Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Softvérové prostriedky pre webovú virtuálnu realitu

Bakalárska práca

2025 **Adam Dargóczi**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Softvérové prostriedky pre webovú virtuálnu realitu

Bakalárska práca

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

Košice 2025 **Adam Dargóczi**

Abstrakt v SJ

Táto práca implementuje verziu hry Minesweeper vo virtuálnej realite (VR), vyvinutú pomocou troch webových 3D frameworkov: A-Frame, React Three Fiber a Babylon.js. Cieľom bolo preskúmať a porovnať možnosti týchto technológií v kontexte vývoja hier, najmä vo VR prostredí, a ukázať ich integráciu s backend serverom vytvoreným pomocou Spring Boot.

Všetky tri prototypy sa pripájajú na backend postavený na Spring Boot, ktorý spravuje stav hry a ukladanie herných relácií. Systém podporuje lokálny režim hry, v ktorom sa dynamicky generuje herné pole a priebeh hry sa aktualizuje v reálnom čase.

Projekt demonštruje vývoj multiplatformových VR hier pomocou moderných webových technológií. V budúcnosti je možné rozšíriť funkcionalitu o podporu pre viacerých hráčov alebo preskúmať alternatívne spôsoby interakcie v imerzívnom prostredí.

Kľúčové slová v SJ

webová virtuálna realita, a-frame, react-three-fiber, babylon

Abstrakt v AJ

This thesis implements a virtual reality (VR) version of the game Minesweeper, developed using three web-based 3D frameworks: A-Frame, React Three Fiber, and Babylon.js. The aim was to explore and compare the capabilities of these technologies in the context of game development, particularly within VR environments, and to demonstrate their integration with a backend server built using Spring Boot.

All three prototypes connect to a Spring Boot backend that manages game states and session storage. The system supports a local play mode, dynamically generating minefields and updating game progress.

The project demonstrates the developing of cross-platform VR games using modern web technologies and provides. Future work may include expanding multiplayer features or exploring alternative interaction methods within immersive environments.

Kľúčové slová v AJ

web virtual reality, a-frame, react-three-fiber, babylon

Bibliografická citácia

DARGÓCZI, Adam. *Softvérové prostriedky pre webovú virtuálnu realitu* . Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2025. 38s. Vedúci práce: Ing. Štefan Korečko, PhD.

77197

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE BAKALÁRSKEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Softvérové prostriedky pre webovú virtuálnu realitu

Software Tools for Web-Based Virtual Reality

Študent: **Adam Dargóczi**
Školiteľ: **Ing. Štefan Korečko, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce:
Pracovisko konzultanta:

Pokyny na vypracovanie bakalárskej práce:

1. Analyzovať aktuálne softvérové prostriedky pre webovú virtuálnu resp. rozšírenú realitu.
2. Navrhnúť webovú aplikáciu, demonštrujúcu možnosti webovej virtuálnej reality.
3. Implementovať verzie navrhnutej webovej aplikácie pomocou analyzovaných softvérových prostriedkov.
4. Na základe skúseností z implementácie zhodnotiť použité softvérové prostriedky.
5. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 23.05.2025
Dátum zadania bakalárskej práce: 31.10.2024

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -10-

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

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

Poďakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce za jeho čas a odborné vedenie počas riešenia mojej záverečnej práce. Rovnako by som sa rád poďakoval svojim rodičom a priateľom za ich podporu a povzbudzovanie počas celého môjho štúdia.

Obsah

Zoznam obrázkov

1 Úvod

Virtálna Realita (VR) [1] zaznamenala v poslednom desaťročí veľký rozvoj. VR umožňuje ľuďom zažiť miesta, hry a videá v pohlcujúcom 3D stereoskopickom prostredí. To sa dosahuje zapojením základných zmyslov používateľa, ako je zrak, sluch a hmat. Pohltenie závisí od toho, či je používateľ umiestnený do interaktívneho sveta a má možnosť preskúmať ho, aby nadobudol dojem, že sa nachádza na inom mieste. Veľkým záujmom mnohých ľudí sú sociálne alebo viacpoužívateľské (multiplayerové) VR zážitky. Tie využívajú rovnaké princípy pohltenia ako predtým, no s pridanou sieťovou funkcionalitou, vďaka ktorej sa môžu rôzni používatelia spojiť, nadväzovať nové priateľstvá alebo umožňuje firmám organizovať stretnutia na diaľku cez internet.

V tejto práci budú preskúmané existujúce softvérové riešenia na tvorbu VR zážitkov v internetovom prehliadači.

Najväčšou výhodou vývoja našej aplikácie v prehliadači je, že nie sú potrebné žiadne dodatočné inštalácie okrem webového prehliadača s podporou Web Extended Reality (WebXR) (predtým známeho ako Web Virtual Reality (WebVR)). Dokonca ani VR headset nie je nevyhnutný, keďže mnohé z týchto aplikácií sú hrateľné aj na bežnom počítači. Ďalšou silnou stránkou je prenosnosť týchto aplikácií, keďže sa nemusíme zaoberať obmedzeniami operačných systémov – stačí, aby bol k dispozícii webový prehliadač.

Niektoré z existujúcich riešení ponúkajú vynikajúcu integráciu pre viacerých hráčov, avšak existujú aj také, ktoré si vyžadujú softvér tretích strán alebo túto funkciu vôbec nepodporujú.

[1] VR: je plne pohlcujúci digitálny zážitok, ktorý nahrádza skutočný svet.

Formulácia úlohy

Cieľom tejto práce je zhodnotiť súčasný stav softvérových nástrojov dostupných pre vývoj imerzívnych VR aplikácií, ktoré bežia v internetovom prehliadači. Práca sa zameriava na vyhodnotenie rôznych knižníc, rámcov a vývojových prostredí, ktoré sú v súčasnosti k dispozícii. Viaceré technológie budú použité na implementáciu funkčne ekvivalentných prototypov.

V priebehu tejto práce sa budeme venovať nasledujúcim oblastiam:

- V kapitole 2 bude popísané, aké technológie sú dostupné a čo ponúkajú,
- Kapitola 3 definuje všeobecnú štruktúru prototypu,
- V kapitole 4 bude popísaná serverová logika podporujúca hernú mechaniku a komunikáciu,
- Kapitola 5 sa bude zameriavať na implementáciu jednotlivých rámca,
- Nakoniec kapitoly 6 a 7 zhrnú skúsenosti, identifikujú výhody, nevýhody a získané poznatky.

Implementáciou rovnakej VR aplikácie v rôznych rámcoch vytvoríme základ na ich porovnanie s cieľom analyzovať ich vhodnosť a jednoduchosť použitia.

2 Dostupné technológie

Táto kapitola sa zameria na súčasný stav virtuálnej reality v prehliadači a technológie použité vo väčšine existujúcich riešení. Spustenie virtuálnej reality v prehliadači si vyžaduje niekoľko komponentov. Najprv potrebujeme 3D grafický renderer s nejakým typom frameworku a knižnice, ktoré zjednodušia 3D renderovanie a umožnia nám ľahko spravovať obsah.

2.1 Technológie pre webovú VR

Táto časť opisuje podporné technológie webovej virtuálnej reality, ktoré sa bežne používajú alebo sú potrebné pre určité funkcie programov.

2.1.1 WebGL

Web Graphics Library (WebGL) je Application Programming Interface (API) zabudované vo väčšine webových prehliadačov, ktoré podporuje renderovanie 3D grafiky. Ide o nízkoúrovňové JavaScriptové grafické API založené na OpenGL ES [1], čo sťažuje jeho samostatné použitie, ale iné knižnice tento proces výrazne uľahčia. Funguje prostredníctvom interakcie s Graphics Processing Unit (GPU), aby efektívne vykreslil našu 3D scénu v reálnom čase. Stále je veľmi populárne a široko používané [2] a mnohé knižnice a frameworky ho využívajú ako hlavný renderovací engine. Samotné WebGL nepodporuje VR obsah, ale tu prichádza na rad Web Extended Reality (WebXR).

2.1.2 WebXR

Predtým známe ako (teraz zastarané) Web Virtual Reality (WebVR), Web Extended Reality (WebXR) je API, ktoré umožňuje používanie VR a Augmented Reality (AR) [2] zariadení v internetovom prehliadači na vytváranie pohlcujúceho zážitku. Poskytuje našej aplikácii vysoký výkon, pretože preferuje jednoduché pro-

[2] AR: v AR sa digitálny obsah v reálnom čase prekrýva s reálnym svetom

stredia, eliminuje potrebu layoutového enginu, ako je HTML alebo CSS, a priamo pristupuje k nášmu renderovaciemu enginu [3]. Webová aplikácia využívajúca WebXR najprv zistí Extended Reality (XR) [3] schopnosti zariadenia a ak sú dostupné, pošle požiadavku na použitie zariadenia. Ak nie sú dostupné žiadne zariadenia, prehliadač sa pokúsi vytvoriť "inline session"[4]. ktorá umožňuje zobraziť rovnaký obsah v 2D v prehliadači. Ak je však dostupné kompatibilné zariadenie, ktoré prijme požiadavku, API získa aktuálnu polohu, orientáciu a ak je potrebné, aj stav ovládačov a pokračuje vo vykresľovaní aplikácie v požadovanom počte snímok za sekundu [4].

2.1.3 WebAssembly

Hoci väčšina webu je postavená na JavaScripte z dobrého dôvodu, pre niektoré úlohy nie je dostatočný, najmä ak chceme náš kód doňho skompilovať. WebAssembly (skrátene Wasm) bol navrhnutý práve z tohto dôvodu. Je to nízkoúrovňový jazyk určený na to, aby priniesol výkon takmer ako pri natívnych aplikáciách do prehliadačov tým, že funguje podobne ako assembler a zároveň sa integruje s JavaScriptom [5]. Programy napísané v C, C++ alebo Ruste sa kompilujú pomocou Emscripten-u aby mohli byť použité v našom prehliadači.

Jeho kľúčovou výhodou je použitie predkompilovaného kódu, aby optimalizácie mohli prebiehať už počas kompilácie, čo umožňuje využitie plného výkonu zariadenia, pričom je zabezpečená pamäťová bezpečnosť. Vďaka tomu je kód kompaktný, čo je ideálne na prenos cez internet, kde môže byť kľúčová šírka pásma, a zlepšuje časy načítania a výkon.

[3] XR: je zastrešujúci pojem, ktorý zahŕňa všetky reálne a virtuálne kombinované prostredia, ako sú VR, AR a MR (zmiešaná realita).

[4] Inline session: Zobrazenie obsahu priamo na stránke pred vstupom do plne pohlcujúcej VR prezentácie.

2.2 Knižnice a rámce

Doteraz uvedené technológie slúžili na umožnenie práce s inými knižnicami a nebudeme ich priamo využívať vo veľkej miere. Primárne vyžijeme v tejto časti popísané technológie:

2.2.1 Three.js

Three.js je open-source JavaScript knižnica, ktorá zjednodušuje 3D renderovanie v prehliadači pomocou WebGL. Táto dodatočná vrstva je žiaduca, keďže WebGL vyžaduje pokročilé znalosti a jeho použitie môže byť pomerne náročné. Neposkytuje vstavaný editor, takže stále pracujeme s čistým kódom. Three.js má niekoľko kľúčových funkcií, ktoré z neho robia ideálny nástroj na prácu [6]:

- Prostredie - scéna, kamera, renderer,
- Geometria, materiály,
- Osvetlenie,
- Textúry, shadery,
- Animácie a interakcie.

