie udalosti

Potom, ako *pn-transition-handler* vykoná prechod v sieti, metóda *handleEvent* v komponente reaguje na túto udalosť a podobne ako v prípade tlačidla, informuje dvere o spustení animácie otvorenia, po ktorej sa vymení navigačná sieť podľa toho, aké dvere boli otvorené. Samotný princíp fungovania páčky sa líši pri klasickej a VR verzii. V klasickej verzii sa na raycaster upevnila neviditeľná „palica“,

ktorá má fyzikálne telo v rámci Ammo.js. Vďaka nej je možné posunúť páčku do správnej polohy, pričom koniec páčky je taktiež súčasťou fyziky, ako dynamický objekt. Základňa páčky je navrhnutá tak, aby samotná páčka sa dala otáčať iba v dovolenom rozsahu. Na Obr. 8.8 je znázornená táto páčka s vyznačenými kolíznymi mapami, pre lepšiu ilustráciu jej funkcie. Rozdiel vo fungovaní vo VR móde je taký, že namiesto páčky má používateľ k dispozícii virtuálnu ruku, ktorými môže fyzicky posunúť páku do správnej polohy, pričom samotný pohyb by sa mal blížiť čo najviac realite.



Obr. 8.8: Fungovanie komponentu páčky

Použitie v scenári

Tento komponent sa používa na interaktívne otvorenie dverí do miestnosti *Usporiadanie* (časť 7.2).

8.2.4 Komponent *enable-text-on-marking*

Komponent *enable-text-on-marking* umožňuje zobrazenie textového poľa v scéne na základe prijatej hodnoty tokenov (*marking*) v mieste v Petriho sieti.

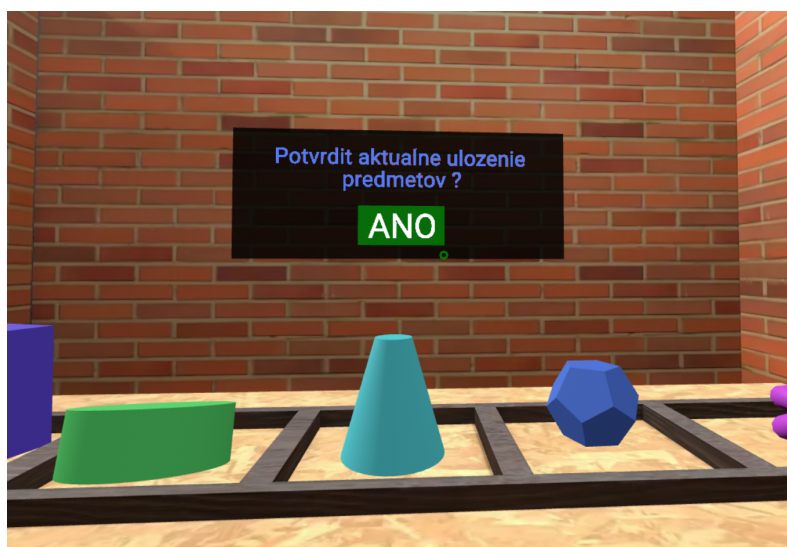
Parametre komponentu

- **yPosition** - Požadovaná výška objektu pri aktivácii.
- **marking** - Číselná hodnota tokenov, ktorá určuje, kedy sa má objekt zobraziť.

Spracovanie udalostí

Komponent reaguje na udalosť `getPlaceMarking`, ktorá obsahuje číselnú hodnotu množstva tokenov. Ak táto hodnota zodpovedá hodnote parametra `marking`, objekt sa zobrazí a jeho výška sa upraví na `yPosition` (slúži na prevenciu nechceného kliknutia, keď je komponent neviditeľný). Ak sa značka nezhoduje, objekt sa skryje a vráti sa na pôvodnú pozíciu.

Použitie v scenári



Obr. 8.9: Ukážka komponentu *enable-text-on-marking*

Tento komponent sa používa v miestnosti *Usporiadanie* na potvrdenie usporiadania predmetov na stole, pred overením ich správnosti.

8.2.5 Komponent *text-marking-update*

Komponent `text-marking-update` slúži na dynamickú aktualizáciu textového obsahu na základe počtu tokenov v mieste (*marking*) získaného z udalosti. Umožňuje tiež vyvolať prechod a po jeho aktivácii zmeniť ďalšie atribúty.

Parametre komponentu

- **initialMarking** - Počiatočná hodnota informačnej tabule.
- **updateEvent** - Udalosť, po ktorej sa aktualizuje hodnota na tabuli.
- **reversed** - Označuje, či sa hodnota pripočítava, alebo odpočítava.
- **finalMarking** - Hodnota označujúca hraničnú hodnotu informačnej tabule.
- **fireTransition** - Označuje, či sa po dosiahnutí hraničnej hodnoty má odpáliť prechod v Petriho sieti.
- **setAttributeAfterFire** - Zoznam atribútov, ktoré sa majú zmeniť po odpálení prechodu. Každá položka zoznamu obsahuje názov atribútu a novú hodnotu vo formáte atribút->hodnota.

Spracovanie udalostí

- **updateEvent** - Pri prijatí udalosti sa získaná hodnota tokenov (*marking*) buď priamo nastaví ako aktuálna zobrazená hodnota, alebo sa vypočíta ako rozdiel medzi počiatočnou hodnotou a prijatou hodnotou, ak je aktivovaná možnosť *reversed*. Táto hodnota sa zobrazí v textovej reprezentácii prvku.
- **fireTransition** - Ak je povolené *fireTransition* a označenie dosiahne hodnotu *finalMarking*, komponent vyšle udalosť *fireTransition* na odpálenie prechodu v Petriho sieti.
- **transitionFired** - Po úspešnom odpálení prechodu prijatím tejto udalosti sa zmenia atribúty objektu podľa zoznamu *setAttributeAfterFire*.

Použitie v scenári

Komponent sa využíva v každej miestnosti na zobrazenie aktuálneho stavu riešenej úlohy, ako je napríklad počet nahraných bodov v miestnosti *Hádzanie*, počet zostavajúcich sekúnd v miestnosti *Usporiadanie* a počet rozdelených predmetov v miestnosti *Triedenie*. Použitie komponentu je možné vidieť na obrázkoch 7.3, 7.4, alebo 7.13

8.2.6 Komponent *timer*

Komponent *timer* slúži na odpočítavanie času a vysielanie udalostí v závislosti od priebehu merania. Čas sa zobrazuje v definovanom prvku a umožňuje zastaviť, resetovať a nastaviť limit času.

Parametre komponentu

- **timeLimit** - Limit času v sekundách.
- **timeStep** - Veľkosť kroku časovača v sekundách.
- **displayElementId** - ID prvku, ktorý zobrazuje aktuálnu hodnotu času.

Spracovanie udalostí

- **transitionFired** - Po odpálení prechodu sa spustí časovač a zobrazí sa aktuálna hodnota času.
- **updateTime** - Po prijatí tejto udalosti sa časovač aktualizuje o hodnotu **timeStep**.
- **timeLimitReached** - Po dosiahnutí limitu času sa vyšle táto udalosť elementu **displayElementId**.
- **stopTimer** - Po prijatí tejto udalosti sa časovač zastaví.
- **resetTimer** - Po prijatí tejto udalosti sa časovač resetuje na počiatočnú hodnotu.

Použitie v scenári

Tento komponent sa využíva v miestnosti *Usporiadanie* na odpočítavanie zvyšného času do ukončenia úlohy. Po dosiahnutí limitu času sa zobrazí informačné okno s výsledkom aktivity.

8.2.7 Komponent *show-finish-info*

Komponent *show-finish-info* slúži na zobrazenie informačnej správy o výsledku aktivity po jej dokončení. Používateľovi sa zobrazí text s farebným označením úspešnosti a interaktívne tlačidlo na potvrdenie.

Spracovanie udalostí

Pri prijatí tejto udalosti komponent získa informáciu o úspešnosti (**isSuccessful**) a názve aktivity (**activityName**). Na základe týchto údajov sa zobrazí textová správa:

- Ak bola aktivita úspešná, text sa nastaví na „Úspešne si dokončil aktivitu [názov]“ a jeho farba bude zelenomodrá.

- Ak bola aktivita neúspešná, text sa nastaví na „Neúspešne si dokončil aktivitu [názov]“ a jeho farba bude červená.

Zároveň sa aktivuje tlačidlo (`finishButtonEl`), ktoré umožní používateľovi skryť informačné okno. Po kliknutí na tlačidlo sa odstráni jeho interaktívna trieda a celé informačné okno sa skryje.

Použitie v scenári

Tento komponent sa využíva na informovanie používateľa o výsledku jeho aktivity. Po úspešnom alebo neúspešnom dokončení úlohy sa zobrazí správa s príslušnou farebnou indikáciou. Popoužívateľ môže správu potvrdiť kliknutím na tlačidlo, čím sa vráti späť do scenára. Príklad použitia komponentu v miestnosti usporiadanie je zobrazený na Obr. 8.10.



Obr. 8.10: Ukážka komponentu *show-finish-info*

8.2.8 Ďalšie komponenty v scéne

Zvyšné komponenty vytvorené pre ukážkovú scénu, ako už bolo spomínané, nie sú univerzálne a dajú sa použiť iba v rámci tejto, alebo veľmi podobnej scény. Slúžia hlavne ako praktický príklad, ako je možné vytvoriť rôzne interakcie a úlohy pre používateľa. Medzi tieto komponenty patria:

- *ball-generator* - Generuje lopty, po úspešnom naskenovaní predmetu v miestnosti *Hádzanie*.

- *thrown-ball-checker* - Riadi pridanie bonusového terča do scény po odpálení príslušného prechodu.
- *gradual-fade* - Riadi postupnú zmenu viditeľnosti otáznika pri vstupe do miestnosti *Triedenie* predmetov. Taktiež riadi zobrazenie klávesnice a vstupného poľa pre zadanie kódu.
- *update-input* - Aktualizuje hodnotu vstupného poľa po kliknutí na klávesnicu pred tajnými dverami. Taktiež odpáli prechod v sieti, aké je zadané heslo správne.

8.2.9 Zhrnutie

Pri tvorbe akcie v scéne, ktorá má spôsobiť zmeny v riadiacej Petriho sieti a následne vyvolať zmeny v scéne reagujúce na akciu je postup udalostí nasledovný:

1. Užívateľ vykoná akciu v scéne, ktorá je zaregistrovaná komponentom, ktorý reaguje na danú akciu (napríklad stlačenie tlačidla v komponente (časť 8.2.2)).
2. Akcia sa spracuje a odošle sa udalosť do komponentu *pn-transition-handler* (časť 8.1.7).
3. V tomto komponente sa rozhodne, o aký typ akcie ide a podľa toho sa vyberie názov prechodu na odpálenie, ktorý sa odošle do komponentu *petri-net-sim* (časť 8.1.8).
4. Ďalej v tomto komponente sa prechod odpáli v Petriho sieti, pričom podľa výsledného stavu a typu akcie sa zavolá metóda z modulu `sceneScript.ts`.
5. V module scenára sa podľa toho, pre ktorý prechod sa prechod odpálila, zavolá príslušná metóda v komponente *pn-transition-handler* (časť 8.1.7).
6. V tejto metóde sa odošle udalosť do posledného komponentu, ktorý po jej prijatí spôsobí zmenu v scéne (napríklad komponent dverí spustí animáciu otvárania a vymení navigačnú sieť (časť 8.2.2)).

