FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Možnosti využití frameworku A-Frame pro rozšířenou realitu
Student:	Tomáš Bilák
Vedoucí:	Ing. Petr Pauš, Ph.D.
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2020/21

Pokyny pro vypracování

A-Frame je opensource framework pro virtuální (VR) a rozšířenou realitu (AR) běžící přímo ve webovém prohlížeči. Na katedře již běží několik projektů pro VR a AR pomocí knihoven závislých na dané platformě, proto by bylo vhodné analyzovat možnosti využití této knihovny pro účely multiplatformní webové demo aplikace.

1) Seznamte se s knihovnou A-Frame.
2) Analyzujte možnosti této knihovny a porovnejte s nativními knihovnami (např. AR Core nebo AR Kit).
3) Navrhněte jednoduchou aplikaci, která pomocí smíšené reality do prostoru umístí objekt (nejlépe budovu).
4) Pokuste se vytvořit scénu interaktivní, tj., uživatel může s objektem interagovat.
5) Rozeberte obecné možnosti návrhu uživatelského rozhraní pro AR aplikace a využijte v prototypu vytvořené aplikace.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 16. dubna 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Možnosti využití frameworku A-Frame pro rozšířenou realitu

Tomáš Bilák

Katedra softwárového inženýrství
Vedúci práce: Ing. Petr Pauš, Ph.D.

16. mája 2019

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 16. mája 2019 .....

Odkaz na túto prácu

Bilák, Tomáš. *Možnosti využití frameworku A-Frame pro rozšířenou realitu.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Účelom tejto bakalárskej práce je analyzovať framework A-Frame a popísať jeho možnosti využitia pre aplikáciu v rozšírenej realite. Práca sa zameriava predovšetkým na rozšírenú realitu pre mobilné zariadenia bez závislosti na platforme. Cieľom je navrhnúť multiplatformné riešenie dostupné na starších zariadeniach, ktoré nepodporujú moderné knižnice, ako ARCore a ARKit, závislé na platforme. Ďalej sa práca venuje všeobecným pravidlám vytvárania používateľského rozhrania pre aplikácie v rozšírenej realite, ktoré následne aplikuje na vytvorený prototyp.

Klíčová slova A-Frame, rozšírená realita, Three.js, AR.js, UI, UX

Abstract

The purpose of this bachelor thesis is to analyze framework A-Frame and outline its possibilities of use for application in augmented reality. Thesis main interest is platform-independent augmented reality for mobile devices. Aim is to design cross-platform solution available on older devices, which do not support modern platform specific frameworks, such as ARCore and ARKit. Further thesis addresses the general rules of creating user interface for applications in augmented reality and later applies them in a developed prototype.

Keywords A-Frame, augmented reality, Three.js, AR.js, UI, UX

Obsah

Zoznam obrázkov

Zoznam kódov

Úvod

V oblasti zaznamenávania udalostí a zobrazovania obrazu sme prešli veľkým vývojom. Od pravekých jaskynných malieb, keď ľudia chceli zachovať spomienku na výnimočný úlovok, neskoršie zdokonalenie maliarstva a sochárstva využitím nových nástrojov, cez zárodky analógovej a digitálnej fotografie, až po dnešné digitálne „sochárstvo a maliarstvo“, či priamu reprodukciu trojrozmerným (3D) skenovaním a následným vytlačením na 3D tlačiarni. Digitálne zachytený objekt si však vieme zobraziť aj vo virtuálnej realite (VR) alebo rozšírenej realite (AR). Môžeme si tak prejsť trebárs pamiatky z historického obdobia. Napríklad v prípade požiaru v katedrále Notre Dame si môžeme pozrieť katedrálu pred nehodou vo VR videu, či prejsť sa priamo po katedrále v hre Assassin's Creed Unity, kde sa nachádza jej približná podoba z roku 1789. [4, 5] Táto práca sa bude zaoberať práve zobrazením budov v rôznych obdobiach v AR pomocou frameworku A-Frame.

Čo je to rozšírená realita

Prv než sa pustíme do analýzy samotného frameworku A-Frame, je potrebné definovať čo to vlastne tá rozšírená realita je. Aký je rozdiel medzi AR a VR. Čo je to zmiešaná realita a ako to s AR súvisí.

„An enhanced version of reality where live direct or indirect views of physical real-world environments are augmented with superimposed computer-generated images over a user's view of the real-world, thus enhancing one's current perception of reality.“ [6] Táto definícia by mohla byť preložená ako „Vylepšená verzia reality, v ktorej je živý, priamy alebo nepriamy pohľad na reálne prostredie doplnený prekrytím používateľovho pohľadu na svet, počítačovo-generovaným obrazom, čím vylepší jeho súčasné vnímanie reality.“ Hirokazu Kato popisuje AR jednoduchšie, tzv. funkčnou definíciou, ktorá v preklade znie takto: „Virtuálne objekty alebo anotácie môžu prekrývať reálny svet, akoby existovali.“ [7]

Často sa môžeme stretnúť tiež s pojmom zmiešaná realita. Zmiešaná realita však popisuje časť kontinua medzi prirodzeným prostredím a virtuálnym prostredím. Na obrázku 0.1 môžeme vidieť, že AR je vlastne podmnožinou zmiešanej reality. Pre viac informácii o navrhnutom rozdelení kontinua medzi realitou a virtualitou odporúčame [1, 8]. Zjednodušene môžeme rozdeliť kontinuum nasledujúc.

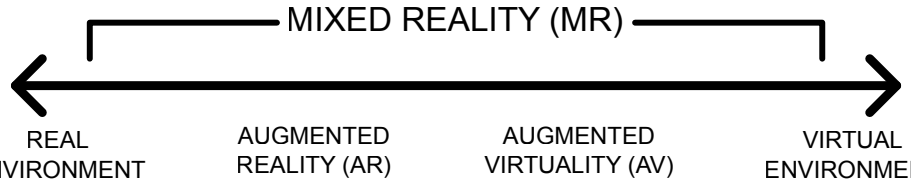
Kontinuum medzi realitou a virtualitou

Reálne prostredie Kompletne prirodzená scéna bez prvkov generovaných počítačom

Rozšírená realita Do obrazu z reálneho sveta sú vložené prvky generované počítačom

Rozšírená virtualita Do prostredia generovaného počítačom sú vkladané reálne objekty

Virtuálne prostredie Scéna je kompletne generovaná počítačom



Obr. 0.1: Kontinuum medzi skutočným prostredím a virtuálnym prostredím [1]

Samotnú podskupinu AR môžeme ďalej kategorizovať podľa princípu, ktorý využíva. [6]