Naším prvým a kľúčovým prvkom je prostredie, ktoré má tri časti. Scéna slúži ako koreňový kontajner pre všetky objekty, kamery, svetlá a siete, pričom komponenty používame na vytvorenie virtuálneho prostredia. Kamera je pohľad na našu scénu, z ktorej sa vykresľuje. Renderer (v tomto prípade WebGL) používa perspektívu kamery na vykreslenie scény, takže nemusíme priamo pracovať s WebGL.

Three.js ponúka vstavané geometrické tvary, ako sú kocky a gule, ale podporuje aj oveľa zložitejšie tvary a 3D modely, ktoré je možné importovať z 3D modelových súborov.

Osvetlenie je nevyhnutné pre každú 3D scénu a tu máme k dispozícii niekoľko typov svetiel. Farbu, intenzitu a spôsob interakcie svetiel s objektmi je možné upraviť. Ambientné svetlo pridáva farbu materiálu objektu, Bodové svetlá sú ako žiarovky, vyžarujú svetlo rovnomerne vo všetkých smeroch z bodu, reflektory pridávajú svetlo vo forme kužeľa a smerové osvetlenie sa používa pre objekty

ako slnko.

Textúry sa aplikujú na povrchy 3D objektov a používajú sa na vytváranie efektov ako farba, hĺbka a drsnosť. Shadery sa používajú pre pokročilú grafiku, čo vývojárom umožňuje kontrolovať vzhľad materiálov a pridávať efekty ako odrazy, lomy a osvetlenie prostredia.

Animácie aplikované na objekty umožňujú zmeny, ako sú zmeny pozície, rotácie, mierky alebo zmeny materiálu. Podporované sú aj externé animačné súbory s možnosťou integrácie do našej scény. Toto je aj priestor pre interakcie používateľa, kde vývojári môžu pridávať interaktivitu a ovládanie pohybu, pričom niektoré funkcie sú už zahrnuté v knižnici.

Three.js však ponúka integráciu so spomínaným WebXR API na vytváranie VR aplikácií od základu. Táto knižnica dokáže spravovať pohľad hráča, ovládače a prechody medzi VR a ne-VR zobrazeniami.

Na záver, Three.js je výkonný nástroj, ktorý nám umožňuje vytvoriť našu VR aplikáciu. Podpora načítania modelov vytvorených v 3D softvéri uľahčuje tvorbu našej scény a existuje mnoho doplnkových knižníc, ktoré môžu byť použité súčasne, z ktorých niektoré môžu pridať simuláciu fyziky, kolízie objektov, gravitáciu a mnoho ďalšieho. Nevýhodou je, že pri tvorbe aplikácie pracujeme s čistým kódom, bez editora, čo môže sťažovať umiestnenie objektov na správne miesto alebo úpravu scén, keďže nemáme okamžitú vizuálnu reprezentáciu.

Možnosti pre viac používateľov sú obmedzené a nie sú natívne podporované. Existujú však riešenia ako externé knižnice, napríklad Socket.IO alebo WebRTC, ale synchronizácia by sa musela implementovať ručne, čo je náročný proces.

2.2.2 A-Frame

A-Frame [7] je softvérový rámec navrhnutý špeciálne pre VR a AR na webe [8]. Je postavený na Three.js s HTML podobnou syntaxou umožňuje tvorbu jednoduchých scén, čím sa stáva jedným z prístupnejších jazykov pre začiatočníkov. Keďže Three.js používa WebGL pre grafiku a dedí všetky jeho funkcie, je podporovaný vo väčšine moderných prehliadačov a s podporou WebXR funguje na akomkoľvek kompatibilnom zariadení.

A-Frame je založený na architektúre Entity Component System (ECS), čo je bežná programová architektúra používaná pri vývoji hier, ktorá oddeľuje objekty a ich správanie. „Entita“ je kontajnerový objekt, ktorý môže obsahovať viaceré komponenty. „Komponenty“ sú opakovane použiteľné časti dát alebo správania.

Konkrétnym príkladom by bolo vytvorenie entity, ktorá predstavuje kocku, ktorá padá vplyvom gravitácie. Najprv by sme vytvorili komponent kocky, ktorý obsahuje údaje ako rozmery, farba, umiestnenie atď. Následne by sme túto kocku skombinovali s komponentom správania, ktorý implementuje gravitáciu, čo znamená, že ovplyvňuje jej pozíciu vždy, keď sa nachádza vo vzduchu. Zbalením týchto dvoch komponentov vznikne naša entita.

„Primitívy“ sú entity, ktoré majú vopred nastavené komponenty s predvolenými hodnotami priamo v A-Frame. Slúžia ako rýchle nástroje na implementáciu bežných entít, ktoré by inak bolo zdhavejšie implementovať.

A-Frame má vstavaný inšpektor s grafickým používateľským rozhraním (GUI) pomocou skratky Ctrl+Alt+I. Je to užitočný nástroj na vykonávanie menších zmien, ako je veľkosť entity, poloha a vlastnosti komponentov bez nutnosti prepínať medzi kódovým editorom a prehliadačom. Aj keď nie je tak výkonný ako iné aplikácie (napr. Unity), môže pomôcť s rýchlym ladením.

Hoci nie natívne, A-Frame má knižnicu s názvom Networked-Aframe (NAF), ktorá pridáva zdieľanie virtuálneho prostredia viacerými používateľmi. Podporuje používanie WebRTC, P2P rozhrania API alebo WebSockets, tradičnejšieho klient-server rozhrania API.

2.2.3 React-three-fiber

React je JavaScriptová knižnica pre webové a natívne používateľské rozhrania využívajúca opakovane použiteľné komponenty a virtuálny DOM.

React three fiber (R3F) je renderer, ktorý preklenuje priepasť medzi Reactom a Three.js. Obaluje kód Three.js a integruje ho do prostredia Reactu, čím uľahčuje pridávanie 3D grafiky do React projektov. R3F má komunitou vytvorené knižnice (napr. XR), ktoré poskytujú komponenty a hooky pre VR funkcionalitu. Hook je funkcia, ktorá poskytuje prístup k hodnote pomoci v pamäti a príkaz na zmenu tejto hodnoty. Dotaz a príkaz.

Aj keď Meta R3F priamo nevyvíja, má o tento framework veľký záujem a odporúča jeho používanie vývojárom hier pre ich VR platformu.

R3F nemá zabudovaný editor, no existuje veľa editorov tretích strán, ako react-three-editor [5] alebo Triplex [6] ktoré dobre spolupracujú s akýmkoľvek projektom postaveným na R3F.

Podpora pre viacero používateľov je podobná ako pri Three.js – technicky je to možné, dokonca existujú riešenia pre sieťovú komunikáciu v R3F, ale neexistujú dedikované knižnice pre zdieľané VR zážitky.

2.2.4 Babylon.js

Babylon.js je open source 3D engine určený na vytváranie 3D webových aplikácií. Babylon.js nie je alternatívou ku Three.js, pretože je definovaný ako 3D engine v reálnom čase, zatiaľ čo Three.js je iba JavaScriptová knižnica. Babylon.js rovnako využíva WebGL a programovací jazyk TypeScript alebo JavaScript.

Nedávno bola pridaná podpora pre API s názvom WebGPU, ktoré umožňuje prehliadaču využiť grafickú kartu používateľa (ak je dostupná), čím sa výrazne zlepšuje výkon v zložitejších aplikáciách.

Podobne ako Three.js, Babylon.js poskytuje základný nástrojový set pre renderovanie, osvetlenie, materiály, no zároveň natívne podporuje aj fyziku, animácie a shadery [9].

Babylon.js je považovaný za jednoduchý na použitie, čo podporuje aj rozsiahla dokumentácia. WebXR je plne podporovaný s integrovanou funkčnosťou pre

[5] Applikácia react-three-editor - Dostupná na: https://github.com/pmndrs/react-three-editor
[6] Applikácia Triplex - Dostupná na: https://triplex.dev/

stereoskopický rendering, ovládače a dokonca aj priestorový zvuk, čo ho prakticky predurčuje na tvorbu webXR. Na rozdiel od Three.js, natívne podporuje aj AR.

Podpora pre viac používateľov nie je natívne dostupná, avšak oficiálna dokumentácia obsahuje príklad použitia knižnice Colyseus [10] pre multiplayer v reálnom čase, ktorý by mohol byť potenciálne použitý aj v našom projekte.

3 Koncepčný návrh

Naše prototypy budú webové aplikácie, čo znamená, že na dosiahnutie nášho cieľa môžeme kombinovať rôzne technológie. Cieľom bude mať hrateľnú hru v 3D prostredí virtuálnej reality. Na tento účel bola vybraná hra Minesweeper, ktorú sme už raz implementovali na hodine Komponentové programovanie podľa tohto repozitára [7].

Cieľom hry Minesweeper je nájsť všetky míny v sieti štvorcov. Všetky štvorce sú na začiatku skryté. Bezpečné štvorce po otvorení odhaľujú čísla s počtom mín, ktoré s daným štvorcom susedia. Môžeme označiť políčka, na ktorých si myslíme, že sa nachádza mína, a kliknutím na mínu sa hra skončí. Otvorenie všetkých bezpečných políčok bude mať za následok výhru.

Každý prototyp musí obsahovať nasledujúce prvky:

- **Stereoskopické zobrazenie:** Hra musí byť hrateľná aj vo VR headsete. Ak je to možné, implementuje sa aj inline session[8].
- **Ovládanie pohybu:** Používateľ musí byť schopný pohybovať sa a interagovať so svetom.
- **Prostredie:** Pre komfort hráča musí existovať nejaké vizuálne prostredie, aby sa zabránilo dezorientácii počas hrania.
- **Interaktívne objekty:** Prototyp musí obsahovať hru Minesweeper, ktorá sa nachádza v hernom svete prototypu, musí byť interaktívna pomocou ovládačov a plne hrateľná.

Používatelia s VR headsetmi budú môcť použiť ovládače na pohyb a interakciu. Pomocou inline relácie je možné štandardné ovládanie pohybom klávesnice a myši.

[7] Repozitár pre Minesweeper - Dostupná na: https://git.kpi.fei.tuke.sk/jaroslav.poruban/gamestudio2024

[8] Inline session: Zobrazenie obsahu priamo na stránke pred vstupom do plne pohlcujúcej VR prezentácie.