9 Vyhodnotenie

Keďže cieľom navrhovaného riešenia bolo aj vytvoriť ucelený návod, respektíve tutoriál pre nových vývojárov, ktorí sa rozhodnú rozšíriť projekt *LirkisEduVePn*, text tejto diplomovej práce bol navrhnutý tak, aby slúžil aj ako dokumentácia riešenia, ktorá je dostatočne detailná a zrozumiteľná pre čitateľa. Na overenie efektívnosti vybraných kapitol, ktoré popisujú všetky aspekty návrhu riešenia bolo nutné vykonať určitú formu prieskumu medzi potenciálnymi vývojármi, ktorí sa so systémom ešte nestretli. Pre tento prieskum bola zvolená metóda SUS [28], ktorá predstavuje 10 otázok týkajúcich sa subjektívneho pocitu a hodnotenia užívateľov z použitia systému. V rámci tohto hodnotenia ale metóda SUS nie je použitá na hodnotenie použiteľnosti aplikácie, keďže implementované zmeny pre túto záverečnú prácu sú skôr komponenty a nástroje pre vývojárov, ale na hodnotenie častí textu samotnej diplomovej práce, ktoré by mali pomôcť popri ukážkovej scéne k lepšiemu pochopeniu systému tvorby nových scén. Tento spôsob prieskumu by mal najlepšie zachytiť dojem vývojárov pri čítaní tejto práce ohľadom ich chápania navrhnutého riešenia. Každá odpoveď na otázku je vyjadrená Likertovou škálou, kde stupeň jedna predstavuje možnosť úplne nesúhlasím a stupeň 5 úplne súhlasím. Týchto desať otázok malo nasledujúce znenie:

1. Myslím si, že by som projekt používal(a) často.
2. Popis systému bol zbytočne komplikovaný a neprehľadný.
3. Myslím si, že popis projektu bol pre mňa ľahko zrozumiteľný.
4. Myslím si, že na pochopenie systému by som potreboval(a) pomoc niekoho iného.
5. Myslím si, že rôzne funkcie tohto systému sú dobre integrované.
6. Myslím si, že tento systém obsahuje príliš veľa nekonzistencií.
7. Dokážem si predstaviť, že väčšina ľudí by sa tento systém naučila používať v rozumnom čase.

8. Systém sa mi zdal veľmi ťažkopádny na používanie.
9. Pri opise systému som sa cítil(a) veľmi sebaisto.
10. Musel(a) som sa naučiť veľa vecí, kým by som mohol(mohla) začať pracovať s týmto systémom.

Súčasťou prieskumu boli aj dve kontrolné otázky, ktoré mali overiť, či respondent skutočne pochopil princíp nových komponentov a procesu vytvárania nových scén, ktoré sú najdôležitejšou súčasťou návrhu riešenia. Tieto otázky mali nasledujúce znenie:

1. Popíšte v skratke, ako by ste postupovali, keby ste chceli do projektu pridať novú scénu, vrátane 3D modelov, scenára a scény.
2. Vysvetlite ako a kedy by ste použili komponent ammo-collision-transition v rámci vlastnej scény.

Pre čo najlepšiu reprezentáciu výsledkov prieskumu boli vybraní respondenti, ktorí majú pokročilé znalosti v oblasti IT, ale nestretli sa ešte s týmto projektom, a ani s vývojom interaktívnych scén pre VR pomocou nástroja A-Frame. Ako veková skupina boli zvolení respondenti vo veku 20-30 rokov, ktorí predstavujú potencionálnych budúcich vývojárov v rámci projektu *LirkisEduVePn*. Takto navrhnutý prieskum by mal overiť nie len efektívnosť návrhu riešenia, ale aj jeho zrozumiteľnosť a jednoduchosť pre nových vývojárov.

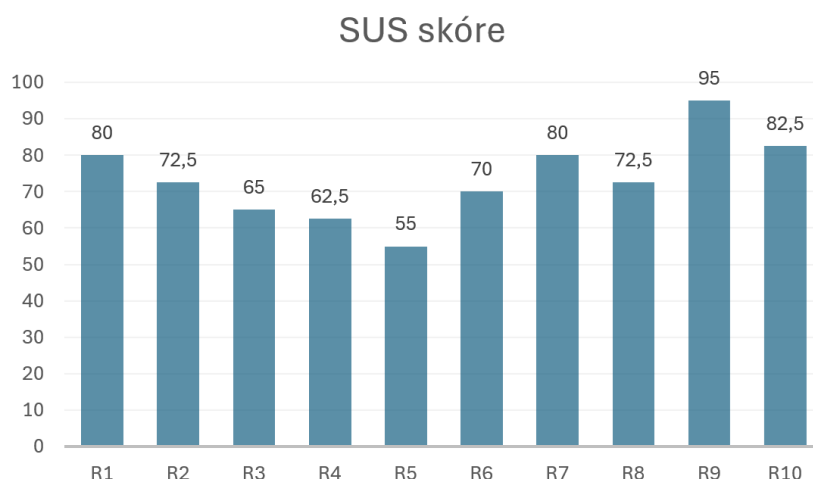
9.1 Výsledky prieskumu

V tejto časti sú uvedené výsledky prieskumu, ktorý bol vykonaný medzi 10 potenciálnymi vývojármi, ktorí sa so systémom ešte nestretli. Už nižší počet respondentov, ktorí boli ochotní sa prieskumu zúčastniť naznačuje, že systémy na vytváranie interaktívnych scén pre VR sú často ťažkopádne na pochopenie, čo veľa ľudí odrádza od ich používania. Napriek tomu ale tento prieskum priniesol zaujímavé výsledky, ktoré sú popísané nižšie.

9.1.1 Výsledky SUS

Ako bolo prezentované v úvode tejto kapitoly, každá z desiatich kladených otázok SUS časti dotazníka je vyjadrená Likertovou škálou, kde stupeň jedna predstavuje

možnosť úplne nesúhlasím a stupeň 5 úplne súhlasím. Pre každú odpoveď je potom vypočítaný súčet bodov, používaný na vyjadrenie hodnoty SUS, ktorá predstavuje použiteľnosť systému. Výsledky SUS hodnotenia sú zobrazené v tabuľke 9.1.



Obr. 9.1: Štruktúra projektového adresára

Z výsledkov SUS hodnotenia je zrejmé, že text tejto diplomovej práce je pomerne zrozumiteľný a nápomocný ku pochopeniu systému, čo vyjadruje aj priemerné dosiahnuté skóre 73,5. Na druhej strane, traja respondenti dosiahli skóre menšie, ako hodnota 68, považovaná za hranicu dobrej použiteľnosti a pochopenia systému, alebo v tom to prípade textu skúmaných kapitol. Z toho vyplýva, že stále existuje priestor na zlepšenie zrozumiteľnosti opisu systému aj napriek jeho vysokej zložitosti.

9.1.2 Výsledky kontrolných otázok

V tejto časti sú uvedené výsledky kontrolných otázok, ktoré mali overiť, či respondenti pochopili princíp nových komponentov a procesu vytvárania nových scén, ktoré sú najdôležitejšou súčasťou návrhu riešenia. Keďže odpovede na tieto otázky boli vo forme textu, každá odpoveď bola hodnotená zvlášť, pričom bola zohľadnená správnosť a úplnosť odpovede. Každá odpoveď bola vo výsledku hodnotená pozitívne, ak respondent dokázal pochopiť systém a naopak. Na obidve otázky dokázali správne odpovedať všetci respondenti, pričom rozdiel medzi nimi bol iba v zhodnosti odpovede so skutočným fungovaním systému. V dvoch prípadoch boli odpovede správne, no stručnejšie, než bolo očakávané, čo môže naznačovať slabšie pochopenie systému, ale aj nezaujím o návrh riešenia prezentovaný v tejto záverečnej práci. Respondenti, ktorí sa vyjadrili takýmto spôsobom

dosiahli nižšie hodnoty SUS, čo potvrdzuje, že opis systému je pre nich príliš zložitý a neprehľadný a potvrdzuje to hypotézu v predošlej časti tejto kapitoly, že opis systému je možné ďalej upraviť pre dosiahnutie väčšieho pochopenia.

10 Záver

Táto diplomová práca sa zaoberá rozšírením systému virtuálnych výučbových prostredí, ktorý je navrhnutý pomocou rámca A-Frame a riadený Petriho sieťami. Hlavným cieľom práce bolo upraviť existujúce komponenty, vytvoriť sadu nových komponentov a ukážkovú scénu, ktorá slúži ako príklad a návod pre vývojárov pri tvorbe nových scén, ale aj prepracovať spôsob interakcie s Petriho sieťou. Práca sa zameriava hlavne na zvýšenie interaktivity a zjednodušenie procesu vývoja, čo má za následok efektívnejšie a pútavejšie edukačné prostredia pre žiakov základných, či stredných škôl.

V teoretickej časti práce je analyzovaný súčasný stav projektu, vrátane jeho limitácií, ako sú nedostatok univerzálnych komponentov, neprehľadný kód a obmedzené možnosti interakcie. Ďalej je prezentovaný systematický prehľad vedeckých prác, ktoré skúmajú využitie virtuálnej reality a seriózných hier vo vzdelávaní. Tieto analýzy viedli k odpovediam na predom definované výskumné otázky a identifikácií kľúčových prvkov, ktoré by mali byť súčasťou interaktívnych edukačných hier.

V praktickej časti práce je navrhnutá a implementovaná ukážková scéna, ktorá pozostáva z troch častí, každá s iným typom úlohy. Tieto úlohy demonštrujú možnosti nových komponentov, ako sú napríklad fyzikálny engine Ammo.js, prepracovaná klávesnica pre VR, komponent na detekciu kolízií, rôzne typy dverí, informačné tabule s aktuálnym stavom scény a mnoho ďalších. Dôležitou súčasťou práce je aj úprava a vylepšenie existujúcich komponentov, ako sú Blink controls alebo Super Hands, aby boli kompatibilné s novými technológiami a poskytovali lepší používateľský zážitok.

Výsledkom práce je kompletná ukážková scéna, ktorá integruje všetky nové funkcionality a slúži ako vzor pre budúci vývoj. Taktiež text tejto záverečnej práce bol písaný, ako dokumentácia, ktorá má pomôcť vývojárom pri tvorbe vlastných scén. Vyhodnotenie práce preukázalo, že navrhnuté riešenie je zrozumiteľné a použiteľné, aj keď existuje priestor na ďalšie vylepšenia pre lepšie pochopenie problematiky, najmä pre menej skúsených vývojárov.