Princípy využívané v AR

- Lokalizácia pomocou značiek (marker)
- Lokalizácia v priestore (GPS, WPS, BT, ai.)
- Projekcia rozšírenia
- Detekcia objektov/prostredia

Lokalizácia pomocou značiek je založená na vyhľadávaní konkrétnych útvarov (značiek), vďaka ktorým vie určiť polohu a rotáciu virtuálneho objektu v závislosti na polohe kamery. Značky sú najčastejšie jednoduché čierno-biele štvorce s unikátnym vzorom v strede. Ich vyhľadávanie v obraze nie je tak náročné. Súčasne sa v obraze môže nachádzať aj väčšie množstvo značiek. Nevýhodou značiek je, že pre generovanie virtuálneho obrazu funguje iba v prípade, že je značka v zachytenom obraze. Často pre dostatočnú registráciu značky musí byť zachytená celá plocha značky. Možným obmedzením sú i svetelné podmienky a uhol, pod ktorým značku sledujeme.

Ďalším princípom je lokalizácia v priestore pomocou GPS. Vďaka presnej polohe na zemi môžeme zobraziť napríklad anotáciu k nejakému významnému bodu, lietadlám, či zobraziť súhvezdia na oblohe v závislosti na polohe pozorovateľa. V prípade použitia v interiéri, kde má technológia GPS problémy s presnosťou, môže byť použitá iná technológia na lokalizovanie zariadenia, zväčša využívajúca triangulóciu signálu, čas letu, uhol príchodu alebo ich kombinácia.

Princíp projekcie rozšírenia na bežné predmety doplní projekciou prvky, s ktorými môže používateľ následne pracovať. V tomto systéme je interakcia vyhodnocovaná ďalším senzorom, ktorý rozlišuje určité gestá, pohyby alebo haptické vstupy. Ako príklad môžeme uviesť digitálnu klávesnicu.

Posledným spomínaným princípom je detekcia objektov alebo prostredia. Ide o problém simultánnej lokalizácie a mapovania (SLAM). Oproti hľadaniu konkrétnych značiek je to výpočtovo omnoho zložitejší problém. V ARKit a ARCore je riešením vizuálna zotrvačná odometria (VIO), resp. súbežná odometria a mapovanie (COM). Viac o daných riešeniach spomenieme v podkapitole 2.2.2, prípadne odporúčame čitateľovi prečítať [9, 10, 11, 12].

Ďalej Milgram a spol. [1] delia AR podľa displeja, ktorým je AR zobrazovaná.

AR podľa použitého displeja

- „Priehľadný“ displej
- Zobrazenie na monitore

Prvou kategóriou je priehľadný displej, ktorý má byť viac pohlcujúci pre používateľa. Využíva polopriehľadné zrkadlá, na ktoré premieta generovaný obsah. Pozorovateľ tak vidí ako obraz za zrkadlom, tak generovaný obraz. Väčšinou sa tieto displeje používajú na HMD. Počítačom vytvorený obsah sa tak priamo spája so sledovaným prostredím bez mediátora ako v druhej spomínanej metóde.

Zobrazenie na monitore je tiež nazývané „Okno do sveta“ alebo nepohlcujúce zobrazenie. Ide o zobrazenie nahrávky alebo živého obrazu z kamery, do ktorého je doplnený počítačovo vykreslený predmet. Mobilné aplikácie využívajú práve tento spôsob, preto sa ďalej v práci budeme venovať iba tejto možnosti.

História AR

Počiatky rozšírenej reality (AR) siahajú až do druhej polovice 60. rokov. Ivan Sutherland na Harvardskej univerzite položil základný kameň pre vznik súčasných head-mounted displejov (HMD). Zariadenie, veľmi zjednodušene, funkciou podobné dnešným Hololens, bolo v tom čase samozrejme neprenosné, kvôli svojej veľkosti. Sutherlandov projekt bol prezývaný Damoklov meč [13], pretože

bol, pre svoju veľkú váhu, ukotvený nad používateľom. Ako ďalej Sutherland vo svojej práci uvádza, toto zariadenie malo značné obmedzenia. Pracovný priestor, v ktorom systém dokázal registrovať posun a rotáciu zariadenia, bol valec s priemerom necelé 2 meter a vysoký približne 1 meter. Sklon hlavy hore a dole bol limitovaný iba na 40 stupňov. Ďalším obmedzením boli zobrazovacie možnosti zariadenia. O vykresľovanie obrazu sa starali malé katódové žiarivky, ktorými boli vykresľované čiarové modely. Dokázali vykresliť 3000 čiar pri frekvencii 30 snímkov za sekundu. [2] V danom období to bol však veľký počin.

Obr. 0.2: Prvý HMD pre AR, projekt „Damoklov meč“ [2]

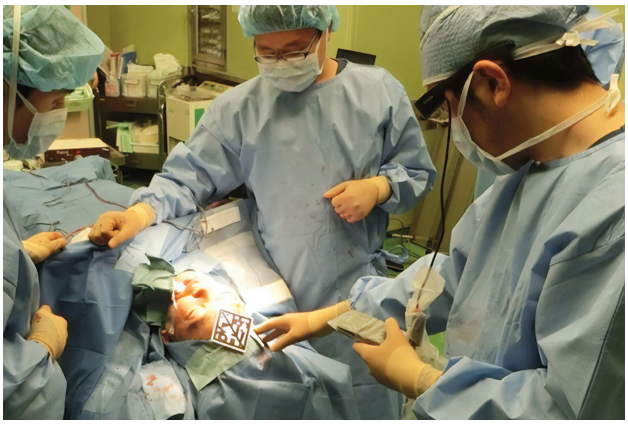
Ďalším veľkým krokom vo vývoji AR bolo vytvorenie ARToolKitu a jeho neskoršie zverejenenie pod open source LGPLv3 licenciou. Kato a Billinghurst v roku 1999 vytvorili ARToolKit. Vďaka ich SDK, ktorý je použiteľný na viacerých platformách, sa značne zjednodušil vývoj AR aplikácii. Neskôr v 2015 bol úplne zverejnený, čo pomohlo jeho pretvoreniu do javascriptu, čím sa spopularizoval i vo webových aplikáciach. [14]

Využitie AR v súčastnosti

Dnes si už mnohokrát ani neuvedomujeme, že používame nejakú AR aplikáciu. Znie to zvláštne, pretože väčšinou si pod AR aplikáciou predstavíme napríklad hru alebo niečo interaktívne. V skutočnosti je AR aplikáciou napríklad i parkovacia kamera. Kamera sníma reálnu situáciu za autom a palubný počítač na základe natočenia kolies zobrazuje budúcu trajektóriu auta. Ďalším bežným

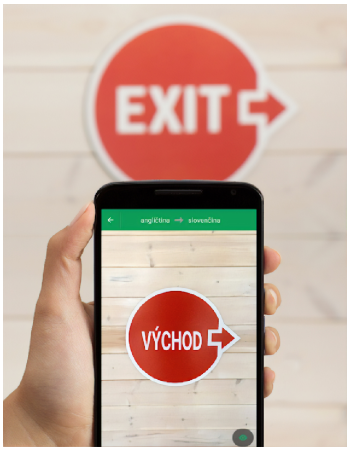
príkladom je aplikácia Google Translate. Stačí snímať akýkoľvek text v jednom z 29 podporovaných jazykov. Aplikácia text preloží a nahradí ho na obrazovke namiesto pôvodného textu.