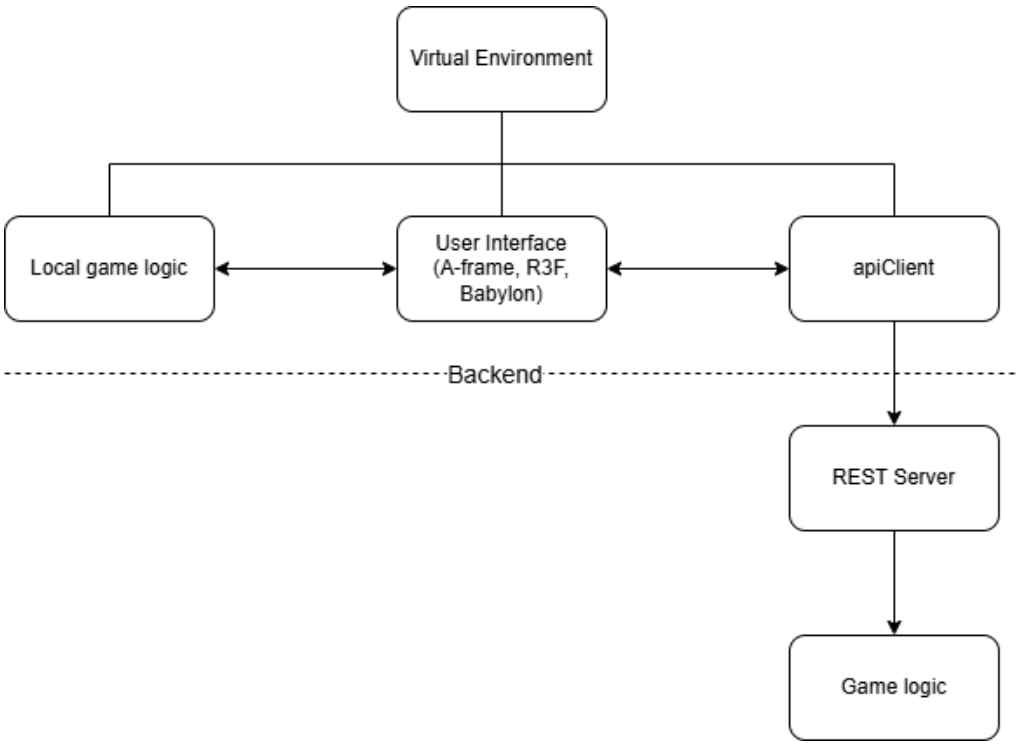
3.1 Rozdelenie prototypov

Našu webovú aplikáciu môžeme rozdeliť na dve časti:

- **Backend** (server) - server na obsluhu hernej logiky,
- **Frontend** (klient) - interakcia s používateľom, vizuálne prvky, komunikácia so serverom

Logika pre hru MineSweeper bude mať dve možnosti; buď komunikáciou so serverom, alebo použitím lokálne implementovanej logiky (Obrázok 3.1). Pri používaní servera klient oznámi každú interakciu používateľa s hrou. Druhá možnosť použije implementáciu logiky na strane klienta, ktorý nebude komunikovať so serverom.

Obrázok 3.1: Diagram so zjednodušeným prototypom



Každý rámec má vlastnú implementáciu spracovania interakcie s používateľom a vizuálov, takže tieto časti budú špecifické pre daný rámca.

Použitý backend bude rovnaký pre každý prototyp. Proces bude nasledovať tieto kroky:

- Prijímať vstupy z frontendu,
- Spracovať ťah,

- Aktualizovať stav hry,
- Komunikovať s frontendom

Vykresľovanie hry bude zabezpečené frontendom, ktorý bude komunikovať s naším serverom, aby bolo možné hrať Minesweeper, alebo použije lokálnu logiku, ak je špecifikovaná. To znamená, že frontend bude musieť byť schopný:

- Vykresliť prostredie,
- Spracovať vizuálnu reprezentáciu (monitor alebo VR headset),
- Prijímať vstupy od používateľa,
- Komunikovať s backendom

Budeme musieť byť schopní vykresliť prostredie prostredníctvom bežného 2D webového prehliadača aj VR headsetu.

Front-end by mal tiež spracovávať používateľské vstupy pre Minesweeper a odosielať ich do back-endu, spolu s aktualizáciou vizuálov s novým herným stavom prijatým zo servera.

3.2 Vybrané technológie

Backend (logika) bude implementovaný v Jave pomocou Spring Boot. Spring Boot a jeho REST API (REpresentational State Transfer) boli zvolené pre tento projekt pre jednoduchú integráciu s Java ekosystémom. Backend bude spravovať aplikačné dáta a poskytovať API endpointy pre komunikáciu s frontendom prostredníctvom HTTP požiadaviek.
Pre frontend boli vybrané tieto rámce:

- A-Frame
- React-three-fiber
- Babylon.js

Spolu s tým, s výnimkou A-Frame, budú tieto softvérové rámce tiež používať Vite.js [9]. ako svoj frontendový nástroj, pričom pre A-Frame sme použili Five--server [10].

Tieto technológie podporujú 3D vykresľovanie a VR, čo ich robí ideálnymi pre náš projekt, pričom sú dostatočne odlišné, aby poskytli rozmanité výsledky.

[9] Vite.js - Dostupná na: https://vite.dev/
[10] Five-server - Dostupná na: https://github.com/yandeu/five-server

4 Implementácia servera

Server slúži na spracovanie logiky hry Minesweeper vo všetkých prototypoch, čo znamená, že pre viacero frontendov stačí spustiť iba jeden server. Podporuje interakciu používateľa tým, že poskytuje údaje o hernom poli a odpovedá na API požiadavky z frontendu.

Implementácia servera, ktorú sme použili, je z repozitára [11], ktoré bolo použité v hodine Komponentové programovanie.

Implementácia využíva Spring Boot, čo je Java framework určený na REST API a webové aplikácie. Hlavná myšlienka je sprístupniť sadu koncový bode, ktoré nám umožnia získať a aktualizovať údaje v reálnom čase.
Sú vystavené dva hlavné koncový body:

- **POST** /mines/json
- **POST** /mines/jsonnew

Tieto sú vytvorené veľmi jednoducho – /mines/jsonnew sa používa na vytvorenie novej inštancie hry, zatiaľ čo /mines/json slúži na interakciu s už aktívnou hrou.

Nová hra vyžaduje 3 argumenty: počet riadkov, stĺpcov a počet mín. Pri interakcii s hrou berieme do úvahy iba jeden vstup naraz, ktorý však môže zmeniť stav viacerých políčok. Z tohto dôvodu stačia na interakciu 3 vstupné hodnoty: riadok, stĺpec políčka a boolean hodnota „marking“. Keď je marking nastavený na „false“, políčko sa otvorí. Keď je „true“, políčko sa označí ako mínové.

Všetky koncové body vracajú rovnaké údaje o hre vo formáte JSON, ktorý pozostáva z:

- **pole** Tiles[ ];
- **reťazec** gameStatus;

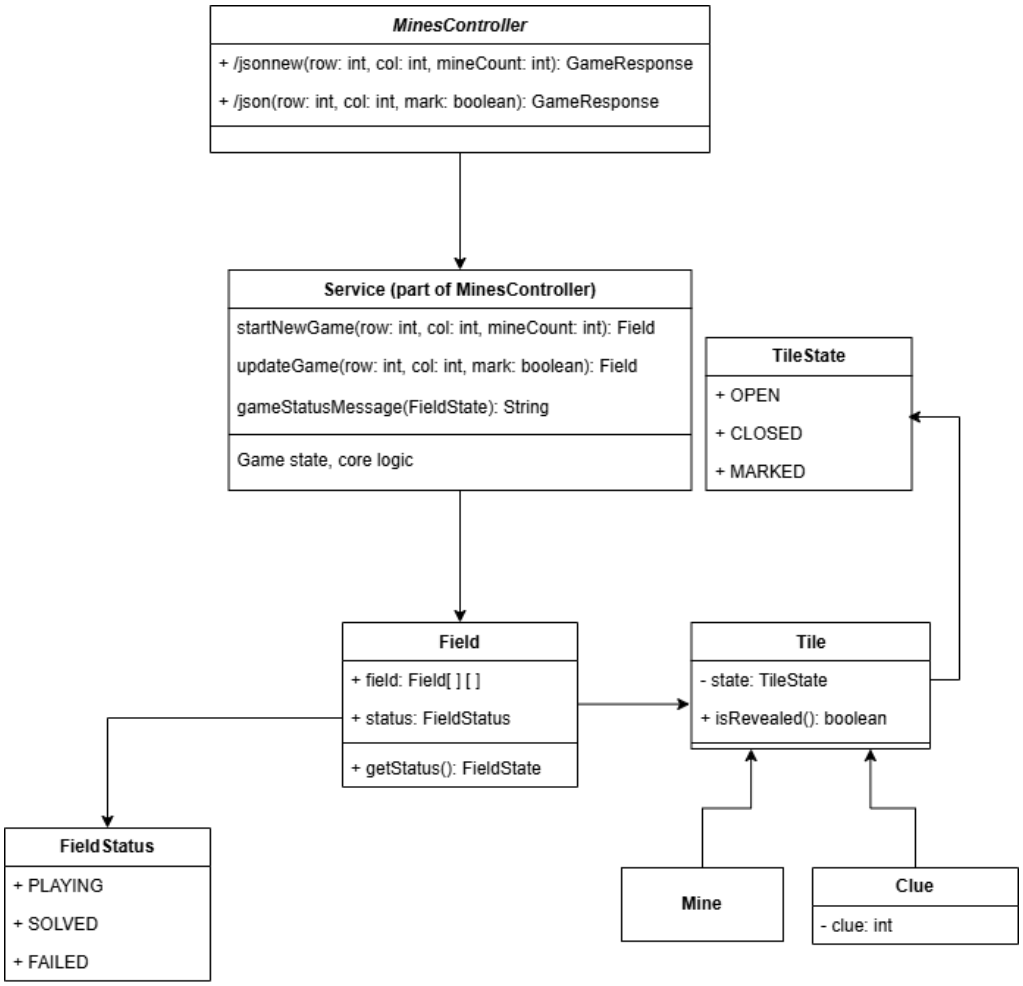
[11] Repozitár Minesweeper - Dostupná na: https://github.com/skorecko/GameStudioTacad.

Reťazec gameStatus obsahuje text, ktorý udáva aktuálny stav hry: môže byť „playing“, „failed“, alebo „complete“. Pole Tiles obsahuje všetky aktívne políčka a údaje o nich, vrátane nasledujúcich vlastností:

- **boolean** revealed;
- **boolean** mine;
- **int** clue;
- **boolean** marked;

Tieto sú pomerne samozrejmé: „revealed“ označuje, či bolo políčko odkryté, „mine“ označuje, či políčko obsahuje mínu, „clue“ je počet mín v okolí políčka a „marked“ označuje, či je políčko označené ako podozrivé. Diagram tried (Obrázok 4.1) znázorňuje, ako funguje server a logika.

Obrázok 4.1: Vizualizácia štruktúry backendu



Požiadavky a komplikácie

Keďže WebXR je považovaný za výkonné API, vyžaduje zabezpečený kontext na zabránenie útokom typu man-in-the-middle, čo znamená, že funguje iba cez HTTPS [11]. Náš server je však určený len pre HTTP. To síce neblokuje načítanie stránky, ale zabraňuje používaniu VR zariadení. Najjednoduchší spôsob, ako toto obísť, je použiť adresu localhost, ktorá vždy odkazuje na počítač používateľa.

5 Implementácia klienta

5.1 Zdieľaná logika

Spracovanie komunikácie so serverom na strane klienta sa môže opakovane použiť vo všetkých prototypoch, preto bolo rozdelené do 3 súborov.

Dva slúžia na obsluhu komunikácie so serverom a jeden na obsluhu hry s lokálnou logikou, ak sa rozhodneme nespustiť server. Tieto 3 súbory sú:

- **apiClient.js**,
- **mineSweeperState.js**,
- **mineSweeperLogic.js**.