Literatúra

1. WICKENS, Christopher D. Virtual reality and education. In: *Proceedings of the 1992 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 1992, zv. 1, s. 842–847. Dostupné z doi: 10.1109/ICSMC.1992.271688.
2. SÁNCHEZ, Ángel; BARREIRO, Jesús María; MAOJO, Víctor. Design of Virtual Reality Systems for Education: A Cognitive Approach. *Education and Information Technologies*. 2000, roč. 5, č. 4, s. 345–362. ISSN 1573-7608. Dostupné z doi: 10.1023/A:1012061809603.
3. ELMQADDEM, Nabil. Augmented Reality and Virtual Reality in Education. Myth or Reality? *International Journal of Emerging Technologies in Learning (iJET)*. 2019, roč. 14, č. 03, s. 234–242. Dostupné z doi: 10.3991/ijet.v14i03.9289.
4. KAVANAGH, Shane; LUXTON-REILLY, Andrew; WÜNSCHE, Burkhard; PLIMMER, Bernhard. A Systematic Review of Virtual Reality in Education. *Themes in Science and Technology Education*. 2017, roč. 10, č. 2, s. 85–119. ISSN 1792-8788. Dostupné tiež z: <https://www.learntechlib.org/p/182115>.
5. DAMAŠEVIČIUS, Robertas; SIDEKERSKIENE, Tatjana. Virtual worlds for learning in metaverse: a narrative review. *Sustainability*. 2024, roč. 16, č. 5, s. 2032.
6. DENGEL, Andreas; IQBAL, Muhammad Zahid; GRAFE, Silke; MANGINA, Eleni. A Review on Augmented Reality Authoring Toolkits for Education. *Frontiers in Virtual Reality*. 2022, roč. 3. Dostupné z doi: 10.3389/frvir.2022.798032. Cited by: 27; All Open Access, Gold Open Access.
7. KAŠELA, Adam. *Výučbové scenáre v rozšírenej realite využívajúce procesné grafy*. Košice, 2022. Dipl. pr. Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky. Vedúci práce PhD. ING. ŠTEFAN KOREČKO.
8. DEMIANENKO, Dmytro. *Spracovanie údajov z virtuálnych výučbových prostredí*. Košice, 2023. Bakalárska práca. Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky. Vedúci práce PhD. ING. ŠTEFAN KOREČKO.

9. PISARČÍK, Lukáš. *Scenárom riadené výučbové prostredia vo virtuálnej realite*. Košice, 2023. Bakalárska práca. Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky. Vedúci práce PhD. ING. ŠTEFAN KOREČKO.
10. CIBULA, Marián. *Scenárom riadené virtuálne výučbové prostredie pre geografiu*. Košice, 2024. Bakalárska práca. Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky. Vedúci práce PhD. ING. ŠTEFAN KOREČKO.
11. RUDY, Juraj. *Systém virtuálnych výučbových prostredí riadených procesnými grafmi*. Košice, 2024. Dipl. pr. Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky. Vedúci práce PhD. ING. ŠTEFAN KOREČKO.
12. BRŮŽA, Mgr Vojtěch. *Education in Collaborative Immersive Virtual Reality*. [B.r.].
13. APANASTASIOU, Georgia; DRIGAS, Athanasios; SKIANIS, Charilaos; LYTRAS, Miltiadis; PAPANASTASIOU, Evangelia. Virtual and Augmented Reality Effects on K-12, Higher and Tertiary Education Students' Twenty-First Century Skills. *Virtual Reality*. 2019, roč. 23, č. 4, s. 425–436. ISSN 1434-9957. Dostupné z DOI: 10.1007/s10055-018-0363-2.
14. FREINA, Laura; OTT, Michela. A Literature Review on Immersive Virtual Reality in Education: State Of The Art and Perspectives. In: *Proceedings of the eLearning and Software for Education (eLSE)*. 2015.
15. WU, Jiajie; CHEN, Der-Thanq. A systematic review of educational digital storytelling. *Computers & Education*. 2020, roč. 147, s. 103786. ISSN 0360-1315. Dostupné z DOI: 10.1016/j.compedu.2019.103786.
16. LUCARDIE, Dirk. The impact of fun and enjoyment on adult's learning. *Procedia - Social and Behavioral Sciences*. 2014, roč. 142, s. 439–446. Dostupné z DOI: 10.1016/j.sbspro.2014.07.696.
17. SMETANA, Lana; BELL, Randy. Computer simulations to support science instruction and learning: A critical review of the literature. *International Journal of Science Education*. 2012, roč. 34, č. 9, s. 1337–1370. Dostupné z DOI: 10.1080/09500693.2011.605182.
18. PLASS, Jan L.; HOMER, Bruce D.; KINZER, Charles K. Foundations of Game-Based Learning. *Educational Psychologist*. 2015, roč. 50, č. 4, s. 258–283. Dostupné z DOI: 10.1080/00461520.2015.1122533.
19. BLIPPBUILDER. *Augmented Reality Creation Technology — Blippbuilder*. 2021. Dostupné na: <https://www.blippar.com/build-arDataset> (Navštívené 10.8.2024).

20. SCAVARELLI, Anthony; ARYA, Ali; TEATHER, Robert J. Virtual reality and augmented reality in social learning spaces: a literature review. *Virtual Reality*. 2021, roč. 25, č. 1, s. 257–277. Dostupné z [doi: 10.1007/s10055-020-00444-8](https://doi.org/10.1007/s10055-020-00444-8). Cited by: 183.
21. BOWMAN, Doug A.; HODGES, Larry F. Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *Journal of Visual Languages and Computing*. 1999, roč. 10, s. 37–53.
22. JR., Joseph J. LaViola; KRUIJFF, Ernst; MCMAHON, Ryan P.; BOWMAN, Doug A.; POUPYREV, Ivan. *3D User Interfaces: Theory and Practice (Usability)*. 2nd. Boston: Addison-Wesley, 2017.
23. BOYTCHEV, Pavel; BOYTCHEVA, Svetla. Gamified evaluation in STEAM for higher education: A case study. *Information*. 2020, roč. 11, č. 6, s. 316.
24. BEEVER, Lee. Exploring Mixed Reality Level Design Workflows. 2023.
25. PIRKER, Johanna; HOLLY, Michael; ALMER, Hannes; GÜTL, Christian; BELCHER, John Winston. Virtual reality STEM education from a teacher's perspective. In: *iLRN 2019 London, Workshop, Long and Short Paper, and Poster Proceedings: from the Fifth Immersive*. 2019.
26. MYSTAKIDIS, Stylianos; THEOLOGI-GOUTI, Penelope; CHRISTOPOULOS, Athanasios; STYLIOS, Chrysostomos. Virtual Museum Gamification for Discovery-Based Online Learning in the Metaverse. In: 2024, zv. 1, s. 711–719. Dostupné z [doi: 10.5220/0012756600003693](https://doi.org/10.5220/0012756600003693). Cited by: 0; All Open Access, Hybrid Gold Open Access.
27. FERGUSON, Chris; BROEK, Egon L. van den; OOSTENDORP, Herre van. On the role of interaction mode and story structure in virtual reality serious games. *COMPUTERS & EDUCATION*. 2020, roč. 143. ISSN 0360-1315. Dostupné z [doi: 10.1016/j.compedu.2019.103671](https://doi.org/10.1016/j.compedu.2019.103671).
28. LEWIS, James R. The System Usability Scale: Past, Present, and Future. *International Journal of Human-Computer Interaction*. 2018, roč. 34, č. 7, s. 577–590. Dostupné z [doi: 10.1080/10447318.2018.1455307](https://doi.org/10.1080/10447318.2018.1455307).

Slovník

Fyzikálny engine je softvérová knižnica alebo systém, ktorý simuluje fyzikálne javy v reálnom čase alebo offline. V 3D grafike a hrách sa používa na simuláciu pohybu, kolízií, gravitácie, trenia a ďalších fyzikálnych interakcií medzi objektmi.

Pečenie osvetlenia (light baking) v 3D grafike spočíva v predpočítaní osvetlenia scény a jeho uloženie do textúr alebo svetelných máp (lightmaps). Tieto textúry sa potom aplikujú na objekty v scéne, čím sa dosiahne realistický vzhľad osvetlenia bez nutnosti dynamických výpočtov pri vykresľovaní.

Raycaster je technika používaná v počítačovej grafike a hrách na detekciu kolízií medzi lúčmi a objektmi v 3D prostredí. Raycaster vysiela "lúče" z určitého bodu (napríklad kamery) a zisťuje, s ktorými objektmi sa tieto lúče stretávajú.

Zoznam príloh

A Používateľská príručka

Používateľská príručka, podobne ako táto práca, vychádza z výskumov a publikácií Dmytra Demianenka [8], Lukáša Pisarčíka [9], Adama Kašelu [7], Juraja Rudyho [11] a Mariána Cibulu [10]. Časti príručky, ktoré naša práca neovplyvnila, zostali v nezmenenej podobe.

A.1 Funkcia programu

Táto práca je zameraná na návrh a implementáciu systému pre analýzu dát v prostredí rozšírenej reality (XR), ktorý má tri hlavné funkcie: (1) uľahčenie vykonávania úloh pre žiakov, (2) poskytnutie kontroly nad úlohami pre učiteľov a (3) podporu administratívnych procesov. Systém je navrhnutý tak, aby zaznamenával údaje o používateľoch počas riešenia úloh a následne generoval detailné štatistické výstupy. Vďaka vstavaným analytickým nástrojom ponúka okamžitý prehľad o výkone a personalizovanú spätnú väzbu, čo prispieva k zvýšeniu efektivity vzdelávacieho procesu.

A.2 Spustenie a inštalácia programu pre administrátorov

Táto kapitola obsahuje popis produktu, vrátane jeho funkčných vlastností a detailnej špecifikácie technických aj softvérových požiadaviek. Ďalej poskytuje podrobný návod na inštaláciu, a to z pohľadu oboch typov používateľov - administrátora aj koncového používateľa.

A.2.1 Požiadavky na technické prostriedky

Na spustenie projektu musia byť splnené tieto minimálne hardvérové požiadavky pre WebGL, a to:

- 64-bitový operačný systém

- 4 GB pamäte RAM
- Integrovaná grafika s podporou WebGL 2.0 OpenGL ES 3.0
- Dvojjadrový procesor Intel/AMD
- Najnovšia verzia prehliadača Chrome, Firefox alebo Opera

A.2.2 Spustenie projektu

Táto časť detailne popisuje pracovné postupy pre dvoch typov používateľov:

- Pre bežných používateľov: spustenie projektu prostredníctvom webového prehliadača
- Pre administrátorov: proces nasadenia projektu na serverovú infraštruktúru

Otvorenie projektu pre používateľov

Prístup k projektu je umožnený všetkým používateľom pripojeným na sieť VPN TUKE alebo Wi-Fi TUKE. Projekt je dostupný prostredníctvom ľubovoľného webového prehliadača na adrese:

`http://147.232.205.222:4200`

Na tejto adrese sa používatelia môžu zaregistrovať (pre nových používateľov), alebo sa prihlásiť (pre existujúcich používateľov).