(a) Lezecká stena ValoClimb [15]



(b) Kalibrácia značky počas chirurgického zákroku [16]

(c) Cielená reklama vo futbalovom zápase (Prebraté z youtube.com, ADI.tv, 2017)



(d) Google translate AR (Prebraté z play.google.com, Google LLC, 2019)

Obr. 0.3: Príklady využitia AR

V zábavnom priemysle sú aplikácie v AR čoraz populárnejšie. Niantic je jednou z najznámejších spoločností vytvárajúcich AR hry. Vytvorili známu hru Ingress a neskôr, v spolupráci so spoločnosťou Nintendo, Pokémon GO. Hra Pokémon GO mala taký úspech, že po týždni od vydania hry sa ceny akcií firmy Nintendo zdvojnásobili. [17] Hry však nemusia byť čisto digitálne. Pekným príkladom za všetky je hra zvaná „Augmented Climbing Wall“, ktorá spája fyzickú námahu na lezeckej stene s interaktívnou projekciou na samotnú stenu. [15]

Televízne prenosy sú často doplnené o rôzne doplňujúce informácie, ktoré môžeme považovať za AR. Jedno z prvých využití AR v zábavnom priemysle bolo v roku 1998 práve pri prenose zápasu v americkom futbale, kde do obrazu dopĺňali yardovú čiaru. [18] Dnes sú počas zápasu na ihrisko dopĺňané rôzne ďalšie informácie ako reklamy sponzorov, stav zápasu, ai. V plávaní, či atletike sú zvyčajne pridané do dráh mená súťažiacich.

V oblasti zdravotníctva existujú riešenia, ktoré pomáhajú chirurgom pri operáciách. Jedno z riešení pri plastických operáciách využíva okuliare kalibrované značkou. Po kalibrácii zobrazuje kosti, či povrch danej časti tela pred zákrokom. Dáta sú pripravené pred zákrokom snímaním 3D skenerom alebo počítačovou tomografiou. [16]

Kapitola 1

Cieľ práce

Základom teoretickej časti práce je analyzovať framework A-Frame a popísať jeho možnosti a obmedzenia. Rozoberieme rôzne možnosti vytvorenia AR aplikácie pomocou frameworku A-Frame a stručne porovnáme ich výhody a nevýhody. Stanovíme odporúčané postupy pri tvorbe užívateľského rozhrania (UI). V praktickej časti sa budeme venovať aplikácii, ktorá má zobrazovať budovy v AR. S budovami bude možné pracovať interaktívne. Podobný projekt je v súčastnosti vyvíjaný pre operačný systém (OS) Android. Projekt je postavený na knižnici ARCore. V tejto práci sa budeme venovať multiplatformnému riešeniu, ktoré bude záložným riešením pre zariadenia používajúce iný OS. Zároveň by aplikácia mala byť dostupná i na zariadeniach, ktoré nepodporujú knižnice ARCore a ARKit.

Kapitola 2

Analýza

Táto časť práce je rozdelená do štyroch sekcií. Postupne sa budeme venovať analýze jednotlivých stavebných kameňov výslednej aplikácie. V prvej sekcii je rozbor frameworku A-Frame, ktorý sa stará o správu médií a ich vykresľovanie. Nasleduje druhá sekcia, ktorá je venovaná AR knižniciam vo všeobecnosti a ich použitiu vo webovom prostredí. Ďalšia sekcia, v poradí tretia, sa zaoberá pravidlami a usmerneniami pri tvorbe rozhrania AR aplikácii, z veľkej časti s ohľadom na UX. V poslednej sekcii definujeme funkčné a nefunkčné požiadavky kladené na výslednú aplikáciu.

2.1 Framework A-Frame

A-Frame je webový framework, ktorý v úvode stránok dokumentácie opisujú v preklade takto: „A-Frame je webový framework pre tvorenie VR zážitkov. A-Frame bol vyvýjaný, pôvodne v Mozille, aby sa stal jednoduchým ale silným spôsobom ako vytvoriť VR obsah. Ako nezávislý open source projekt sa A-Frame rozrástol na jednu z najväčších a najprívetivejších VR komunít.“ [3] Na prvý pohľad sa môže zdať A-Frame ako značkovací jazyk alebo graf 3D scény. Nie je tomu tak. A-Frame je vďaka Entity-Component-System (ECS) architektúre deklaratívnym a štrukturálnym rozšírením pre knižnicu Three.js.

V tejto sekcii budeme čerpať prevažne z dokumentácie frameworku A-Frame, či Three.js. [3, 19]

2.1.1 ECS architektúra

Vo vývoji hier a 3D aplikácii je ECS vhodným návrhovým vzorom, pretože uprednostňuje kompozíciu pred dedením a hierarchiou. ECS prináša vysokú mieru flexibility pri definovaní objektov, vďaka využitiu menších znovupoužiteľných častí, tzv. komponentov. Zároveň odstraňuje problém dlhých zreťazených dedičností s prepletenou komplexnou funkčnosťou. Dovoľuje vytvárať čistý

návrh odstránením závislostí, zapuzdrením, modularizáciou a znovupoužiteľnosťou.

ECS architektúra sa skladá z troch častí:

Entita

Základný objekt v scéne, ktorému môžeme priraďovať jednotlivé komponenty.

Komponent

Znovupoužiteľný modul alebo dátový kontajner, ktorý pripájame k entitám, aby sme im pridali vzhľad, vlastnosti alebo funkcionalitu. Miešaním a konfiguráciou jednotlivých komponentov definujeme rôzne objekty. Komponentami implementujeme i logiku.