`apiClient` je súbor zodpovedný za komunikáciu so serverom. Má preddefinovanú adresu, ktorú kontaktuje pomocou vopred definovaných POST metód.

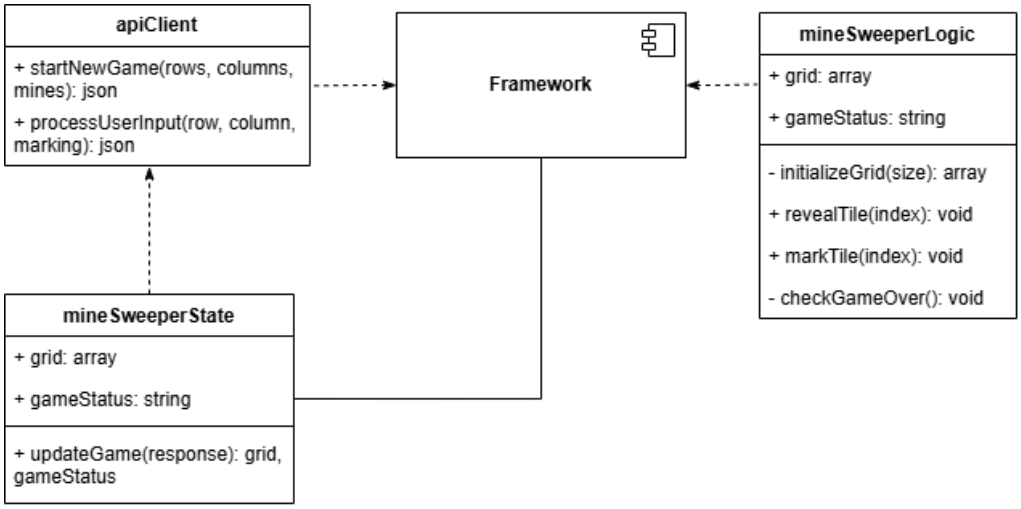
`mineSweeperState` je trieda, ktorá obsahuje pomocnú funkciu na preklad prijatého JSON objektu zo servera. Ďalšou jej úlohou je uchovávať pole políčok a stav hry, ktoré je možné jednoducho získať.

`mineSweeperLogic` je posledný súbor, ktorý obsahuje logiku hry MineSweeper v rámci triedy. Pri vytvorení inštancie triedy môžeme určiť veľkosť hracej mriežky (mriežka bude vždy štvorcová) a počet mín, ktoré sa majú umiestniť. Trieda obsahuje aj funkcie na logickú aktualizáciu hry, ako je otváranie/označovanie políčok a detekciu a odhalenie celej mriežky po ukončení hry.

5.2 Všeobecná časť logiky

Vo všetkých prototypoch sa bude herná logika implementovaná rovnako (Obrázok 5.1).

Obrázok 5.1: Všeobecná herná štruktúra rámci



Časť, ktorá implementuje rámec bude obsahovať niekoľko vecí na implementáciu hry. Po prvé, uchováva referenciu na `mineSweeperState` a ďalšie premenné zobrazené v diagrame 5.1. Po druhé, obsahuje funkcie na logickú aktualizáciu hry a jej stavu, buď komunikáciou so serverom, alebo použitím lokálnej (offline) logiky, ak je to špecifikované. Nakoniec vykonáva grafické aktualizácie políčok.

Na vytvorenie hry alebo spracovanie vstupu časť, ktorá implementuje rámec, volá skript **apiClient**, ktorý zabezpečuje komunikáciu so serverom. Po prijatí odpovede deleguje spracovanie informácií na **mineSweeperState**, čo je objekt, ktorý uchováva a aktualizuje pole políčok.

Ak je to špecifikované, je možné použiť aj lokálnu logiku. Keď je táto možnosť povolená, Framework využíva triedu `mineSweeperLogic`. **mineSweeperLogic** obsahuje logiku potrebnú na hranie hry bežným spôsobom, pričom uchováva vlastné pole políčok a stav hry. V podstate kombinuje funkcie `apiClient` a `mineSweeperState`.

Každý framework bude tiež obsahovať triedu **Tile**, ktorej úlohou je vytvoriť alebo aktualizovať dlaždicu vždy, keď nastanú zmeny, a tiež obsahuje informácie o sebe. Každá dlaždica má svoje vlastné interakcie s používateľom.

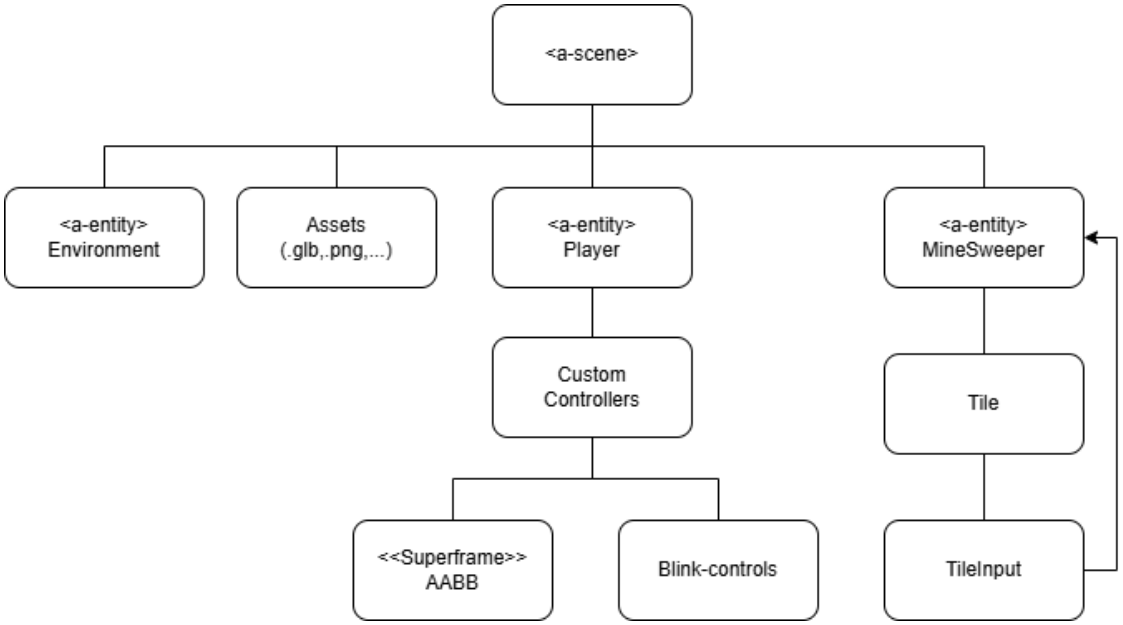
5.3 Implementácia A-Frame

Prvá implementácia bola vytvorená pomocou frameworku A-Frame. Je považovaný za jednoduchý, ale schopný nástroj na začiatok, a to vďaka svojej štruktúre podobnej HTML a množstvu užitočných komponentov, ktoré sú predinštalované.

Architektúra a komponenty

Ako už bolo spomenuté, A-Frame používa HTML a Document Object Model (DOM). Náš HTML súbor slúži ako hlavný súbor, ktorý obsahuje všetky skripty a komponenty, ktoré používame. A-Frame používa značku <a-scene> (Obrázok 5.2), ktorá obsahuje všetky komponenty, ktoré chceme zahrnúť do scény.

Obrázok 5.2: Vizualizácia obsahu značky <a-scene>



A-Frame poskytuje tzv. „primitívy“, čo sú predpripravené komponenty ako napríklad <a-box> (kocka) alebo <a-sky> (obloha) s prednastavenými a prispôsobiteľnými hodnotami.

Na tvorbu vlastných skriptov a elementov v hlavnom HTML súbore A-Frame odporúča vytvárať tzv. komponenty. Komponent predstavuje jednoduchý spôsob, ako pripojiť logiku a/alebo objekty k entite. Komponenty obsahujú niekoľko hlavných funkcií a štruktúra údajov, ako napríklad:

- **schema**,

- **init**,
- **update**,
- **remove**.

Schema definuje vlastnosti komponentu pomocou kľúčov a hodnôt. Po pripojení komponentu k entite môžeme dynamicky zadávať hodnoty a tým meniť správanie komponentu.

Init sa volá na začiatku životného cyklu komponentu. **Update** sa volá vždy, keď sa zmenia vlastnosti komponentu. **Remove** sa volá pri odstránení komponentu z entity.

Okrem komponentu Minesweeper budú naše prototypy obsahovať aj niekoľko pomocných komponentov, ako napríklad vlastné modely ovládačov a ovládacie prvky (na interakciu s dlaždicami), komponent Tile na vykresľovanie a komponent TileInput na spracovanie interakcií.

Použité balíky

A-Frame nám umožňuje využívať balíky vytvorené komunitou, ktoré pomáhajú vyplniť medzery alebo zjednodušiť vývoj niektorých funkcionalít. Pri implementácii prototypu boli použité tri balíky:

- Superframe[12] – detekcia kolízií objektov
- Blink Controls[13] – ovládanie pohybu VR hráča
- Environment[14] – predpripravené prostredie pre scény

Detaily implementácie

Aby bola zachovaná štruktúra ECS (Entity-Component-System), implementácia hry Minesweeper je zabalená do jedného komponentu spolu s ďalšími pomocnými komponentmi.

Nasleduje prehľad hlavných komponentov:

[12]Knižnica Superframe - Dostupná na: https://github.com/supermedium/superframe.

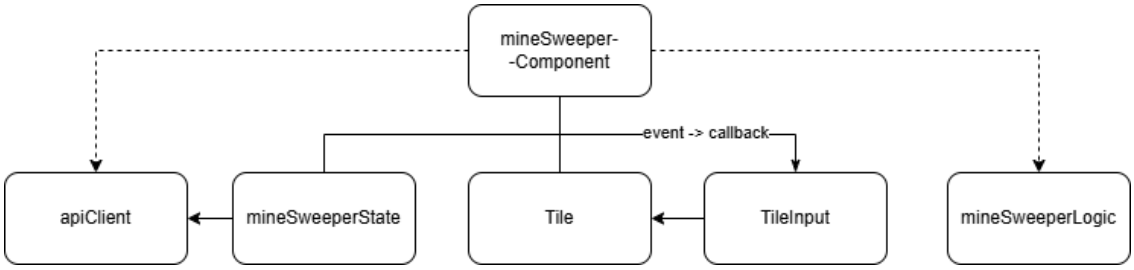
[13]Knižnica aframe-blink-controls - Dostupná na: https://github.com/jure/aframe-blink-controls.

[14]Knižnica Environment - Dostupná na: https://github.com/supermedium/aframe-environment-component.

mineSweeper, Tile, TileInput

Jadro hernej logiky Minesweeper. Tento komponent je zodpovedný za generovanie hracieho poľa, sledovanie stavu hry, vyhodnocovanie výhry/prehry a aktualizáciu vizuálu dlaždíc na základe ich stavu. Obrázok 5.3 nám ukazuje interakciu medzi komponentmi.

Obrázok 5.3: Diagram zobrazujúci interakciu medzi komponentmi pre Minesweeper



Každá dlaždica v hre Minesweeper je entita s pripojeným komponentom tile. Tento komponent uchováva informácie o svojom stave a aktualizuje svoj vzhľad.