Nasadenie aplikácie pre administrátorov

Pre nasadenie projektu na server je potrebný nástroj Docker¹. Proces nasadenia pozostáva z nasledujúcich krokov:

1. Vytvorenie pracovného priečinka na serveri 147.232.205.222 s povinnými konfiguračnými súbormi:
 - `docker-compose.yaml` - hlavný konfiguračný súbor definujúci služby, siete a zväzky pre Docker kontajnery
 - `servers.json` - doplnkový konfiguračný súbor

¹Docker je open-source platforma umožňujúca kontajnerizáciu aplikácií a vytváranie izolovaných, prenosných prostredí. Podrobnejšie informácie sú dostupné na: <https://www.docker.com/>

2. Vytvorenie súboru `docker-compose.yml` slúži ako centrálny manažér pre viackontajnerové aplikácie, umožňujúci definíciu a konfiguráciu služieb, správu sieťových pripojení a prácu so zväzkami (volumes).

Príklad základnej konfigurácie `docker-compose.yml`:

```

6  version: '3.8'
7
8  services:
9
10 db-postgres:
11   image: "postgres:15.1"
12   container_name: lirkis-database
13   volumes:
14   | - lirkis-data:/var/lib/postgresql/data
15   ports:
16   | - "5432:5432"
17   environment:
18   | - POSTGRES_DB=postgres
19   | - POSTGRES_USER=postgres
20   | - POSTGRES_PASSWORD=postgres
21
22 pgadmin:
23   image: dpage/pgadmin4:7.1
24   container_name: lirkis-pgadmin
25   depends_on:
26   | - db-postgres
27   ports:
28   | - "3000:80"
29   environment:
30   | - PGADMIN_DEFAULT_EMAIL=admin@admin.com
31   | - PGADMIN_DEFAULT_PASSWORD=admin
32   volumes:
33   | - ./servers.json:/pgadmin4/servers.json
34
35 spring:
36   build: ./LirkisEduVePn-service
37   container_name: lirkis-service
38   ports:
39   | - "8443:8443"
40   links:
41   | - db-postgres
42
43 angular:
44   build: ./LirkisEduVePn-UI
45   container_name: lirkis-ui
46   ports:
47   | - "4200:443"
48   links:
49   | - spring
50
51 volumes:
52   lirkis-data:

```

Obr. A.1: Ukážka súboru docker-compose.yaml

V tomto súbore sú definované štyri služby: db-postgres, pgadmin, spring a angular. Nižšie je uvedený súbor servers.json, ktorý používa služba pgadmin na konfiguráciu pripojenia k serveru PostgreSQL:

```
1  {
2    "Servers": {
3      "1": {
4        "Name": "Local",
5        "Group": "Servers",
6        "Host": "db-postgres",
7        "Port": 5432,
8        "MaintenanceDB": "postgres",
9        "Username": "postgres",
10       "SSLMode": "prefer",
11       "SSLCert": "<STORAGE_DIR>/.postgresql/postgresql.crt",
12       "SSLKey": "<STORAGE_DIR>/.postgresql/postgresql.key",
13       "SSLCompression": 0,
14       "Timeout": 10,
15       "UseSSHTunnel": 0,
16       "TunnelPort": "22",
17       "TunnelAuthentication": 0
18     }
19   }
20 }
```

Obr. A.2: Ukážka súboru servers.json

A.2.3 Spustenie a konfigurácia systému

Po úspešnom vytvorení konfiguračných súborov je systém pripravený na spustenie. V závislosti od operačného systému použite príslušný príkaz:

- Pre Windows: `docker compose up`
- Pre Linux: `docker-compose up`

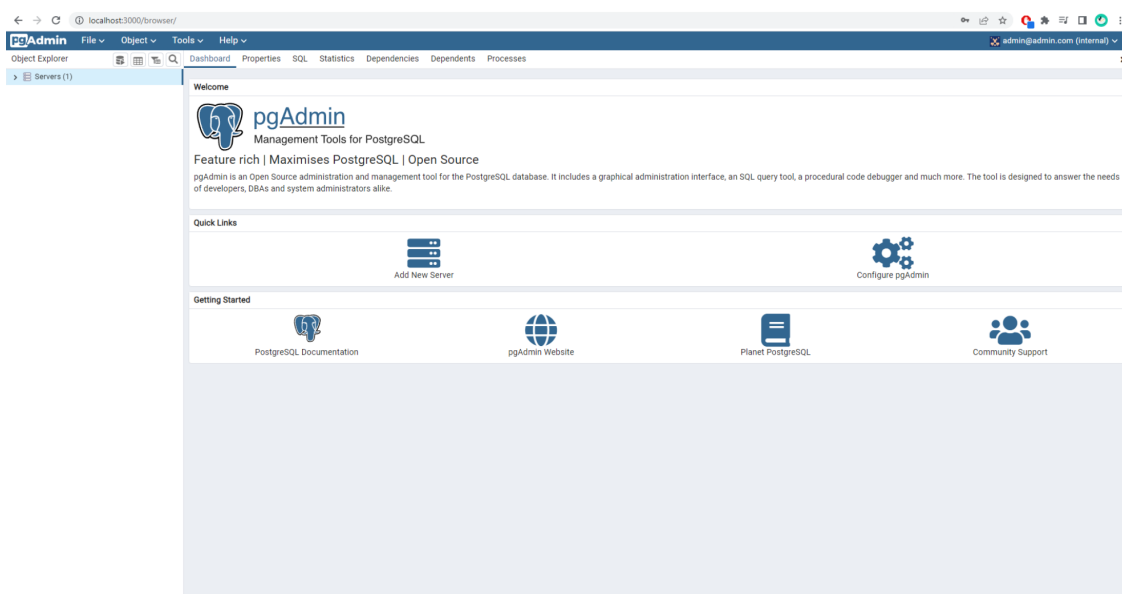
A.2.4 Prístup k aplikácii

Po štarte aplikácie je dostupná na adrese `https://147.232.205.222:4200` so všetkými funkciami identickými s používateľským režimom.

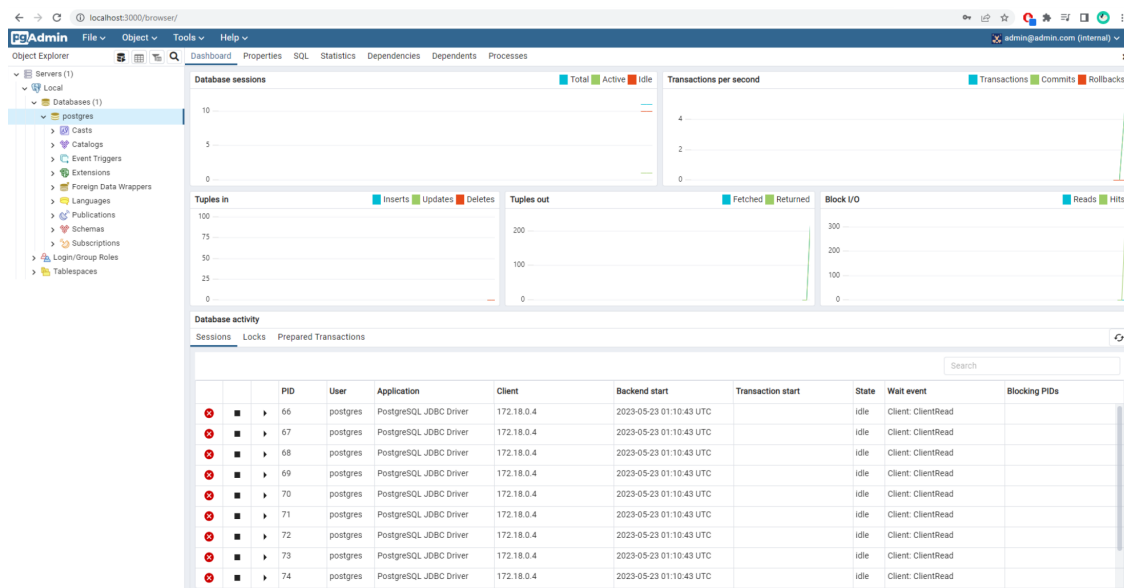
A.2.5 Manuálna konfigurácia administrátora

Kvôli obmedzeniam pri tvorbe databázy pomocou SQL skriptov je potrebné administrátorský účet pridať manuálne:

1. Prejdite na administračný panel databázy dostupný na adrese `http://147.232.205.222:3000`
2. Prihláste sa pomocou prihlasovacích údajov:
 - Email: `admin@admin.com`
 - Heslo: `admin`
3. V ovládacom paneli (pozri obrázok A.3) vyhľadajte našu databázu
4. Pri pokuse o prístup k databáze zadajte heslo: `postgres`
5. Úspešné pripojenie k databáze je znázornené na obrázku A.4



Obr. A.3: Okno so servermi v pgAdmin



Obr. A.4: Databáza v pgAdmin

A.2.6 Pridanie administrátorského účtu

Pre vytvorenie používateľa s administrátorskými oprávneniami vykonajte nasledovné kroky:

1. Spustíte SQL editor pomocou:
 - Kliknutia na tlačidlo Query Tool v hlavnom menu
 - Alebo klávesovej skratky **Alt + Shift + Q**
2. Do otvoreného editora zadajte nasledujúci SQL dotaz:

Zdrojový kód A.1: SQL dotaz na pridanie administrátora

```
INSERT INTO users(
    email, firstname, is_enabled,
    lastname, username, password, role
)
VALUES (
    'admin@admin', 'admin', true, 'admin', 'admin',
    's2a$12$3HMoKSwt7wbRayzlmoApieJTfvP0oqtc5ZTYmsPy04PSJoy9TeDmW',
    'ADMIN'
);
```

Po úspešnom vytvorení administrátorského účtu je možné vykonať prihlásenie do systému pomocou `admin@admin` a hesla `admin`, ktoré potom môžete zmeniť. Po autentifikácii bude môcť administrátor využívať všetky funkcie systému.

A.3 Použitie programu

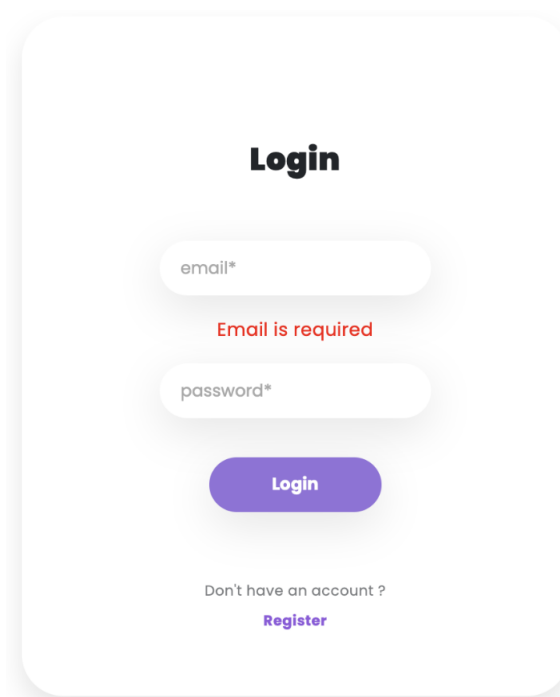
Navrhovaný systém poskytuje používateľom nasledujúce kľúčové možnosti:

- Interaktívne vykonávanie úloh v prostredí virtuálnej reality na základe preddefinovaných scenárov
- Editor scenárov umožňujúci vytváranie nových výukových scenárov, ako aj úpravu existujúcich scenárov.
- Historický prehľad všetkých vykonaných pokusov s možnosťou ich analýzy

V nasledujúcej časti sú podrobne popísané jednotlivé funkcie aplikácie a postup práce s nimi.