Systém

Poskytuje manažment a služby pre triedy komponentov. Systémy môžeme využiť pre oddelenie logiky a dát. Systém sa môže starať o logiku a komponenty plnia funkciu dátových kontajnerov.

Frameworku A-Frame reprezentuje entity HTML elementom `<a-entity>`. Jednotlivé komponenty sú entitám pridávané ako HTML atribúty. A-Frame prevedie komponenty na objekty obsahujúce dátovú schému, manažéra životného cyklu objektu a metódy komponentu. Komponenty môžeme registrovať API volaním `AFRAME.registerComponent(name, definition)`. Viac o vytváraní komponentov v časti 2.1.2. Systémy pridávame cez HTML atribúty elementu `<a-scene>`.

2.1.2 Definícia vlastných komponentov

Komponenty registrujeme volaním funkcie `registerComponent` z A-Frame API. Parametrami sú:

`name` - reťazec znakov, ktorý bude dostupný ako HTML atribút

`definition` - objekt, ktorý popisuje schému a metódy životného cykla komponentu

Schéma

Základom komponentu je definovanie jeho vlastnosti. Na to slúži schéma. Ide o objekt, ktorý definuje jednotlivé vlastnosti ako pár kľúča a hodnoty. V príklade 2.1 vidíme registráciu komponentu a v ukážke 2.2 zas jeho použitie.

Komponenty môžu pozostávať z jednej alebo z viacerých vlastností. Tie jednoduchšie popisujeme klasickým HTML atribútom. Hodnoty tých zložitejších, v HTML, nastavujeme podobne ako CSS inline štýly. V ukážke kódu 2.2 môžeme vidieť rozdiel v zápise.

```
AFRAME.registerComponent('light', {
  schema: {
    type: {default: 'point'},
    color: {default: '#FFF'}
  }
}
```

Kód 2.1: Ukážka registrácie komponentu

```
<!-- Setting single-property component 'position' -->
<a-entity position="1 2 3"></a-entity>

<!-- Setting multi-property component 'light' -->
<a-entity light="type: point; color: crimson"></a-
    entity>
```

Kód 2.2: Ukážka nastavenia hodnoty jednoduchého a zložitého komponentu

Životný cyklus komponentu

Metódy zabezpečujúce životný cyklus komponentu využívajú dáta zo schémy aby modifikovali entitu. V prehľade metód životného cykla komponentu uvádzame všetky názvy metód, kedy sú volané a ich funkciu.

Prehľad metód životného cyklu komponentu

`init`

Metóda je volaná pri inicializácii komponentu. Používa sa na nastavenie počiatočného stavu a inicializáciu premenných.

`update`

Metóda je volaná pri inicializácii a vždy pri zmene vlastností komponentu. Používa sa na modifikáciu entity.

`remove`

Metóda je volaná pri odstránení komponentu z entity alebo zo scény. Používa sa na odobranie modifikácii entity.

`tick`

Metóda je volaná pri každom cykle vykreslenia alebo tick volaní scény. Používa sa na kontinuálne zmeny a kontroly.

`tock`

Metóda je volaná po každom vykreslení. Používa sa pre post processing efekty a logiku, ktorá vyžaduje vykreslenú scénu.

`play`

Metóda je volaná pri inicializácií komponentu a vždy keď scéna alebo entita spustí pridanie nejakého správania, či dynamického, alebo na pozadí. Používa sa pre začatie alebo pokračovanie správania.

`pause`

Metóda je volaná pri odobraní komponentu z entity alebo odobraní entity zo scény a vždy keď entita alebo scéna pozastaví nejaké správanie, či dynamického, alebo na pozadí. Používa sa pre pozastavenie správania.

`updateSchema`

Metóda je volaná vždy pri zmene vlastností komponentu. Používa sa na dynamickú zmenu schémy.

Závislosti

Niekedy môže jeden komponent vyžadovať existenciu iného komponentu alebo množiny komponentov. To môžeme definovať doplnením poľa `dependencies`, v ktorom postupne vypíšeme potrebné závislosti. A-Frame sa potom postará o postupnú inicializáciu závislostí zľava. V prípade ďalších závislostí v závislom komponente doplní ďalšie závislosti do inicializačného zoznamu.

Viac inštancií

Jedna entita môže mať priradených viac inštancií jedného komponentu. Musí to byť povolené pri registrácii komponentu. Slúži na to premenná `multiple`, ktorej predvolená hodnota je `false`. Ak je povolené pridanie viacerých inštancií, môžeme v DOM rozoznávať jednotlivé inštancie prípony v tvare dvojitého podčiarkovníka a unikátneho ID (`__<ID>`).

Podstatným rozdielom je ako chceme potom s komponentom pracovať. V metódach komponentu, vieme rozlišovať medzi komponentmi pristúpením k premennej `this.id`. Ak chceme vytvoriť Three.js objekt, musíme použiť `this.attrName`, kde je uložený celý názov atributu z DOM. Je to podstatné kvôli mapovaniu v `object3DMap`.

2.1.3 Definícia primitív

Primitíva sú entity s definovanými základnými komponentmi a ich predvolenými hodnotami. Ide o pohodlné definovanie často využívaných objektov. Pri definovaní primitív musíme nastaviť entite 3 základné časti:

- Sémantické meno (napr. `<a-box>`)
- Množinu komponentov s ich predvolenými hodnotami
- Mapovanie dát komponentov na HTML atribúty

Zavolaním funkcie `registerPrimitive` z A-Frame API zaregistrujeme nové primitívum. Parameter `name` je textový reťazec, ktorý musí obsahovať pomlčku (napr. `'a-box'`). Parameter `definition` je JavaScript objekt, ktorý definuje základné vlastnosti: `defaultComponents` a `mappings`. V kóde 2.3 vidíme ukážku registrácie primitíva `<a-ocean>`

```
1  AFRAME.registerPrimitive('a-ocean', {
2    // Attaches the 'ocean' component by default.
3    // Defaults the ocean to be parallel to the ground.
4    defaultComponents: {
5      ocean: {},
6      rotation: {x: -90, y: 0, z: 0}
7    },
8
9    // Maps HTML attributes to the 'ocean' component's
       properties.
10   mappings: {
11     width: 'ocean.width',
12     depth: 'ocean.depth',
13     density: 'ocean.density',
14     color: 'ocean.color',
15     opacity: 'ocean.opacity'
16   }
17 });
```

Kód 2.3: Ukážka registrácie primitíva `<a-ocean>` [3]