TileInput detekuje vstupy hráča — či už ide o kliknutie myšou alebo interakciu pomocou VR ovládača — a odosiela príslušné príkazy hernej logike. Keď sa hráč pozrie na dlaždicu, tá získa atribút 'active'. TileInput potom vyhľadá aktívnu dlaždicu a vykoná požadovanú akciu.

custom.controller, flagCollider, stickCollider

Pre VR interakciu sú 3D modely nástrojov (vlajka a palica) pripojené k ľavému a pravému VR ovládaču. Komponent `custom.controller` sa stará o zmenu modelov a pripojenie kolíznych boxov na správne miesto. Komponenty `flagCollider` a `stickCollider` sú pomocné komponenty, ktoré volajú správne akcie pri kolízii s dlaždicami.

Ovládanie na počítači a interakcia s používateľom

Na pohyb v 3D scéne poskytuje A-Frame základné, ale dobre použiteľné ovládanie pomocou kláves WASD a myši (pohyb kamery).

Na otvorenie alebo označenie dlaždice treba ukázať na požadovanú dlaždicu stredom obrazovky (kde sa nachádza kurzor). Otvorenie sa vykonáva ľavým klikom, označenie klávesom 'E'.

Vizuál a zloženie scény

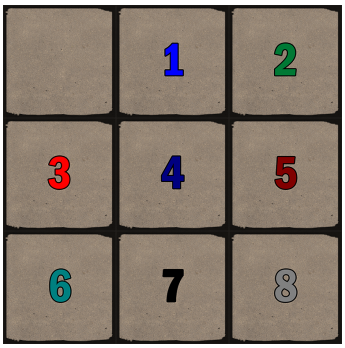
Ako bolo spomenuté, prostredie bolo vytvorené pomocou balíka Environment, ktorý bol upravený pre jedinečný vzhľad hry. Okrem toho boli do scény pridané aj 3D modely vo formáte glb/gltf. Tieto formáty sú najviac podporované a odporúčané dokumentáciou. Obsahujú modely aj textúry, ktoré sa dajú použiť v scéne.

Oficiálna dokumentácia odporúča načítať všetky textúry a objekty, ktoré chceme použiť, vopred do značky <a-assets>.

Použitie objektov je jednoduché — pomocou atribútu 'gltf-model:' v značke <a-entity> sa model načíta zo zdroja a následne je možné upraviť jeho pozíciu, veľkosť, tieňovanie a ďalšie vlastnosti.

Na vizuálne rozlíšenie čísel na dlaždiciach boli použité textúry s číslami, ktoré sa zobrazia po odhalení dlaždice. Na tento účel bol použitý **textúrový atlas** (Obrázok 5.4).

Obrázok 5.4: Príklad textúrového atlasu



Textúrový atlas je spôsob, ako zabaliť viacero textúr do jedného súboru s cieľom šetriť výkon procesora a zlepšiť rýchlosť načítania tým, že sa viacero textúr načíta naraz[12], a následne sa „rozbalia“ pomocou UV mapovania [15]. Aj keď musíme načítať viacero textúr (8 čísel, otvorené, zatvorené a mínové dlaždice), tento spôsob má len minimálny dopad na výkon, najmä na moderných zariadeniach.

VR nastavenie

Na VR podporu nebolo potrebné meniť veľa z toho, čo A-Frame poskytuje. Jediné, čo bolo upravené alebo pridané, boli modely ovládačov, interakcia s dlaždicami

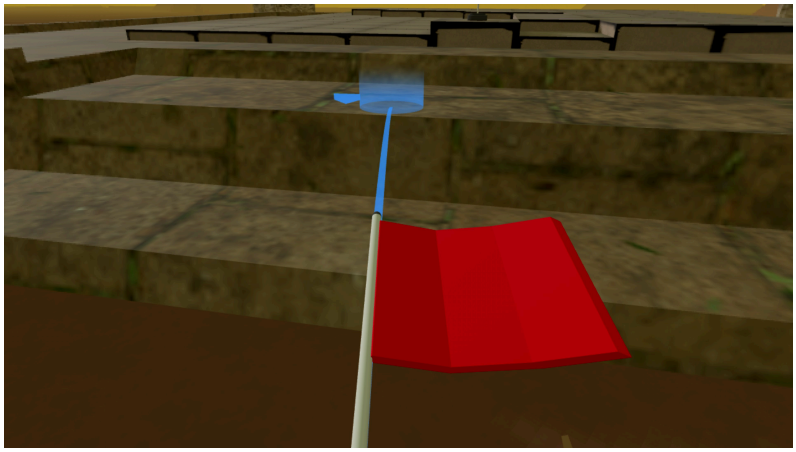
[15] "UV mapovanie je proces v 3D modelovaní, pri ktorom sa povrch 3D modelu projektuje na 2D obraz pre potreby texturovania." Wikipedia

a spôsob pohybu.

Modely ovládačov, interakcia a spôsob pohybu boli už popísané v časti o implementácii: custom.controller, flagCollider, stickCollider.

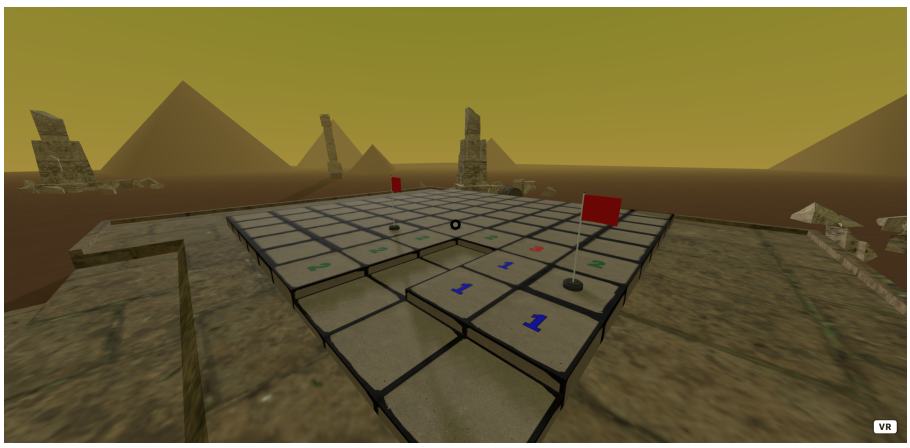
Na VR pohyb bol použitý balík aframe-blink-controls. Funguje tak, že hráč ukáže ovládačom na miesto, kam sa chce presunúť, a pomocou joysticku spustí teleportáciu (Obrázok 5.5).

Obrázok 5.5: Fungovanie Blink Controls v hre



Vo VR sa dlaždice otvárajú pomocou špičky palice a označujú/odznačujú pomocou základne vlajky.

Obrázok 5.6: Finálna scéna v A-Frame, zachytená na počítači



5.4 Implementácia Babylon.js

Babylon.js sa prezentuje ako viacúčelová platforma určená na interaktívne prezentácie a občas aj 2D hry. Napriek tomu natívne podporuje VR. Ako bolo spomenuté v kapitole 2, Babylon nepoužíva Three.js ako svoj vykresľovací engine. Výsledkom je o niečo unikátnejší, ale stále známy spôsob tvorby scén.

Architektúra a skripty

Babylon podporuje vývoj v JavaScripte aj TypeScripte. Keďže dokumentácia a príklady sú prevažne v JavaScripte, tento prototyp bol implementovaný práve v ňom.

Tvorba scény v Babylone pôsobí podobne ako pri React aplikácii. V HTML súbore stačí vytvoriť element `<canvas>` s identifikátorom a pripojiť hlavný JavaScriptový súbor ako skript.

V hlavnom súbore najprv inicializujeme triedu `BabylonApp`, po čom môžeme začať vytvárať scénu. Namiesto skladania entít jednu po druhej vytvárame objekt `scene`, do ktorého všetko pridávame. Až po jeho dokončení ho odovzdávame vykresľovaciemu enginu, ktorý spustí hru.

Projekt si vystačí s veľmi malým množstvom súborov. Okrem zdieľaného priečinka s logikou pozostáva iba z troch JavaScriptových súborov:

- main.js,
- tile.js,
- controlsSetup.js.

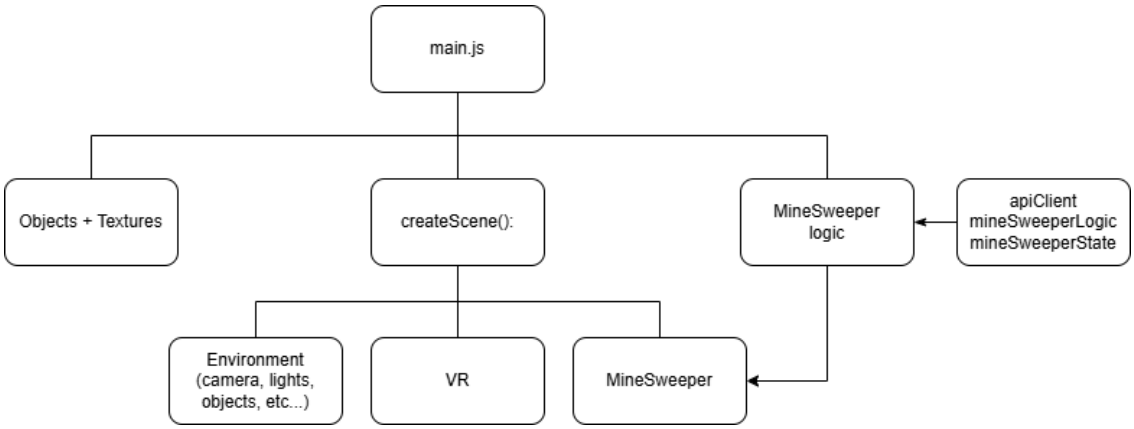
Použité knižnice

Babylon.js nevyžadoval žiadne externé balíčky alebo knižnice.

Detaily implementácie

Súbor `main.js` (Obrázok 5.7) obsahuje väčšinu použitej logiky. Zodpovedá za tvorbu scény, implementáciu hry Minesweeper a podporu VR.

Obrázok 5.7: Zjednodušená štruktúra `main.js`



createScene

Hlavná funkcia, ktorá vytvára scénu, ktorú následne odovzdávame rendereru. Aplikácia sa nespustí, pokiaľ táto funkcia neskončí.
Jej zodpovednosti sú:

- Tvorba vizuálneho prostredia,
- Implementácia logiky hry Minesweeper,
- Pridanie VR podpory.

Vizuálne prostredie a VR podpora sú rozobrané v samostatných sekciách.

Minesweeper, Herná logika, Tile.js

Súbor `main.js` obsahuje logiku komunikácie so serverom alebo s lokálnym herným enginom, generuje herné pole, sleduje stav hry (výhra/prehra) a aktualizuje vzhľad políčok.

Logika je veľmi podobná diagramu 5.3, s výnimkou modulu `TileInput`. Interakcia s políčkami bola zjednodušená pomocou `ActionManagera`, ktorý pri kliknutí spustí príslušnú funkciu.

Súbor `tile.js` plní rovnakú úlohu ako v A-Frame – vytvára políčka pre herné pole a aktualizuje ich vizuálny stav podľa potreby.

ControlsSetup.js

Tento súbor má jednoduchú úlohu: pridáva podporu pre ovládanie pomocou kláves WASD a myši.