A.3.1 Prihlasovacia obrazovka

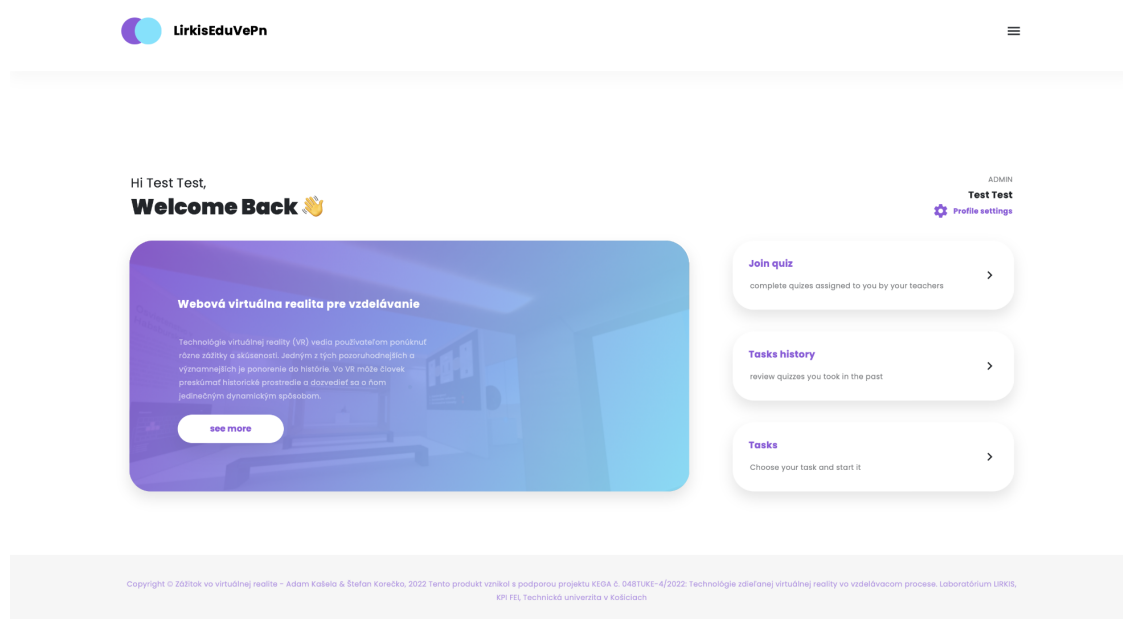
Pri spustení aplikácie sa používateľovi zobrazí prihlasovacia obrazovka (obrázok A.5), ktorá pozostáva z dvoch vstupných polí pre prihlasovacie údaje - používateľské meno a heslo - spolu s potvrdzovacím tlačidlom **Prihlásiť sa**. Noví používatelia môžu prejsť k registrácii účtu prostredníctvom príslušného odkazu umiestneného pod prihlasovacími poľami.



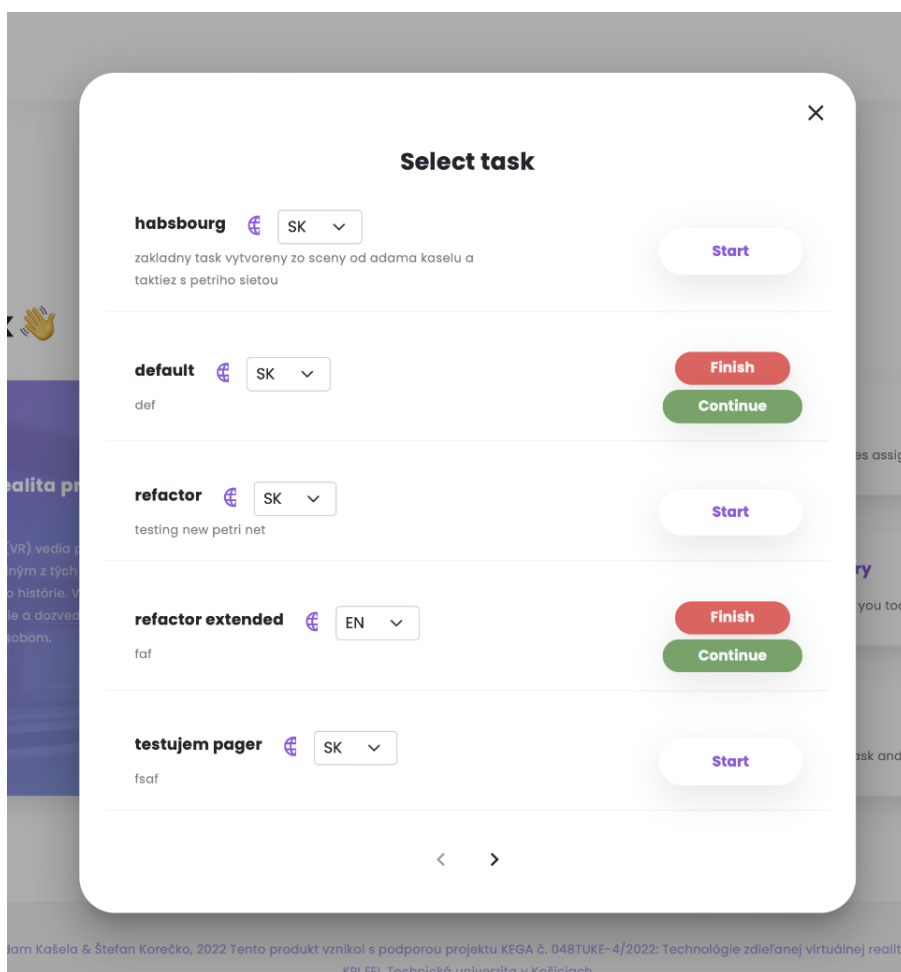
Obr. A.5: Prihlasovacie okno

A.3.2 Hlavná obrazovka

Hlavné menu žiaka bolo navrhnuté s dôrazom na prehľadnosť a funkčnosť. Centrálnym prvkom rozhrania je zoznam všetkých úloh pridelených učiteľom, doplnený o možnosť zobrazenia výsledkov dokončených úloh. Pri výbere konkrétnej úlohy má žiak k dispozícii niekoľko možností: výber jazyka rozhrania, pokračovanie v nedokončenej úlohe alebo jej definitívne ukončenie (obrázok A.7). Rozhranie ďalej ponúka prístup k profilovým nastaveniam, kde je možné upraviť údaje účtu. Celkové usporiadanie hlavnej obrazovky je znázornené na obrázku A.6.



Obr. A.6: Hlavná obrazovka



Obr. A.7: Výber úlohy

A.3.3 Dashboard pre učiteľov a adminov

Administračný dashboard bol navrhnutý ako komplexný nástroj pre správu vzdelávacieho obsahu a používateľského prostredia. Ponúka širokú škálu funkcií určených pre efektívnu organizáciu výučbového procesu.

Správa výučbového obsahu

Hlavnou funkciou dashboardu je možnosť tvorby a správy výučbových materiálov. Učitelia a administrátori môžu:

- Vytvárať nové scénare pomocou vizuálneho editora Petriho sietí doplneného o jazykové súbory s popismi
- Pripravovať jednotlivé scény ako štruktúrované výučbové jednotky
- Generovať úlohy pre študentov s možnosťou špecifikácie názvu, popisu a priradených scén

Správa používateľov a skupín

Rozhranie umožňuje komplexnú správu používateľských účtov a ich organizáciu do skupín. Administrátori majú k dispozícii:

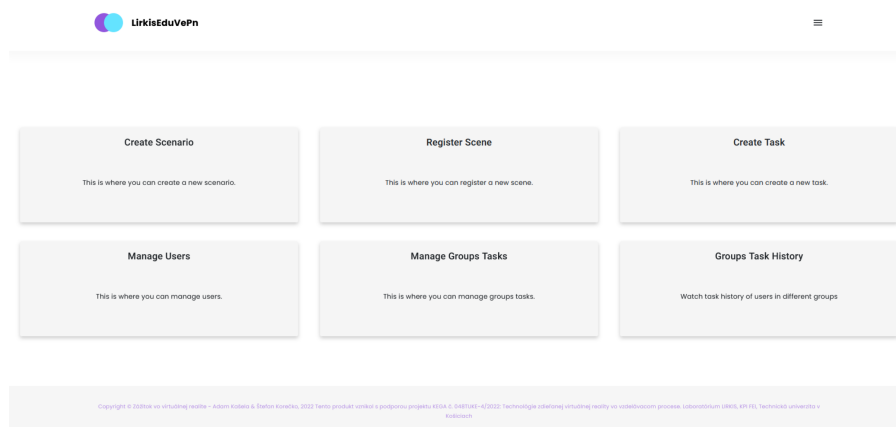
- Nástroje na vytváranie, úpravu a deaktiváciu používateľských účtov
- Možnosť priradovania používateľov do skupín podľa tried, kurzov alebo iných kritérií
- Systém na správu oprávnení a prístupových práv k obsahu