2.1.4 Načítanie modelov

O správu modelov a zdrojov všeobecne sa stará `asset management system`. Na jednom mieste sú definované všetky zdroje ako modely, materiály, obrázky, zvukové nahrávky a video textúry. Stačí do scény pridať element `<a-assets>`, ktoré potomkovia sú požadovanými zdrojmi. A-Frame všetky zdroje načíta dopredu a ukladá do cache pre lepší výkon. Načítavanie zdrojov trvá maximálny čas definovaný v atribúte `timeout`. Ak sa teda nepodarí načítať zdroj do daného času, považuje sa za nedosiahnuteľný a vykresľovanie začína bez neho. Predvolená hodnota `timeout` je nastavená na 3 sekundy. Môžeme ju zmeniť, pridaním atribútu `timeout` elementu `<a-assets>`, nastavením hodnoty v milisekundách. Kvôli zlému pripojeniu alebo odozve serveru môže načítanie trvať dlhšie. Ak vieme, že aplikácia bude často využívaná v priestoroch, kde

môžu nastať tieto problémy, mali by sme nastaviť vyššiu hodnotu pre `timeout`. Ukážkový kód 2.4 využíva zmenu nastavenia `timeout` na 5 sekúnd.

```
1  <a-scene>
2    <!-- Asset management system. -->
3    <a-assets timeout="5000">
4      <a-asset-item id="horse-obj" src="horse.obj"></
         a-asset-item>
5      <a-asset-item id="horse-mtl" src="horse.mtl"></
         a-asset-item>
6      <a-mixin id="giant" scale="5 5 5"></a-mixin>
7      <audio id="neigh" src="neigh.mp3"></audio>
8      <img id="advertisement" src="ad.png">
9      <!-- Blocking call in loading of video texture with
           'preload="auto"' -->
10     <video id="kentucky-derby" src="derby.mp4" preload=
         "auto"></video>
11   </a-assets>
12
13   <!-- Scene. -->
14   <a-plane src="#advertisement"></a-plane>
15   <a-sound src="#neigh"></a-sound>
16   <a-entity geometry="primitive: plane" material="src:
       #kentucky-derby"></a-entity>
17   <a-entity mixin="giant" obj-model="obj: #horse-obj;
       mtl: #horse-mtl"></a-entity>
18 </a-scene>
```

Kód 2.4: Ukážka využitia `<a-assets>` na správu zdrojov [3]

Pre elementy `<audio>` a `<video>` volí A-Frame predvolene neblokovacie načítavanie. Blokovacie načítavanie môžeme vynútiť. Načítavanie zdrojov po začiatku vykresľovania by brzdilo vykresľovacie jadro a vizuálne zhoršilo zážitok používateľa, preto je vhodné používať neblokovacie načítavanie iba výnimočne. V ukážke kódu 2.4 na riadku 10 je použité blokovacie volanie načítavania video textúry.

Pre multimediálne zdroje, či podporné skripty, je vhodné využívať siete na doručovanie obsahu (CDN). Z bezpečnostného hľadiska, keďže ide o zdieľanie zdrojov z iného pôvodu (CORS), je potrebné posielať špeciálnu XMLHttp požiadavku (XHR), ktorá obsahuje CORS hlavičku. Zabezpečuje to tiež `asset management system`.

Ak v zdrojoch uvádzame `a-asset-item`, A-Frame zavolá `FileLoader` z knižnice `Three.js`. Podľa dokumentácie `three.js` [19] ide o triedu na nízkej úrovni načítavajúcu zdroje pomocou XHR. Každý „loader“ dedí od tejto triedy. Po načítaní sú dáta uložené do `THREE.Cache`. Je to centrálna cache pre všetky „loader“ triedy.

2.1.5 Práca s DOM API

DOM API je rozsiahle a implementuje mnoho funkcionalít. V tejto časti spomenieme tie najpodstatnejšie, ktoré následne budeme pravdepodobne potrebovať pri implementácii webovej AR aplikácie. Viac informácii a kompletnú dokumentáciu DOM API čitateľ nájde na stránkach W3Schools. [20]

Vďaka zápisu scény v grafe, pomocou DOM v HTML, môžeme ľahko využívať DOM API, napríklad funkciou `querySelector()`. Dostaneme tak presne jeden element, na ktorý sme odkazovali v požiadavke. Odkazovať môžeme na selektory kaskádových štýlov (CSS), ktoré sa môžu zhodovať s ID, triedou, typom, atribútom alebo hodnotou atribútu. DOM API ponúka viac metód pre požadovanie elementov.

Špecifickou metódou na požiadanie podľa ID je `getElementById()`. Ide o efektívnejšiu variantu, pretože oproti `querySelector()` metóde nemusí analyzovať celý CSS strom. Ďalšími špecifickými metódami sú `getElementsByName()`, `getElementsByTagName()` a `getElementsByClassName()`. Všetky tieto metódy môže nahradiť metóda `querySelectorAll()`, ktorá, rovnako ako zvyšné metódy, vráti `NodeList` obsahujúci všetky elementy spĺňajúce požiadavky.

DOM API umožňuje vytvárať nové elementy volaním `createElement()`. Po nastavení parametrov môžeme vytvorený element priradiť inému elementu pomocou metódy `appendChild()`. Samozrejme elementy môžeme analogicky aj odoberať metódou `removeChild()`.

K atribútom elementov môžeme pristupovať metódou `getAttribute()`. Pre nastavenie atribútov elementov je k dispozícií metóda `setAttribute()`. Vo vlastných komponentoch, o ktorých sme písali v časti 2.1.2, môžeme definovať `parse` funkciu. V nej môžeme popísať ako sa má reťazec rozdeliť na hodnoty. Nakoniec ak je potreba, komponent môžeme volaním `removeAttribute()` odstrániť.

V nasledujúcej časti sa budeme venovať poslednej skupine metód. Jedná sa o skupinu udalostí a poslucháčov udalostí.

2.1.6 Interakcia a udalosti

Základný princíp akcie a reakcie sa uplatňuje pri udalostiach a ich poslucháčoch. O to sa starajú 3 základné funkcie:

`emit()` Využíva sa na vyvolanie oznámenia o udalosti.

`addEventListener()` Využíva sa na vyvolanie reakcie na udalosť.

`removeEventListener()` Využíva sa na odobranie reagovania na udalosť.