Ovládanie a interakcia na desktopoch

Babylon ponúka niekoľko typov kamier s rôznymi schémami ovládania. Pre naše účely bola najvhodnejšia universalCamera. Po nakonfigurovaní ovládania a zmene klávesových skratiek máme desktopovú navigáciu hotovú.

Na objekty a mesh-e je možné pripojiť ActionManager, ktorý umožňuje spúšťať funkcie pri špecifických udalostiach (napr. ľavé/pravé kliknutie), ako napríklad odkrytie alebo označenie políčka.

Vizualizácia a skladba scény

Osvetlenie a generátor tieňov boli jednoduché na implementáciu a editor (Shift+Ctrl+Alt+I) výrazne uľahčil ich prispôsobenie.

Skybox musel byť pridaný manuálne. V skutočnosti ide o veľkú guľu so špeciálnym materiálom, ktorý imituje oblohu so slnkom. Tento materiál umožňuje prispôsobiť uhol svetla, jeho intenzitu a farbu oblohy.

Obrázok 5.8: Demonštrácia materiálu oblohy z oficiálnej dokumentácie

Na aplikáciu textúr na políčka bolo najprv potrebné vytvoriť a načítať samotné textúry. Tentoraz sa rozhodlo upustiť od používania atlasu textúr, pretože minimálna optimalizácia, ktorú poskytoval neospravedlňovala potrebné úsilie.

Scéna má špeciálny parameter **metadata**, v ktorom môžeme ukladať rôzne užitočné referencie – napríklad na textúry, materiály alebo `mesh-e`.

Metadáta sa použili na:

- uchovanie textúr pre čísla na odkrytých políčkach,
- referencie na načítané objekty (podlaha, dekorácie, ...),
- inštanciu triedy, ktorá umožňuje tieňovanie.

Na použitie textúry je potrebné najprv vytvoriť `StandardMaterial` pre každý stav políčka, potom načítať a nakonfigurovať textúru, a nakoniec uložiť referenciu do metadát. Textúru potom môžeme kedykoľvek použiť cez túto referenciu.

Na načítanie 3D modelu je potrebné najprv načítať celý súbor do kontajnera, ktorý obsahuje všetky prvky (mesh-e, podmesh-e, textúry a materiály). Aby bol objekt viditeľný v scéne, je potrebné ho pridať zo svojho kontajnera do scény. Potom môžeme upravovať jeho vlastnosti.

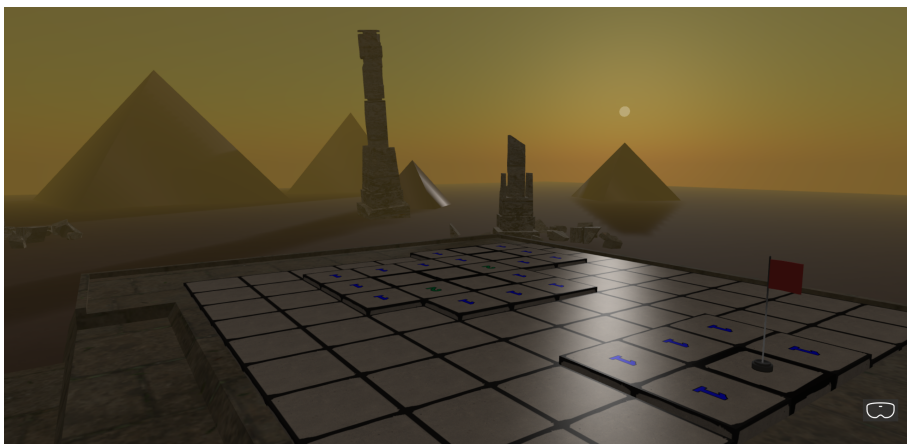
Ak chceme použiť model viackrát, najlepšou praxou je klonovanie už existujúceho kontajnera, aby sme predišli problémom s výkonom a správou objektov.

VR podpora

Babylon natívne podporuje VR, čo značne uľahčuje jeho implementáciu. Obsahuje aj teleportáciu, ktorá funguje podobne ako na obrázku 5.5. Stačí definovať, s akými meshmi môže ukazovateľ teleportácie kolidovať, a následne nastaviť ovládanie interakcie s políčkami.

Rovnako ako v A-Frame verzii, aj tu nahrádzame ľavý a pravý ovládač modelmi vlajky a paličky. Na správne miesta týchto objektov pridávame hitboxy. Po stlačení triggeru sa kontroluje, či daný hitbox koliduje s nejakým políčkom a vykoná sa príslušná akcia.

Obrázok 5.9: Demonštrácia finálnej scény v Babylone



5.5 Implementácia React Three Fiber

Ako už názov napovedá, React Three Fiber (R3F) je renderer postavený na Three.js určený pre použitie s Reactom. Hoci aj A-Frame používa Three.js, v predchádzajúcich prototypoch sme s ním nikdy nepracovali priamo – až doteraz.

Architektúra a komponenty

React podporuje JavaScript aj TypeScript, no v tomto projekte sme ostali pri JavaScripte.

Po načítaní hlavného skriptu v HTML súbore môžeme spustiť renderer, ktorý vykreslí náš hlavný komponent `App`. Kľúčovým prvkom v R3F je komponent `<Canvas>`, do ktorého vkladáme všetky prvky scény, ktoré chceme vykresliť.

V rámci komponentu `<Canvas>` sa nachádzajú komponenty ako vlastné ovládanie, samotná hra MineSweeper, nastavenie scény (objekty, podlaha, svetlo) a XR podpora.

Použité komponenty v projekte:

- mineSweeper,
- Tile,
- LeftController, RightController,
- PlayerControls, VRControls,
- Komponenty pre modely.

Použité balíky a zdroje

Zoznam použitých knižníc:

- XR[16] – podpora pre VR,
- drei[17] – kolekcia hotových komponentov a abstrakcií,
- react-three-rapier[18] – fyzikálny engine.

[16]Knižnica XR - Dostupná na: https://github.com/pmndrs/xr
[17]Knižnica drei - Dostupná na: https://github.com/pmndrs/drei
[18]Knižnica react-three-rapier - Dostupná na:: https://github.com/pmndrs/react-three-rapier

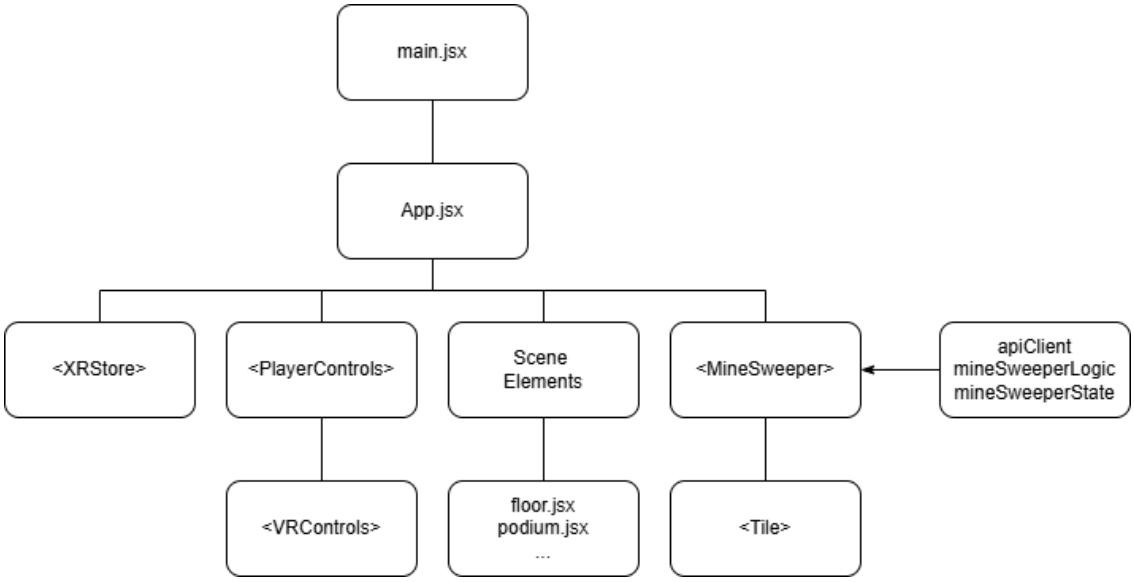
Pohyb hráča pre počítače a VR využíva implementáciu z tohto článku [13], pričom bol doplnený o ovládanie myšou z tohto príkladu[19], ktorý je k dostupný online.

Modely v `.glb/.gltf` formáte boli konvertované na komponenty pomocou nástroja `gltfjsx`[20].

Detaily implementácie

Štruktúra a kľúčové zložky programu sú znázornené na obrázku 5.10, ktorý slúži na nasledujúce vysvetlenie.

Obrázok 5.10: Zjednodušená štruktúra súboru App.jsx v R3F



mineSweeper, Tile

Komponent `mineSweeper` je hlavnou súčasťou hry Minesweeper. Využíva spoločnú hernú logiku, vytvára herné pole a spracováva interakciu s dlaždicami. Funguje obdobne ako v predchádzajúcich prototypoch.

Minesweeper tiež kontroluje, či kolízne políčka umiestnené na modeloch, ktoré nahrádzajú ovládače VR, s niečím kolidujú, a po splnení všetkých podmienok zavolá príslušnú funkciu. Používa tiež komponentu `Tile`, ktorý sa stará o interakciu a vizuálne zmeny jednotlivých políčok.

[19]Ukážka R3F v CodeSandbox - Dostupná na: https://codesandbox.io/p/sandbox/r3f-wasd--controls-wft0n?file=%2Fsrc%2FLookControls.js%3A17%2C41

[20]Nástroj gltfjsx - Dostupná na: https://github.com/pmndrs/gltfjsx

PlayerControls, VRControls, LeftController, RightController

`PlayerControls` je vlastný komponent pre ovládanie postavy. Obsahuje pohyb pomocou kláves (WASD), skákanie a otáčanie pomocou ťahania myšou.

`VRControls` zabezpečuje pohyb v prostredí VR. Využíva funkciu useXR-ControllerLocomotion z XR doplnku.

Komponenty `LeftController` a `RightController` slúžia na zmenu vzhľadu VR ovládačov a spracovanie stlačenia triggerov, pričom vyvolávajú príslušnú akciu.

Modely

Každý použitý model v scéne má vlastný komponent, čo uľahčuje jeho použitie a pozíciu v scéne.

Ovládanie a interakcia

R3F nepodporuje natívne ovládanie ako v predchádzajúcich prototypoch, preto sme museli implementovať vlastné. Našťastie, existujú online zdroje, ktoré sme mohli upraviť podľa potreby.

Zatiaľ čo pôvodné ovládanie fungovalo na desktop verzii, v prostredí VR nastali problémy, ktoré si vyžiadali výraznú úpravu pohybu hráča. Tento problém bol vyriešený s použitím knižnice `react-three-rapier`, ktorá poskytuje fyzikálny engine.

Vďaka tomu je hráč schopný skákať a interagovať s objektmi pomocou gravitácie, čo vytvára realistickejší pohyb.

Vizualizácia a kompozícia scény

Scéna sa nastavuje podobne ako v Babylone – definujeme vlastnú kameru, osvetlenie, podlahu, skybox a ďalšie prvky.

Skybox je implementovaný komponentom `<Sky>` z knižnice `drei`, ktorý si vieme prispôsobiť podľa potreby.