Monitorovanie a hodnotenie výkonu

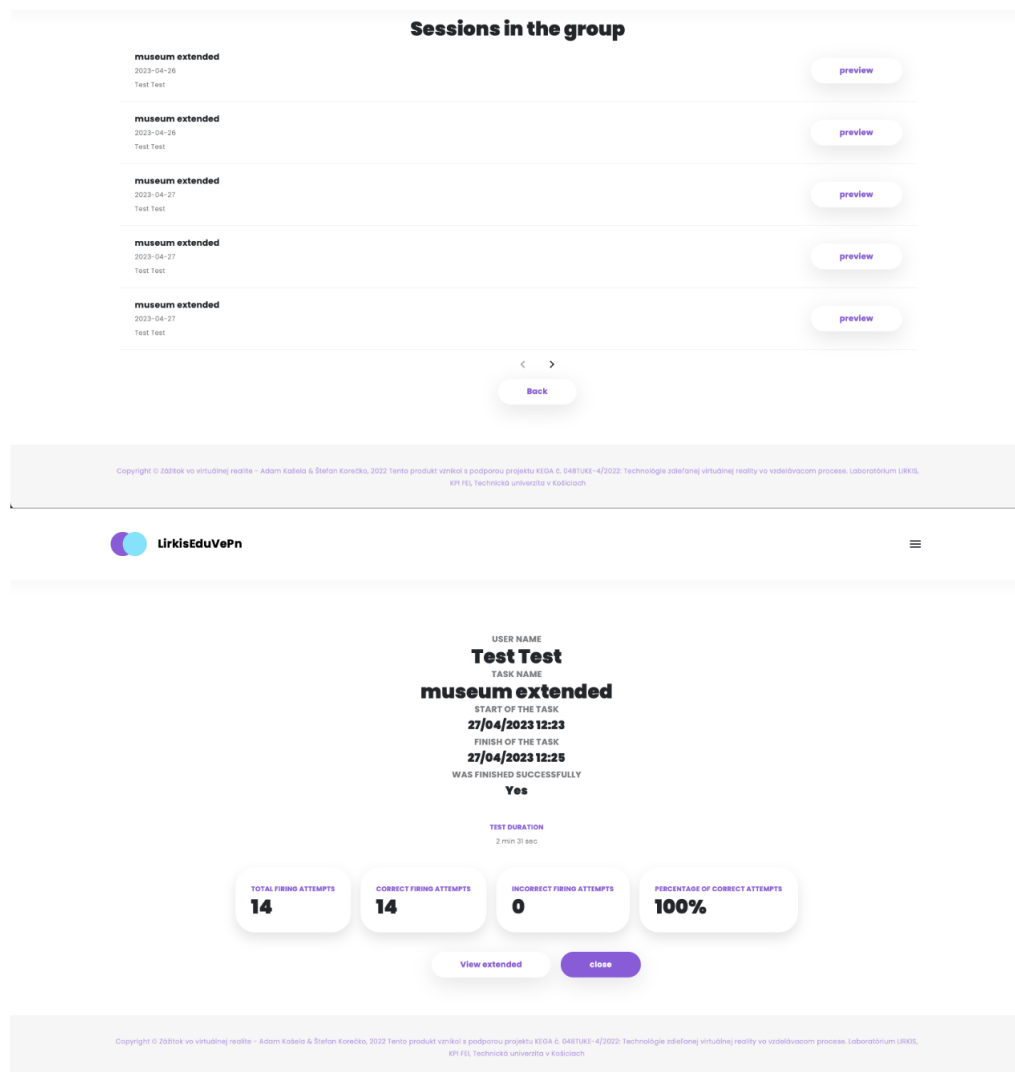
Dôležitou súčasťou dashboardu je systém na sledovanie a hodnotenie výkonu študentov. Ako je znázornené na obrázku A.9, rozhranie poskytuje:

- Prehľad histórie riešení úloh v rámci jednotlivých skupín
- Detailné štatistiky úspešnosti jednotlivých prechodov v Petriho sieťach
- Možnosť zobrazenia časových údajov o aktiváciách prechodov
- Nástroje na vizualizáciu vývojových trendov v skupinách

Prostredníctvom tlačidla Preview je možné zobrazíť súhrnné štatistiky pre konkrétnu úlohu, zatiaľ čo funkcia View extended ponúka detailný pohľad na jednotlivé aktivácie prechodov vrátane informácií o ich úspešnosti a časovom priebehu.



Obr. A.8: Používateľský dashboard



Obr. A.9: Prehľad histórie úloh

A.3.4 Virtuálna scéna

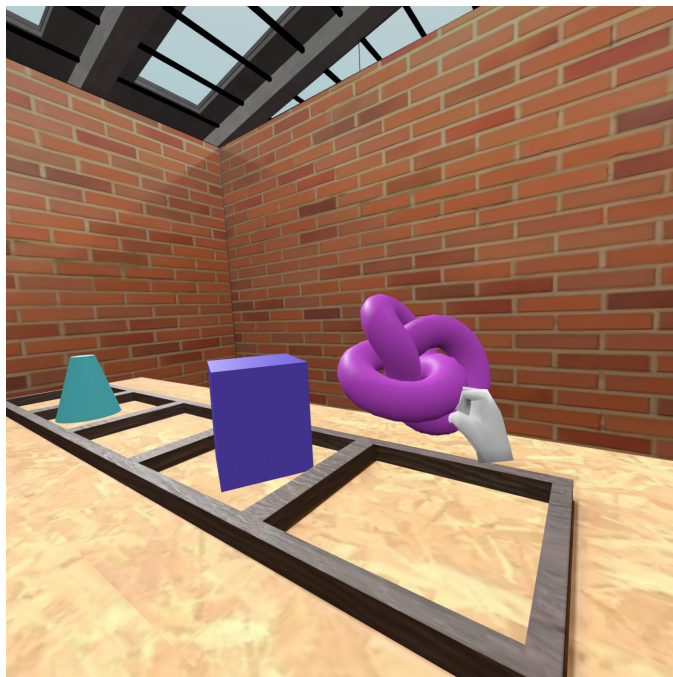
Virtuálna scéna predstavuje simulované prostredie vytvorené v rámci aplikácie, ktoré umožňuje interaktívnu prácu používateľov s výučbovým obsahom. Toto prostredie slúži ako hlavný pracovný priestor pre realizáciu vzdelávacích úloh a aktivít. Scéna je navrhnutá tak, aby podporovala široké spektrum interakcií nevyhnutných pre splnenie stanovených výučbových cieľov. Medzi kľúčové interaktívne prvky patria:

- Uchopenie objektov
- Hádzanie objektov
- Fyzické otváranie rôznych dverí
- Písanie na klávesnici

- Potvrdzovanie úlohy

A.3.5 Manipulácia s objektami

V novej ukážkovej miestnosti sú od poslednej verzie programu rozšírené možnosti, ako manipulovať s predmetmi. Predmety, ktoré sú súčasťou úlohy majú nastavenú fyziku, čiže okrem ich uchopenia je možné ich aj hádzať a umiestňovať na rôzne povrchy, ako je ilustrované na obrázku A.10.



Obr. A.10: Manipulácia s objektami v scéne

A.3.6 Otváranie dverí

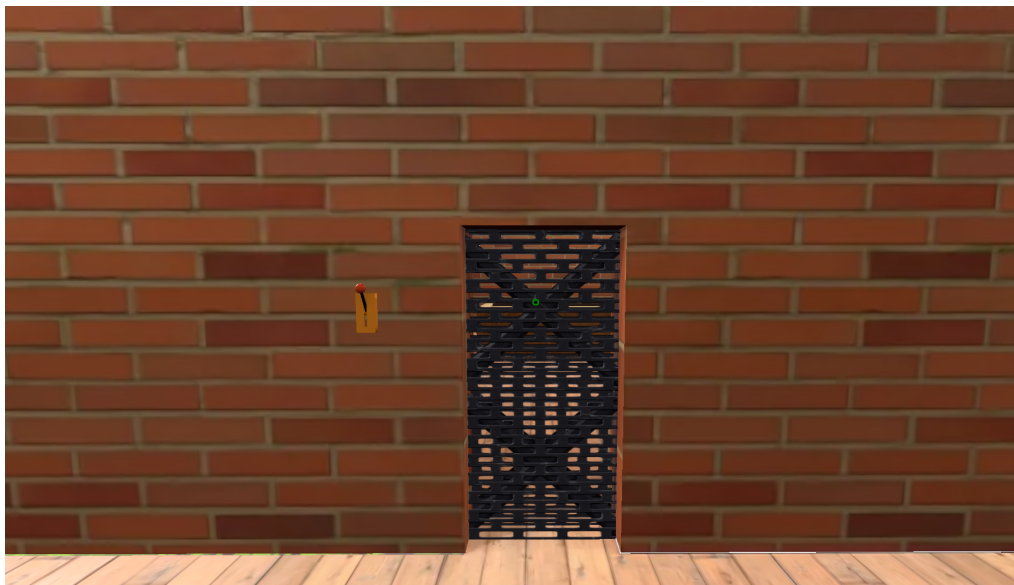
Na vstup do miestností a spustenie aktivít v miestnostiach je nutné otvoriť jedny z dverí. V projekte sa nachádzajú 3 typy dverí:

- **Dvere s tlačidlom** - Na otvorenie dverí je potrebné kliknúť, (alebo fyzicky stlačiť vo VR móde) tlačidlo umiestnené vedľa dverí.
- **Dvere s páčkou** - Na otvorenie dverí je potrebné myšou namieriť nad páčku (alebo uchopiť páčku vo VR móde) a potiahnuť ju smerom nadol.
- **Tajné dvere** - Na otvorenie dverí je nutné zadať tajné heslo do klávesnice, ktorá sa nachádza vedľa dverí.

Na nasledujúcich obrázkoch sú zobrazené jednotlivé typy dverí, ktoré sú v projekte implementované. Na obrázku A.11 sú zobrazené dvere s tlačidlom, na obrázku A.12 sú zobrazené dvere s páčkou a na obrázku A.13 sú zobrazené tajné dvere.



Obr. A.11: Dvere s tlačidlom



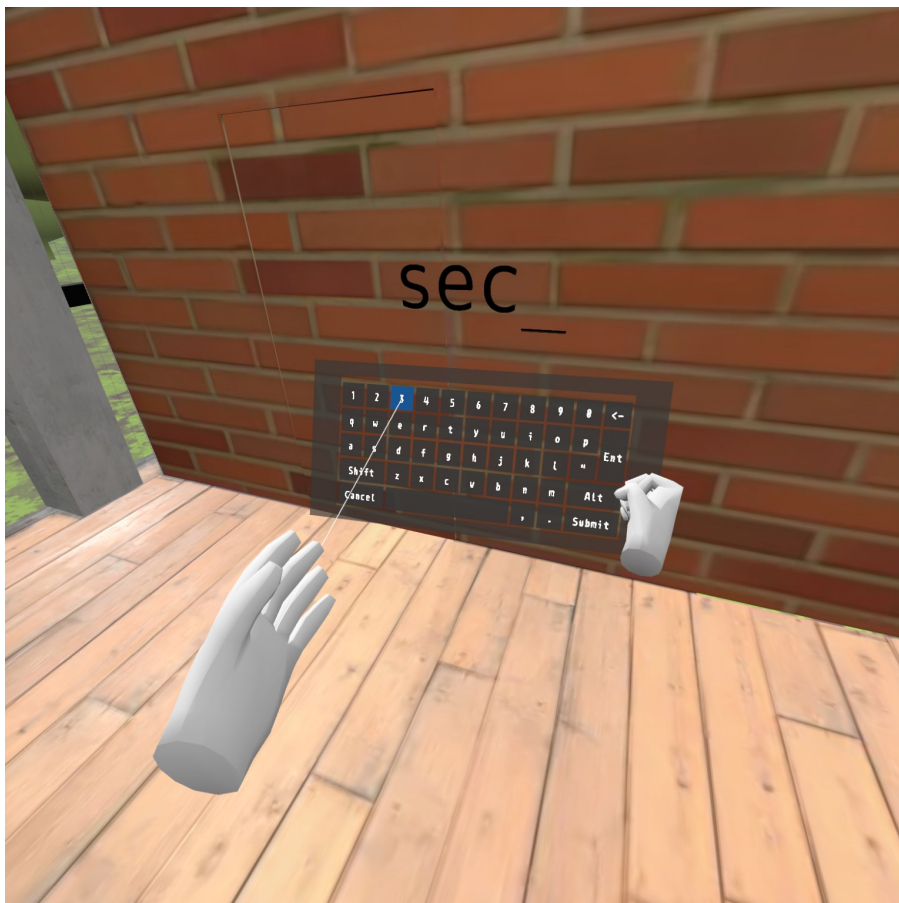
Obr. A.12: Dvere s páčkou



Obr. A.13: Tajné dvere

A.3.7 Písanie na klávesnici

Pri tretích dverách ukážkovej miestnosti je nutné zadať heslo pomocou klávesnice. Tá sa dá posúvať myšou, alebo uchopiť vo VR móde pravou rukou. Písanie je možné klikaním po klávesách, alebo pomocou raycastera vo VR móde. Na obrázku A.14 je zobrazená klávesnica, ktorá sa nachádza vedľa tajných dverí. Po zadaní správneho hesla sa dvere otvoria a používateľ môže pokračovať v úlohe.