Metóda `emit()` prijíma 3 parametre. Iba prvý je povinný a tým je `name`. Tým určíme akú udalosť spúšťame. Ďalším parametrom je `detail`. V detaile môže byť informácia o „terči“, elemente ktorého sa udalosť týka. Posledný

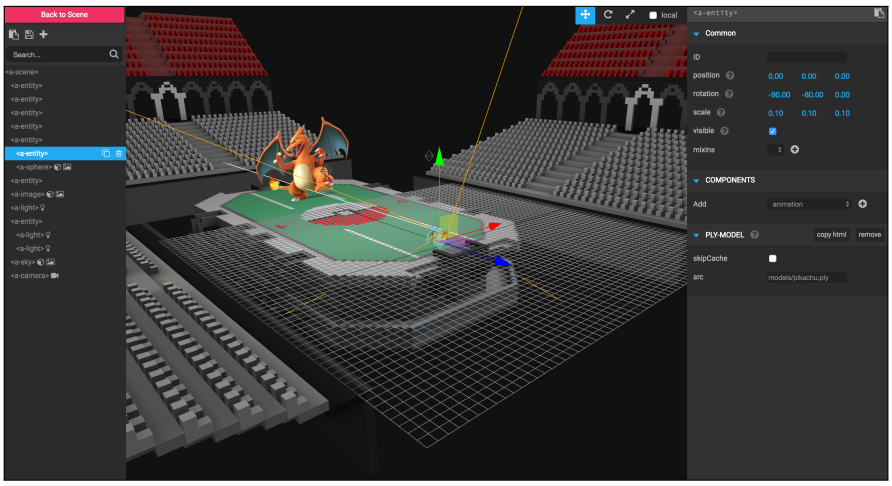
parameter je `bubbles`. Ten určuje, či sa informácia o konaní udalosti posiela predkom v DOM strome.

Ďalšia metóda, ktorou môžeme pridávať reakcie na udalosti má 2 parametre. Naprv v `addEventListener()` určujeme názov udalosti `eventName`, na ktorú je potreba reagovať. Nasledujúcim parametrom je `function`, teda funkcia, ktorou daný element reaguje.

Poslednou metódou `removeEventListener()` môžeme danú reakciu odstrániť. Parametre sú rovnaké s funkciou `addEventListener()`.

2.1.7 Inšpektor a nástroje pre vývojárov

A-Frame poskytuje nástroj na kontrolu stavu aplikácie. Kedykoľvek potrebujeme zistiť informácie o entite a jej komponentoch, môžeme spustiť inšpektora. Po stlačení kombinácie `<ctrl>` + `<alt>` + `i` sa pozastaví aplikácia a zobrazí sa okno inšpektora. Obrázok 2.1 je ukážkou ako inšpektor vyzerá. Na ľavej strane inšpektora máme celý strom od elementu `<a-scene>`. Uprostred je náhľad scény v súčasnom stave s prvkami na manipuláciu objektov. Na pravej strane vidíme komponenty a hodnoty aktuálne zvolenej entity.



Obr. 2.1: A-Frame inšpektor scény

Výhoda inšpektora spočíva v tom, že môžeme okamžite upravovať hodnoty komponentov a vidieť ich efekt. To dokáže veľmi urýchliť vývoj a úpravy scény. Ide o podobný nástroj ako prezeranie DOM stromu priamo v prehliadači, no s tým rozdielom, že je špecificky prispôsobený pre A-Frame, resp. vývoj scény v 3D prostredí.

Pri vývoji VR aplikácie je veľmi užitočným nástrojom zachytenie pohybu. V tejto práci síce tento nástroj zrejme nebudeme potrebovať, no spomenieme jeho hlavné výhody. Pomáha hneď z niekoľkých hľadísk. Azda najpodstatnejšou výhodou je schopnosť spustiť nahrávku a testovať aplikáciu bez VR zariadenia.

Dovoľuje to viacerým vývojárom sústrediť sa na vývoj bez nárokov na počet VR zariadení, či problémov s opakovaným nasadzovaním zariadenia na hlavu po každej zmene. Zdieľanie nahrávok ako forma nahlasovania chýb v aplikácii je taktiež možnosťou. Ďalšou výhodou je využitie automatizovaného testovania. S pomocou nahrávok, frameworku Karma a Mocha môžeme podrobiť aplikáciu unit testovaniu. [21]

Systém `avatar-replayer` umožňuje spustiť mód pozorovateľa. Ak chceme niekomu čosi pomocou aplikácie demonštrovať, ide o vhodnú formu. Odstraňuje roztrasený pohyb kamery, čo značne vylepší zážitok a pozorovanie používateľa. Navyše dovoľuje používateľovi voliť akýkoľvek pozorovací bod. Môže sa teda sústrediť na tú najpodstatnejšiu časť nahrávky.

2.1.8 Komunitné rozšírenia

Okolo frameworku A-Frame sa postupne rozrastá komunita aktívnych vývojárov a nadšencov. V súčasnosti má komunitný Slack cez 7000 používateľov. Komunita ponúka rôzne rozšírenia v podobe komponentov. Kedysi najväčším zdrojom komunitných komponentov bol A-Frame register, dnes sú všetky komponenty dostupné na npm registry. [22, 23] A-Frame sa stále vyvíja, čo občas spôsobuje problémy s kompatibilitou jednotlivých komponentov s niektorými jeho verziami.

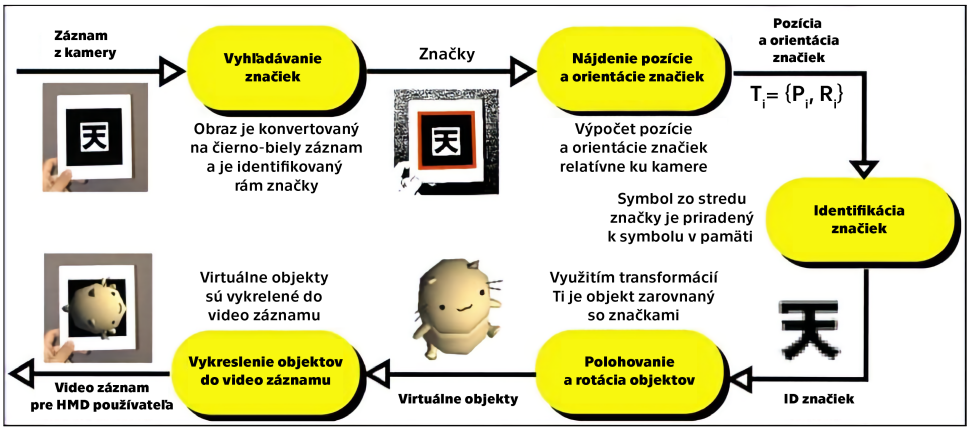
2.2 Porovnanie AR knižníc

Ako sme v predchádzajúcej sekcii spomínali, A-Frame komunita prináša mnoho rozšírení. Jedným z možných rozšírení je aj pridanie AR systému do scény. Objavili tri, z čoho boli principiálne odlišné iba dve. AR.js a aframe-ar boli podľa githubu dve najpopulárnejšie. [24, 25] Prvá spomínaná knižnica využíva princíp založený na lokalizácii pomocou značiek. Druhá v poradí naopak využíva detekciu prostredia.