Objekty umiestňujeme jednoducho – stačí použiť komponent a určiť jeho pozíciu.

Pri nastavovaní tieňov je potrebné špecifikovať, ktoré svetlo má vrhať tiene a na ktoré objekty.

VR konfigurácia

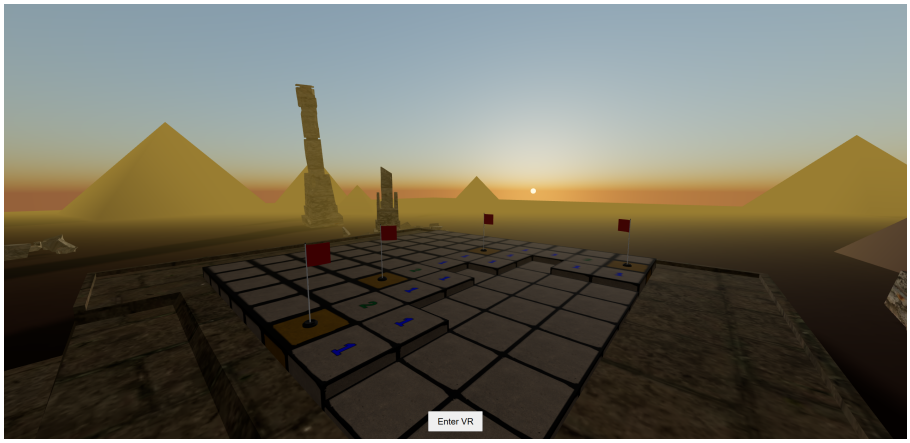
Kým na monitore nepresná poloha kamery nie je zásadná, vo VR je dôležité, aby sa hráč mohol dostať nad herné pole. Predvolený teleport vo VR spôsoboval problémy – indikátor sa mohol prekrývať s objektami, čo často viedlo k tomu, že sa hráč ocitol v ich vnútri.

VR podpora je realizovaná pomocou funkcie `createXRStore`, ktorej výstup vložíme do komponentu `<XR>` v rámci `<Canvas>`.

V rámci komponentu `<XR>` môžeme nastaviť napríklad umiestnenie ovládačov v emulačnom režime a referencie na ne. V našom prípade sme museli prispôsobiť modely oboch ovládačov a zabezpečiť, aby po stlačení triggera vykonali príslušnú akciu.

Hitboxy boli pridané na vhodné časti modelov ovládačov. Každý frame sa kontroluje, či hitbox koliduje s niektorou z dlaždíc. Táto kontrola sa nachádza v komponente `mineSweeper`, ktorý má referenciu na všetky dlaždice.

Obrázok 5.11: Hotová scéna v R3F



6 Vyhodnotenie

V tejto kapitole sa pokúsime zhodnotiť každý framework prezentovaním našich pozorovaní a zodpovedaním niekoľkých usmerňujúcich otázok.

- Aká jednoduchá bola samotná vývojová fáza?
- Bola dokumentácia alebo komunita nápomocná?
- Ako bola zabezpečená kompatibilita s VR?
- Podporuje framework kvalitnejšiu architektúru systému?

A-Frame

A-Frame bol pravdepodobne najprívetivejší framework na používanie. HTML štruktúra s použitím značiek na pridávanie komponentov a objektov do scény bola prehľadne jednoduchá. ECS architektúra však môže byť spočiatku mätúca.

Aká jednoduchá bola samotná vývojová fáza?

Vytvorenie scény nevyžadovalo veľa úsilia a dokonca aj jej úpravy boli jednoduché vďaka vstavanému editoru scén, ktorý bol zároveň veľmi užitočný aj pri ladení.

Implementácia hernej logiky pre Minesweeper nepredstavovala žiadny problém.

Zvláštnosťou bolo, že ak ste chceli zavolať funkciu z iného komponentu, bolo potrebné najprv vyhľadať daný element v DOM (alebo získať komponent rodiča aktuálneho elementu) a až potom zavolať požadovanú funkciu, keďže priamy prenos referencií medzi komponentmi je v dokumentácia A-Frame skôr neodporúča.

Bola dokumentácia alebo komunita nápomocná?

Oficiálna dokumentácia A-Frame je veľmi kvalitná. Nájdete v nej všetky systémy, komponenty a primitíva, ktoré framework poskytuje, spolu s príkladmi ich použitia. Je to tiež open-source projekt, takže ak je to potrebné, môžete si jednotlivé systémy upraviť.

Keďže v systéme A-Frame môžeme vytvárať vlastné komponenty, znamená to, že si môžeme stiahnuť komponent, ktorý je práve pripravený na použitie v našom projekte. Toto prostredie sa nakoniec skopírovalo všade inde.

Ako bola zabezpečená kompatibilita s VR?

A-Frame bol navrhnutý s ohľadom na VR, takže každá scéna vytvorená vo frameworku je pripravená na VR použitie. Neobsahuje síce vstavané ovládanie pohybu alebo interakciu s ovládačmi, no opäť to dokáže vyriešiť komunita alebo vlastné riešenia.

Podporuje framework kvalitnejšiu architektúru systému?

Samotné nastavenie frameworku podporuje štruktúrovanie kódu tak, že sa logika komponentu uchováva v samostatnom súbore.

Aj keď je možné prehľadne oddeliť vizuálne prvky a logiku, úpravy vizuálov môžu byť dosť zdĺhavé (napr. cez ‘object.setAttribute(... , ...)‘) a už spomínané prehľadávanie DOM-u či získavanie komponentu nadradeného prvku môžu viesť k neprehľadnému kódu.

Babylon.js

Babylon je výkonný framework, no pracuje na nižšej úrovni než ostatné spomínané nástroje. Poskytuje množstvo možností, ako pristupovať k tvorbe rôznych projektov – od produktových prezentácií až po hry.

Aká jednoduchá bola samotná vývojová fáza?

Zatiaľ čo vytvorenie počiatočnej scény nie je také jednoduché ako v A-Frame, Babylon ponúka užitočnú sadu nástrojov pre neskoršie fázy vývoja aplikácie. Programátor musí vytvoriť scénu a odovzdať ju rendereru, ktorý následne preberá kontrolu nad vykresľovaním.

Používanie 3D modelov však nie je úplne priamočiare. Babylon od nás vyžaduje najprv vytvorenie kontajnera (ideálne pomocou asynchrónnej funkcie), ktorý obsahuje všetky informácie o modeli. Až po jeho načítaní môžeme model

pripojiť do scény a následne s ním pracovať.

Modely nie sú spájané do jedného objektu. Ak teda náš model pozostáva z viacerých častí, akákoľvek zmena sa musí aplikovať na všetky jeho podobjekty.

Materiály použité v modeli sa ukladajú do zoznamu materiálov. Duplikácia modelu zároveň vytvorí aj kópie materiálov. Tento zoznam však neobsahuje informáciu o tom, ku ktorému modelu daný materiál patrí, čo pri ladení spôsobuje problémy – pokiaľ si každý materiál a model osobitne nepomenovávame.

Tieto vlastnosti môžu prácu uľahčiť, ak chceme pracovať s animáciami, pokročilými materiálmi či kostrami, ale zároveň pridávajú veľa krokov navyše.

Bola dokumentácia alebo komunita nápomocná?

Oficiálna dokumentácia je porovnateľná s dokumentáciou pre A-Frame. Babylon používa vlastný rendering engine, no napriek tomu nie je dokumentácia príliš technická – obsahuje funkčné príklady, ktoré priamo ukazujú, ako jednotlivé funkcie používať.

Aj keď Babylon neponúka toľko komunitných rozšírení ako iné frameworky, stále má aktívnu komunitu, ktorá vytvára obsah a návody.

Ako bola zabezpečená kompatibilita s VR?

Babylon má natívnu podporu pre VR (aj XR). Základná VR kompatibilita sa dá pridať do aplikácie veľmi jednoducho, pričom rámec už obsahuje aj pohyb (locomotion) a interakciu s objektmi s minimálnou konfiguráciou.

Vytváranie vlastných interakcií a ovládačov bolo o niečo náročnejšie, no vzhľadom na množstvo logiky, ktorú dokáže Babylon automaticky spracovať, je výsledný kód prekvapivo stručný.

Podporuje framework kvalitnejšiu architektúru systému?

Babylon poskytuje programátorovi úplnú slobodu a nepodporuje žiadny konkrétny architektonický vzor, takže organizácia kódu je úplne na nás. To môže byť výhodou pre skúsených vývojárov, no zároveň nočnou morou pre začiatočníkov.

React Three Fiber (R3F)

R3F je renderer pre React, ktorý využíva Three.js, čo znamená, že celá aplikácia má štruktúru Reactu. Niektoré časti prototypov sa implementovali jednoduchšie než iné.

Aká jednoduchá bola samotná vývojová fáza?

Rovnako ako pri Babylone, našou hlavnou úlohou bolo vytvoriť scénu, ktorú sme následne odovzdali rendereru, ktorý sa postaral o zvyšok. Tvorba scény bola pomerne jednoduchá – rozdelenie modelov do vlastných komponentov umožnilo hladkú výstavbu scény a integrácia hry bola priamočiara.

Problémy však nastali pri implementácii pohybu hráča (v bežnom aj VR režime). R3F neobsahuje žiaden predvolený spôsob pohybu pomocou kláves WASD a myši, takže sme si museli vytvoriť vlastné riešenie. Pohyb mohol byť vyriešený pomocou "plávajúcej kamery", ako to bolo použité doteraz, no kvôli nekonzistencii ovládania vo VR (vysvetlené v nasledujúcej časti) pri pohybe na objektoch sme museli zvoliť iný prístup. To nakoniec viedlo k tomu, že hráč bol ovplyvnený gravitáciou a získal možnosť skákať.

R3F taktiež neponúka žiadny vstavaný editor scény v reálnom čase.

Bola dokumentácia alebo komunita nápomocná?

Oficiálna dokumentácia R3F bola najslabšia zo všetkých porovnávaných frameworkov. Keďže R3F je iba renderer, väčšina kódu je vlastne React. Táto dokumentácia môže byť postačujúca pre skúsených vývojárov, ktorí už poznajú React, ale začiatočníkom veľa nepomôže.

Podpora VR je v R3F zabezpečená komunitná knižnica XR, ktorá je samo osebe rozšírením pre React. Dokumentácia sa tak rozdelila na viaceré časti a nebolo vždy zrejmé, kde hľadať konkrétne informácie. Aj dokumentácia k samotnému XR rozšíreniu často nebola dostatočná.

Komunitná podpora je prijateľná. Spoločnosť Meta odviedla kus práce na základnej infraštruktúre a online možno nájsť viaceré komunitné príklady. Keďže R3F využíva Three.js, mnohé problémy sa dajú vyriešiť pomocou dostupných informácií o Three.js, ktorý má vďaka svojej dlhoročnej existencii veľkú dokumentačnú základňu.

Ako bola zabezpečená kompatibilita s VR?

VR nie je v R3F natívne podporované, preto sme použili komunitné knižnica XR. Táto knižnica síce plní svoj účel a jej implementácia je pomerne jednoduchá, no má svoje problémy.