Obr. A.14: Klávesnica

Ostatné scény v projekte ostali nezmenené a ich ovládanie je opísané v príručkách priložených ku predošlým prácam, ktoré aplikáciu *LirkisEduVePn* rozširovali.

B Systémová príručka

Systémová príručka, podobne ako táto práca, vychádza z výskumov a publikácií Dmytra Demianenka [8], Lukáša Pisarčíka [9], Adama Kašelu [7], Juraja Rudyho [11] a Mariána Cibulu [10]. Časti príručky, ktoré naša práca neovplyvnila, zostali v nezmenenej podobe.

B.1 Funkcia programu

Táto práca sa zameriava na vývoj systému analýzy údajov v rozšírenej realite (XR). Systém uľahčuje žiakom plnenie úloh, poskytuje učiteľom dohľad nad úlohami a ponúka administratívne funkcie. Jeho hlavným cieľom je zhromažďovať údaje o používateľoch počas plnenia úloh a generovať podrobné štatistiky o ich splnení. Integrované analytické nástroje umožňujú prehľad v reálnom čase a personalizovanú spätnú väzbu, čím prispievajú k zlepšeniu výsledkov vzdelávania.

B.2 Popis programu

Táto časť detailne popisuje základnú architektúru navrhnutého systému. Obsahuje popis všetkých implementovaných modelov, modulov a komponentov, vrátane dôležitých algoritmov a použitých premenných.

B.2.1 Popis riešenia

Systém je implementovaný pomocou Entity-Component-System (ECS) architektúry, ktorá poskytuje jasné oddelenie entít, komponentov a systémov spracovania. Táto architektúra bola rozšírená o webový framework Angular, ktorý zabezpečuje:

- Robustnú správu užívateľského rozhrania
- Efektívnu organizáciu komponentov a modulov

- Škalovateľnosť pre väčšie projekty

Kompletná štruktúra Java a Angular časti systému *LirkisEduVePn* je detailne popísaná v predchádzajúcej dokumentácii. V rámci tejto práce došlo k modifikácii komponentov zodpovedných za interakciu s 3D scénou, ktoré majú nasledujúcu štruktúru:



Obr. B.1: Štruktúra komponentov

B.3 Popis komponentov

V tejto časti sú opísané všetky novo pridané komponenty, ktoré sú zodpovedné za interakciu s 3D scénou. Tieto komponenty sú navrhnuté tak, aby umožnili používateľom efektívne manipulovať s objektmi v scéne a vykonávať rôzne akcie. Zvyšné komponenty, ktoré sa nachádzajú v projekte sú popísané v predošlých prácach, ktoré aplikáciu *LirkisEduVePn* rozširovali.

B.3.1 Aframe komponenty

- **autofitGltf.component.js** - Generuje kolízne mapy pre modely GLTF a inicializuje ich fyzikálne vlastnosti.
- **blinkControls.component.js** - Upravená verzia pre teleportáciu s prevenciou prechodu cez steny.
- **superHands-ammo.component.js** - Aktualizovaná verzia komponentu Super Hands pre kompatibilitu s Ammo.js.
- **aframeKeyboard.component.js** - Vylepšená virtuálna klávesnica s priestorovou manipuláciou a podporou raycasteru.
- **pnTransitionHandler.component.js** - Univerzálny komponent pre prechody v Petriho sieťach a spracovanie udalostí.
- **petriNetSim.component.js** - Upravený komponent, ktorý spravuje vykonávanie Petriho sietí a interakcie so scénou.
- **ammoCollisionTransition.component.js** - Spúšťa prechody pri kolíziách objektov.
- **buttonControl.component.js** - Spracúva stlačenie tlačidiel pre otváranie dverí a mechanizmov.
- **leverControl.component.js** - Spravuje interakcie s páčkou s fyzikálne simulovanou rotáciou.
- **enableText.component.js-on-marking** - Zobrazuje text na základe počtu tokenov v Petriho sieti.
- **textMarking-update.component.js** - Dynamicky aktualizuje text podľa stavu miest v Petriho sieti.

- **timerComponent.js** - Implementuje odpočítavanie času pre časovo obmedzené úlohy.
- **showFinishInfo.component.js** - Zobrazuje správy o úspechu/neúspechu po dokončení úlohy.
- **ballGenerator.component.js** - Generuje lopty pre úlohy s hádzaním.
- **gradualFade.component.js** - Riadi postupnú zmenu viditeľnosti objektov.
- **updateInput.component.js** - Spravuje textový vstup z virtuálnej klávesnice.

B.3.2 Typescript objekty

- **transitions.ts** - Definuje prechody v Petriho sieti a ich vlastnosti pomocou definovaných rozhraní.
- **actionTypes.enum.ts** - Definuje typy akcií pre prechody v Petriho sieti.

B.4 Využité technológie

echnológie, ktoré sme použili na vytvorenie aplikácie. V zozname je uvedená aj minimálna verzia technológie, ktorá je potrebná na úspešné spustenie programu:

- **Aframe** - verzia 1.4.1
- **Angular** - verzia 15.0.4
- **Bootstrap** - verzia 5.2.3
- **Java** - verzia 17
- **PostgreSQL** - verzia 15.3
- **Docker**

B.5 Spustenie programu

Na lokálny vývoj a spustenie aplikácie lokálne je potrebné postupovať podľa nasledujúceho postupu:

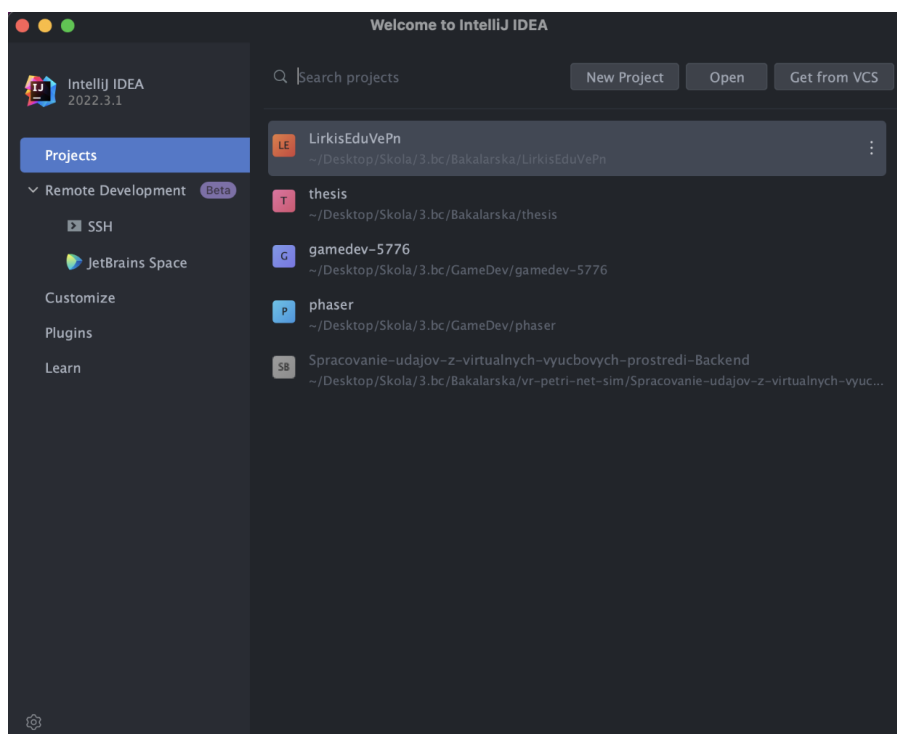
B.5.1 Stiahnutie zdrojových kódov

Súčasťou príloh tejto práce sú zdrojové kódy aplikácie, ktoré sú dostupné v komprimovanom balíku. Pre prístup k nim je potrebné balík stiahnuť a následne extrahovať jeho obsah.

B.5.2 Otvorenie v IDE

Po úspešnom stiahnutí a extrakcii projektových súborov je možné aplikáciu otvoriť v ľubovoľnom integrovanom vývojovom prostredí (IDE). V rámci tohto návodu odporúčame a používame IntelliJ IDEA od spoločnosti JetBrains. Postup spustenia:

1. Spustíte IntelliJ IDEA (úvodné okno je znázornené na obrázku B.1)
2. Vyberte možnosť Open z úvodného menu
3. Nájdite a označte rozbalený priečinok so zdrojovými kódmi aplikácie
4. Potvrďte výber tlačidlom OK



Obr. B.2: Otvorenie projektu

B.5.3 Príprava projektu na spustenie

Pred prvým spustením aplikácie je potrebné splniť nasledovné systémové požiadavky:

- **Java Development Kit:** verzia 17 alebo novšia
- **PostgreSQL:** databázový systém verzie 15.3
- **Node.js:** verzia 8.x alebo novšia (vrátane balíkového manažéra npm)

Nastavenie databázy

Ďalej je potrebné mať správne nakonfigurovanú databázu. Po nainštalovaní PostgreSQL vytvorte lokálnu prázdnu databázu v termináli (psql) alebo desktopovej aplikácii (pgAdmin), ktorú poskytuje PostgreSQL počas inštalácie. Následne je potrebné nastaviť vytvorenú databázu pre SpringBoot a to v súbore ktorý nájdete v LinksEduVePn-service/src/main/resources/application.properties. V súbore application.properties je potrebné upraviť url, username a password podľa nastavení PostgreSQL.