AR.js vytvoril Jerome Etienne a ide o spracovanie knižnice ARToolKit do webového prostredia. Rozšírenie aframe-ar vytvorila organizácia „chenzlabs“, ktorá ale na Github-e nemá žiadnych verejných členov. Je postavená na knižnici Three.ar.js, ktorá vie využiť ARCore aj ARKit.

2.2.1 ARToolKit

Už v úvode sme spomínali knižnicu ARToolKit a jej vznik. Knižnica problém 6 stupňov slobody (6DOF) rieši využívaním značiek v scéne. Počítačovým videním vykoná rozbor obrazu a vypočíta tak polohu a rotáciu kamery relatívne ku scéne, resp. naopak. V diagrame na obrázku 2.2 môžeme vidieť jednotlivé kroky v priebehu spracovania obrazu a jeho následného rozšírenia.



Obr. 2.2: Diagram spracovania obrazu knižnicou ARToolKit

V spracovaní sú dve kľúčové zložky. Prvou zložkou je určenie pozície a pózy značky. Druhou je kalibrácia kamery a HMD. Keďže v tejto práci uvažujeme využitím v mobilnej aplikácii, môžeme druhú zložku zanedbať.

Kato a Billinghurst [26] popisujú prvú zložku nasledovne. Cieľom analýzy obrazu je získať transformačnú maticu T_{cm}, ktorú môžeme vidieť v rovnici (2.1).

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} & V_{13} & W_x \\ V_{21} & V_{22} & V_{23} & W_y \\ V_{31} & V_{32} & V_{33} & W_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = T_{cm} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \tag{2.1}$$

Po úprave obrazu do čierno-bielej verzie, nasleduje hľadanie hrán rámu značky. Nájdením všetkých 4 hrán dostávame vrcholy hľadaného štvorca. Tieto body prieniku sú uložené pre neskoršie spracovanie. Pre identifikovanie vzoru musí byť obraz najprv normalizovaný. Obraz je normalizovaný perspektívnou transformáciou z rovnice (2.2). Všetky premenné v transformačnej matici sú determinované substitúciou súradníc obrazu a značky zistenými 4 vrcholmi za (x_c, y_c) a X_m, Y_m

$$\begin{bmatrix} hx_c \\ hy_c \\ h \end{bmatrix} = \begin{bmatrix} N_{11} & N_{12} & N_{13} \\ N_{21} & N_{22} & N_{23} \\ N_{31} & N_{32} & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ 1 \end{bmatrix} \tag{2.2}$$

Premietnutím 2 paralelných strán značky na obrazovku, vzniknú priamky, ktorých rovnice na obrazovke budú:

$$a_1 x + b_1 y + c_1 = 0, \; a_2 x + b_2 y + c_2 = 0 \tag{2.3}$$

Hodnota týchto parametrov je získaná už v procese hľadania hrán. Perspektívna projekčná matica z rovnice (2.4) je získaná z údajov nastavenia použitej

kamery. Rovnice rovín, ktoré obsahujú tieto dve strany môžu byť reprezentované ako rovnice (2.5) na obrazovke substitúciou x_c a y_c v rovnici (2.4) za x a y z rovnice(2.3).

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & 0 \\ 0 & P_{22} & P_{23} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} hx_c \\ hy_c \\ h \\ 1 \end{bmatrix} = P \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{2.4}$$

$$\begin{aligned} a_1P_{11}X_c + (a_1P_{12} + b_1P_{22})Y_c + (a_1P_{13} + b_1P_{23} + c_1)Z_c = 0 \\ a_2P_{11}X_c + (a_2P_{12} + b_2P_{22})Y_c + (a_2P_{13} + b_2P_{23} + c_2)Z_c = 0 \end{aligned} \tag{2.5}$$

Nech normálové vektory k získaným rovinám sú n_1 a n_2, potom smerový vektor 2 paralelných strán štvorca je daný vonkajším produktom n_1 a n_2. Nech tieto dva jednotkové smerové vektory, získané z dvojíc paralelných strán štvorca, sú u_1 a u_2 a mali by na seba byť kolmé. Dôsledkom chýb vzniknutých spracovaním obrazu vektory nemusia byť kolmé. Pre kompenzáciu vzniknutých chýb, sú zvolené vektory v_1 a v_2, ktoré sa nachádzajú v rovnakej rovine ako u_1 a u_2, no sú na seba kolmé. Nech je jednotkový smerový vektor kolmý na oba vektory v_1 a v_2, potom je to smerový vektor v_3. Rotačná časť V_{3x3} z transformačnej matice T_{cm} z rovnice (2.1) je $[V_1^t, V_2^t, V_3^t]$.

So známou rotačnou časťou V_{3x3} z transformačnej matice je translačná časť W_x W_y W_z dopočítaná použitím rovníc (2.1), (2.4) a získaných vrcholov značky v súradniciach značky a obrazovky. Postup vygeneruje 8 rovníc, ktoré obsahujú translačnú časť.

Identifikácia značky vyžaduje dobré svetelné podmienky a pomerne ostrý obraz.

2.2.2 ARKit/ARCore

Knižnice ARKit a ARCore pracujú na rovnakom princípe. Využívajú mapovanie priestoru v spojení s metrikami z akcelerometra a gyroskopu.

Najprv musí systém nasnímať niekoľko obrázkov, v ktorých deteguje takzvané význačné body. Na detekciu význačných bodov sú využívané jednoduché detekcie rovín. Využitie komplexných algoritmov, ktoré zvyčajne vytvárajú husté mračno bodov, by zbytočne míňalo čas procesoru a energiu batérie. [11] Potom porovnáva jednotlivé body a zmeny medzi nimi. Stačia malé zhluky bodov aby boli vytvorené roviny, na ktoré potom môže používateľ položiť objekty v AR. Pretože obe knižnice využívajú detekciu význačných bodov, je vhodné aby snímaný priestor nebol jednofarebný bez textúr.