Okrem slabej dokumentácie bol najväčší problém v pohybe hráča. Doteraz bol pohyb hráča zabezpečený pomocou joysticku, ktorý zobrazil lúč – ukazovateľ toho, kam a v akom smere sa používateľ teleportuje. Táto funkcionalita existuje aj v R3F, no ukazovateľ pre teleportáciu sa správal nekonzistentne – často prechá-

dzal cez objekty. To spôsobovalo problémy, napríklad pri snahe dostať sa na vrch objektov alebo pohybovať sa po nich, keďže používateľ sa mohol ocitnúť vo vnútri objektu.

Podporuje framework kvalitnejšiu architektúru systému?

Pri dodržiavaní osvedčených praktík v Reacte nám R3F umožňuje vytvoriť aplikáciu s prehľadnou štruktúrou a čitateľnosťou. Vďaka deklaratívnej povahe Reactu R3F abstrahuje imperatívne spravovanie scény – napríklad načítanie modelu cez JSX a jeho priame používanie ako komponentu v plátne.

7 Záver

Táto práca sa zaoberala vývojom webových aplikácií pre virtuálnu realitu navrhnutím a implementáciou série prototypov pomocou rôznych frontendových frameworkov. Cieľom bolo zhodnotiť aktuálny stav softvérových nástrojov dostupných pre tento účel.

Každý prototyp bol rozdelený na dve časti: backend (alebo server) a frontend (klient). Všetky prototypy využívali rovnaký server, ktorý zabezpečoval logiku hry Minesweeper. Klientská strana taktiež zdieľala spoločný kód pre komunikáciu so serverom a vlastnú lokálnu logiku, v prípade, že by sme server nechceli využiť.

Počas práce boli hodnotené viaceré frameworky, konkrétne A-Frame, React Three Fiber a Babylon.js. Každý z nich bol subjektívne vyhodnotený z hľadiska zložitosti, dokumentácie, podpory VR a architektonickej prehľadnosti. Zjednotený backend napísaný v jazyku Java poskytoval konzistentné rozhranie hernej logiky prostredníctvom REST API.

Ciele stanovené na začiatku práce sa podarilo úspešne splniť. Všetky prototypy boli implementované podľa jednotného návrhového plánu a výsledky umožnili identifikovať silné a slabé stránky jednotlivých frameworkov v kontexte vývoja webových hier s podporou virtuálnej reality.

Napriek rozsahu práce sa niektoré aspekty nepreskúmali. Výkonové rozdiely medzi frameworkami sa nehodnotili, pretože scény boli príliš jednoduché a líšili sa spôsobom implementácie. Téma multiplayeru bola taktiež zaujímavá, no nebola predmetom skúmania.

Ako ďalší krok by budúca práca mohla rozšíriť prototypy o zložitejšie herné mechaniky alebo implementovať multiplayer. Vhodné by bolo aj preskúmať tvorbu scén pre Augmented Reality (AR), ako pokračovanie tejto práce.

Zoznam použitej literatúry

1. *LOW-LEVEL 3D GRAPHICS API BASED ON OPENGL ES* [online]. Khronos group [cit. 2024-10-26]. Dostupné z: https://www.khronos.org/webgl/.
2. *Stack Overflow developer survey* [online]. Stack Overflow [cit. 2024-10-26]. Dostupné z: https://survey.stackoverflow.co/2023/#section-admired-and-desired-other-tools.
3. *PICO developer WebXR Concepts* [online]. PICO [cit. 2024-10-26]. Dostupné z: https://developer.picoxr.com/document/web/webxr-performance/.
4. *WebXR Device API explained* [online]. Stack Overflow [cit. 2024-10-26]. Dostupné z: https://immersive-web.github.io/webxr/explainer.html#lifetime-of-a-vr-web-app.
5. HAAS, Andreas; ROSSBERG, Andreas; SCHUFF, Derek L.; TITZER, Ben L.; HOLMAN, Michael; GOHMAN, Dan; WAGNER, Luke; ZAKAI, Alon; BASTIEN, JF. Bringing the web up to speed with WebAssembly. *SIGPLAN Not.* 2017, roč. 52, č. 6, s. 185–200. ISSN 0362-1340. Dostupné z DOI: 10.1145/3140587.3062363.
6. DIRKSEN, Jos et al. *Three. js essentials*. Packt Publishing, 2014.
7. *A-Frame official webpage* [online]. Supermedium,Mozilla VR team [cit. 2024-11-27]. Dostupné z: https://aframe.io/.
8. NEELAKANTAM, Srushtika; PANT, Tanay. Introduction to A-Frame. In: *Learning Web-based Virtual Reality: Build and Deploy Web-based Virtual Reality Technology*. Berkeley, CA: Apress, 2017, s. 17–38. ISBN 978-1-4842-2710-7. Dostupné z DOI: 10.1007/978-1-4842-2710-7_4.
9. MOREAU-MATHIS, Julien. *Babylon. js Essentials*. Packt Publishing Ltd, 2016.

10. *Babylon.js documentation: Real time Multiplayer with Colyseus* [online]. Babylon.js [cit. 2024-11-02]. Dostupné z: https://doc.babylonjs.com/guidedLearning/networking/Colyseus.

11. *WebXR Device API* [online]. Mozilla [cit. 2025-03-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API.

12. MARTÍNEZ BAYONA, Jonàs. *Space-optimized texture atlases*. 2009. Dostupné tiež z: http://hdl.handle.net/2099.1/7720. Diz. pr. UPC, Facultat d'Informàtica de Barcelona, Departament de Llenguatges i Sistemes Informàtics.

13. F-KAITO. *React Three Fiber (Translated:) I tried making a Minecraft-like game* [online]. Qiita [cit. 2025-05-18]. Dostupné z: https://qiita.com/f-kaito/items/a68ea9fd1e5b378f178e#react-three-fiber%5C%E3%5C%81%5C%A8%5C%E3%5C%81%5C%AF.

Zoznam skratiek

API Application Programming Interface.

AR Augmented Reality.

ECS Entity Component System.

GPU Graphics Processing Unit.

VR Virtálna Realita.

WebVR Web Virtual Reality.

WebXR Web Extended Reality.

XR Extended Reality.

Zoznam príloh

Príloha A Používateľská príručka

Príloha B CD médium – záverečná práca v elektronickej podobe,

Príloha C Repozitár výsledného projekty sú dostupný na adrese
`https://git.kpi.fei.tuke.sk/kpi-zp/2025/bp.adam.dargoczi/bp-aframe`
`https://git.kpi.fei.tuke.sk/kpi-zp/2025/bp.adam.dargoczi/bp-babylon-js`
`https://git.kpi.fei.tuke.sk/kpi-zp/2025/bp.adam.dargoczi/bp-r3f`

A Používateľská príručka

A.1 Požiadavky

Pre spustenie verzií aplikácie vytvorených pomocou A-Frame, Babylon.js a React Three Fiber (R3F) je potrebný Node Package Manager (npm). Inštalátor Node.js je dostupný na [21].

Po nainštalovaní Node.js musíme nainštalovať správcu balíčkov npm pomocou príkazu:

```
npm install -g npm
```

Na spustenie backend servera implementovaného pomocou Spring Boot je potrebné mať nainštalované JDK, ktoré je dostupné na [22]. Zdrojový kód používa Apache Maven, no už prekompilovaná verzia sa nachádza v repozitári v zložke: `/release/minesweeperServer.jar`.

A.2 Spustenie a konfigurácia aplikácií

Po klonovaní repozitára A-Frame, Babylon.js alebo R3F je najskôr potrebné nainštalovať všetky závislosti pomocou príkazu:

```
npm install
```

A-Frame

Aplikáciu A-Frame spustíme pomocou príkazu:

```
npm start
```

Týmto sa spustí server a otvorí sa okno prehliadača na adrese `127.0.0.1:8000`. V konzole sa zobrazia aj ďalšie dostupné adresy.

[21] Stránka Node.js - Dostupná na:https://nodejs.org/en/download/
[22] Stránka Java JDK - Dostupná na:https://jdk.java.net/

Poznámka: Na backend serveri sú povolené iba adresy
`127.0.0.1:8000` a
`localhost:8000`.

Na konfiguráciu veľkosti hracieho poľa, počtu mín, veľkosť jednej dlaždice a konfiguráciu offline režimu, je potrebné upraviť súbor `index.html`, konkrétne tag:

```
<a-entity id="field" minesweeper="size: (int); mines:
(int); tileSize: (int); offline: (boolean)"></a-entity>
```

Babylon, R3F

Pre verzie Babylon.js alebo R3F použijeme príkaz:

```
npm run dev
```

Týmto sa spustí server, ktorý beží predvolene na adrese `localhost:5173`.

Konfigurácia pre Babylon.js: V súbore `main.js` upravte riadok:

```
const app = new BabylonApp(...);
```

Konštruktor `BabylonApp` prijíma tri parametre: veľkosť poľa (int), offline režim (boolean) a počet mín (int).

Konfigurácia pre R3F: V súbore `App.jsx` nájdeme komponent `<MineSweeper>`, kde môžeme upraviť offline režim (boolean) a počet mín (int).

Java server

Po otvorení repozitára môžeme spustiť backend server príkazom:

```
java -jar release/minesweeperServer.jar
```

Server bude dostupný na adrese `localhost:8080` a jeho rozhranie môžeme preskúmať pomocou Swagger-ui na:

```
http://localhost:8080/swagger-ui/index.html
```

A.3 Cieľ hry

Hráč musí odkryť všetky políčka bez mín na hernom poli hry Minesweeper.

- Míny sú rozmiestnené náhodne, ich počet je možné konfigurovať.
- Číselné nápovedy zobrazujú počet mín v susedstve daného políčka.
- Otvorenie políčka s mínou vedie k prehre.

Hru je možné resetovať obnovením stránky.

A.4 Ovládanie cez klávesnicu a VR

Na vstup do režimu VR musí mať používateľ podporované zariadenie. Po jeho detekovaní je možné vstúpiť do VR kliknutím na tlačidlo „VR“, „Enter VR“ alebo ikonu headsetu.

A-Frame

Nasledujúca tabuľka zhrňuje ovládanie verzie **A-Frame**:

Akcia	Klávesnica	VR
Pohyb	WASD	L+R Joystick
Otvoriť	Ľavé tlačidlo	RT (pravý trigger)
Označiť	E	LT (ľavý trigger)
Mierenie	Zameriavač	Objekt

Označovanie sa vykonáva spodkom vlajky v ľavej ruke pri stlačení triggeru. Otváranie políčka prebieha vrchom palice v pravej ruke pri kolízii s políčkom a stlačení triggeru.

Babylon

V **Babylon** verzii zostáva ovládanie podobné ako v A-Frame, no umožňuje aj klikanie na políčka pre ich otvorenie.

Akcia	Klávesnica	VR
Pohyb	WASD	L+R Joystick
Otvoriť	Ľavé tlačidlo	RT (pravý trigger)
Označiť	Pravé tlačidlo	LT (ľavý trigger)
Mierenie	Kurzory	Objekt

R3F

Vo verzii R3F bola pridaná možnosť skoku.

Akcia	Klávesnica	VR
Pohyb	WASD	L+R Joystick
Otvoriť	Ľavé tlačidlo	RT (pravý trigger)
Označiť	Pravé tlačidlo	LT (ľavý trigger)
Mierenie	Kurzory	Objekt
Skok	Medzerník	Tlačidlo 'A'

Hráč musí teraz skočiť, aby prekonal prekážky.