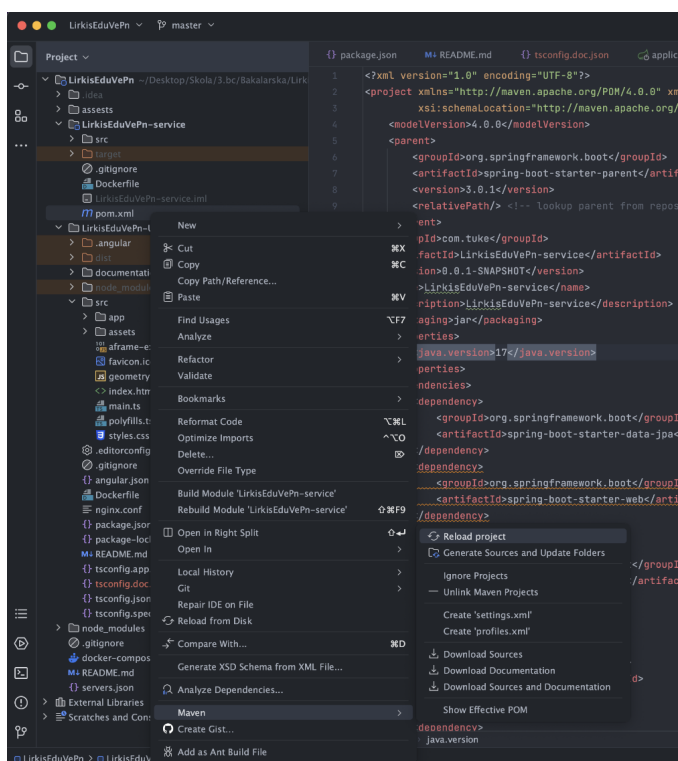
```
1  spring.datasource.url=jdbc:postgresql://db-postgres:5432/postgres
2  spring.datasource.username=postgres
3  spring.datasource.password=postgres
4  spring.jpa.hibernate.ddl-auto=update
5  jwt.secret=655368566D5971337436773979244226452948404D635166546A576E5A723475
6  jwt.expiration=120
7  spring.servlet.multipart.max-file-size=10MB
8  spring.servlet.multipart.max-request-size=10MB
```

Obr. B.3: Príklad súboru application.properties

Inštalácia závislostí

Serverová časť aplikácie vyžaduje nasledovné kroky pred spustením:

1. Uistite sa, že máte nainštalovaný a nakonfigurovaný **Apache Maven**
2. Prejdite do priečinka LirkisEduVePn-service
3. Kliknite pravým tlačidlom myši v priečinku
4. V kontextovom menu vyberte možnosť Maven
5. Zvoľte Reload project na stiahnutie všetkých závislostí



Obr. B.4: Maven závislosti

Pre správne fungovanie klientskej časti aplikácie je potrebné nainštalovať všetky závislosti špecifikované v súbore `package.json`. Otvorte terminál a prejdite do priečinka `LirkisEduVePn-UI`. Následne spustíte príkaz na inštaláciu závislostí:

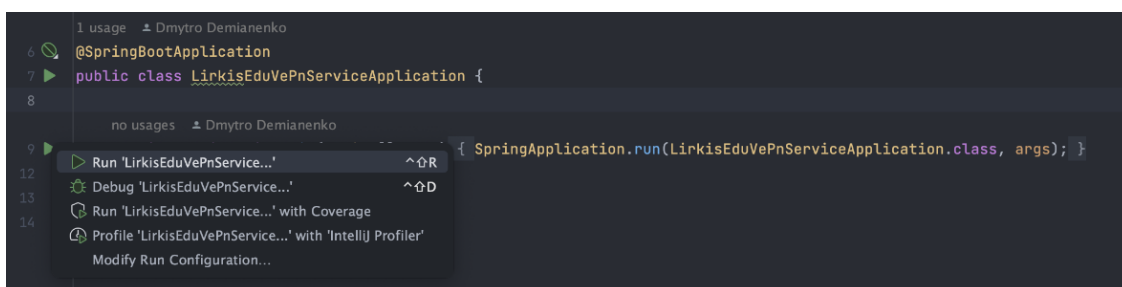
```
npm install
```

B.5.4 Spustenie projektu

Aplikácia vyžaduje samostatné spustenie klientskej a serverovej časti. V termináli prejdite do priečinka `LirkisEduVePn-UI` a spustíte príkaz:

```
ng serve
```

Pre serverovú časť otvorte súbor `LirkisEduVePnServiceApplication.java` (cesta: `LirkisEduVePn-service/src/main/java/com/tuke/lirkiseduvepnservice/`) a spustíte aplikáciu kliknutím na zelenú šípku v IDE (B.5).

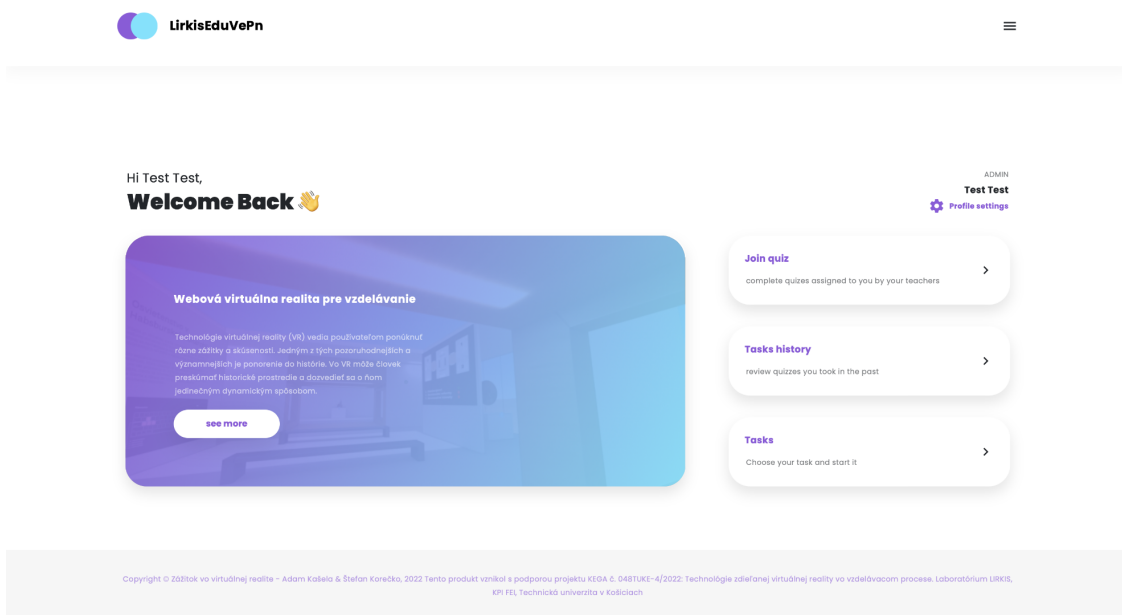


Obr. B.5: Spustenie servera

Po úspešnom spustení oboch častí systému je aplikácia dostupná v webovom prehliadači na adrese:

<https://localhost:4200>

Používateľ má na úvodnej obrazovke dve možnosti. Prihlásenie existujúcim účtom, alebo registrácia nového účtu (ak používateľ ešte nemá vytvorené prihlasovacie údaje). Po úspešnej autentifikácii systém zobrazí hlavné rozhranie aplikácie B.6, ktoré umožňuje prístup ku všetkým funkciám systému.



Obr. B.6: Úvodná obrazovka aplikácie

B.6 Nasadenie na server

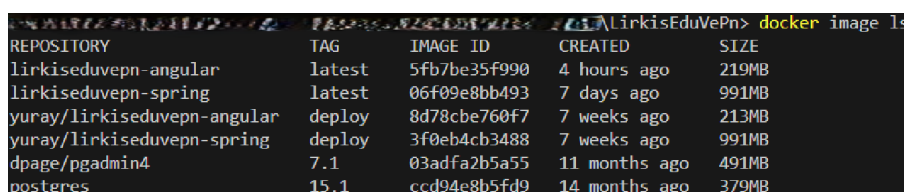
Pre nasadenie novej verzie na server je potrebný nasledovný postup:

1. Nahratie potrebných obrazov kontajnerov do Dockerhubu

2. Stiahnutie obrazov na server
3. Spustenie systému

B.6.1 Nahratie potrebných obrazov kontajnerov do Dockerhubu

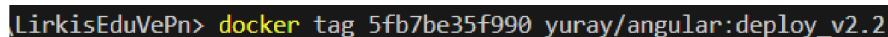
Pre nahratie obrazov (container image) do Dockerhubu odporúčame používať tagy. Tie slúžia na rozoznávanie rôznych verzií nahratých obrazov. Najskôr si vylistujeme zoznam obrazov. Potom si nad konkrétnym obrazom vytvoríme tagovanú kópiu. Následne pushneme obraz do Dockerhubu. Ako príklad použijeme tag v tvare "deploy_v2.2". Tieto kroky sú znázornené na obrázkoch nižšie.



REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
lirkiseduvepn-angular	latest	5fb7be35f990	4 hours ago	219MB
lirkiseduvepn-spring	latest	06f09e8bb493	7 days ago	991MB
yuray/lirkiseduvepn-angular	deploy	8d78cbe760f7	7 weeks ago	213MB
yuray/lirkiseduvepn-spring	deploy	3f0eb4cb3488	7 weeks ago	991MB
dpage/pgadmin4	7.1	03adfa2b5a55	11 months ago	491MB
postgres	15.1	ccd94e8b5fd9	14 months ago	379MB

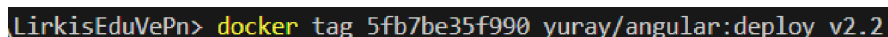
Obr. B.7: Vylisovanie obrazov

```
LirkisEduVePn> docker tag 5fb7be35f990 yuray/angular:deploy_v2.2
```



```
LirkisEduVePn> docker tag 5fb7be35f990 yuray/angular:deploy_v2.2
```

Obr. B.8: Tagovanie obrazu



```
LirkisEduVePn> docker tag 5fb7be35f990 yuray/angular:deploy_v2.2
```

Obr. B.9: Push do Dockerhubu

Tento postup je potrebné vykonať pre Spring aj pre Angular obrazy.

B.6.2 Stiahnutie obrazov na server

Po úspešnom prihlásení na server je potrebné získať potrebné Docker obrazy z Dockerhubu. Pri tomto príkaze použijete rovnaký tag obrazu, aký bol špecifikovaný v predchádzajúcom kroku konfigurácie. Tento proces zabezpečí, že všetky komponenty budú používať kompatibilné verzie softvéru.

```
docker pull yuray/angular:deploy_v2.2
```

```
docker pull yuray/spring:deploy_v2.2
```

Keď máme stiahnuté obrazy, potrebujeme si buď stiahnuť, alebo vytvoriť súbor `docker-compose.yaml`.

B.6.3 Spustenie systému

Pre spustenie systému použijeme tento príkaz:

```
docker-compose up --build -d
```

Kľúčové prepínače pre spúšťanie Docker kontajnerov majú špecifické funkcie. Prepínač `-build` zabezpečí novú inštaláciu všetkých obrazov, zatiaľ čo prepínač `-d` umožní spustenie kontajnerov v detached móde, čím uvoľní terminál pre ďalšie operácie. V prípade, že nepoužijete prepínač `-d`, terminál zostane blokovaný až do ukončenia činnosti kontajnerov. Toto správanie je však užitočné pri ladení, keď je potrebné sledovať výstup aplikácie v reálnom čase.