Sledovaním scény môže byť určená poloha zariadenia, no zvyčajne je na to potrebné mať dve kamery. Z dvoch kamier so známou vzájomnou vzdialenosťou je možné stereoskopicky dopočítať polohu. Mobilné telefóny zvyčajne ale nemajú stereoskopické kamery. Práve na to je slúži jednotka merania zotrvačnosti

(IMU). Frekvencia optických signálov je 30–60 Hz, zatiaľ čo frekvencia IMU signálov až 1000 Hz. Medzi jednotlivými obrazmi z kamery je vypočítavaná relatívna zmena polohy zariadenia. [11]

Vidíme tu výhodu prepojenia jednotlivých technológií. Ak zlyháva optické sledovanie, meranie zotrvačnosti ho na istú dobu môže zastúpiť. Samotná IMU však rýchlo stratí presnosť, preto ju vhodne dopĺňa optické mapovanie. Súčasne najväčším problémom v tejto symbióze je veľká spotreba energie, ktorej v mobilnom zariadení nie je veľa.

Jeden zásadnejší rozdiel medzi ARCore a ARKit je v mapovaní prostredia. Presnejšie v uchovávaní informácií o prostredí. ARCore si uchováva oveľa väčšiu mapu prostredia. Na druhú stranu ARKit je vyvíjaný pre špecifickejšiu množinu zariadení, vďaka čomu môže mať lepšiu korekciu chýb z IMU. [10]

Obe knižnice ponúkajú základný odhad osvetlenia v scéne, vďaka čomu môžu výrazne vylepšiť užívateľský zážitok z aplikácie. Ide však iba o intenzitu a teplotu osvetlenia, bez závislosti na ostrosti tieňov.

2.2.3 Rozdiely prístupov

Každý prístup má svoje výhody a nevýhody. Podstatnou výhodou mapovania priestoru je flexibilita a stabilnosť systému. Svetelné podmienky majú značne menší vplyv na zobrazenie scény. Vďaka mapovaniu a využívaniu kotiev v scéne dokážu zobrazovať objekty stojace na rovine aj keď zem vôbec nesnímame. Na druhú stranu potrebujú výkonnejší hardvér a spotrebujú viac batérie.

AR využívajúca sledovanie značiek je výhodná svojimi nižšími nárokmi na výkon zariadenia. Preto ide o vhodné riešenie ak chceme podporovať väčšiu skupinu zariadení. Nevýhodou pri používaní tejto techniky je väčšie obmedzenie na vplyv okolitého osvetlenia. Ďalšou nevýhodou je obmedzenie pohybu. Používateľ sa nemôže vzdialiť príliš ďaleko od značky a zároveň značka nesmie opustiť priestor zaberaný kamerou. Príliš veľa pohybu môže spôsobiť rozmazanie obrazu, čo tiež znemožní rozšírenie scény.

2.3 Základné pravidlá pri návrhu UI pre AR

Táto sekcia sa venuje zaužívaným pravidlám pri vytváraní UI s ohľadom na čo najlepší zážitok (UX) pre používateľa. Väčšina zdrojov vychádza z dokumentácie ku knižnici ARKit a konferencie Google I/O 18'. [27, 28]

V Google hovoria o piatich základných pilieroch pri návrhu AR aplikácie. Tými piatimi piliermi sú:

- Pochopenie užívateľského prostredia
- Naplánovanie pohybu používateľa
- Plynulá inicializácia používateľa do AR

- Návrh interakcie s prirodzenými objektmi
- Rovnováha rozhrania medzi prvkami na obrazovke a v scéne

AR aplikácia má byť pútavá a pohlcujúca, preto by mala využívať celú plochu obrazovky. Čím je aplikácia viac zapojená do prostredia používateľa, tým zaujímavejšou sa stáva. Väčšina aplikácií zobrazuje scénu iba na jednej ploche. Pútavejšie je využiť viac plôch, prípadne ich nejak prepojiť. Príkladom môže byť rozbitie objektu, ktoré pokračuje padnutím kúskov na zem.

Zmena osvetlenia môže fungovať ako spúšťač. Po zhasnutí sa môžu rozsvietiť svetlá objektov v scéne. Pri návrhu aplikácie by sme mali úplne zabudnúť na klasický návrh 2D aplikácie. Nie je dobré sa zväzovať zaužívanými technikami pre ploché aplikácie, pretože to pravdepodobne bude viesť k vytvoreniu nudnej aplikácie, ktorá nevyužije možnosti otvoreného 3D sveta.

Ľudia sú zvyknutý na uzatvorené 2D aplikácie a preto si neuvedomia, že v AR aplikácii majú slobodnejšie možnosti a mali by sa hýbať. Preto je doporučené vkladať do scény pohyblivé objekty aby prinútili používateľa hýbať kamerou a rozhliadať sa v novom svete. Treba však myslieť na to aby mal používateľ dostatok miesta na pohyb. Tu sa vyčleňujú tri veľkostné modely:

- stolová
- izbová
- otvorená

Zle vybraný veľkostný model môže znamenať, že v polovici používania aplikácie užívateľ zistí, že nemá dosť miesta a musí aplikáciu prerušiť. Preto je ideálne už pri pokladaní scény zdôrazniť jej nároky na priestor.

Používateľ by sa mal cítiť pohodlne po celú dobu používania aplikácie, preto treba myslieť na to aby nebol nútený držať zariadenie dlhodobo v neprirodzenej polohe. Ak je potrebné aby sa používateľ počas používania aplikácie intenzívnejšie hýbal, mala by ho aplikácia viesť k pohybu postupne. Vynútenie pohybu hneď po vstupe do AR aplikácie vedie k nepríjemnému zážitku.

V záujme bezpečnosti by aplikácia nemala vyžadovať od používateľa príliš rýchle a rozsiahle pohyby. V okolí môžu byť ľudia, steny, či nábytok.

Ak sa používateľ ocitne kamerou na mieste, kde by nemal byť, aplikácia by mu to jasne mala ukázať. Väčšinou nie je vhodné aby kamera vošla do nejakého modelu. V tom prípade by sa zobrazená scéna mala zmeniť nejakým efektom aby bolo zrejmé, že je v neprípustnom stave.

Pri presúvaní objektu je vhodné zobrazovať detekciu zeme, aby používateľ vnímal výšku, v ktorej sa objekt nachádza. Pre pútavejšiu skúsenosť by aplikácia mala reagovať na interakciu s prostredím vydaním zvuku alebo haptickou odozvou. Vhodným doplnením skúsenosti je aj atmosferická hudba.

V AR scéne majú používatelia často tendenciu prehliadať 2D elementy priamo na obrazovke a sústredia sa tak viac na pohyblivú 3D scénu. Preto je

lepšie vkladať UI elementy priamo do scény. Jediné ploché elementy, ktoré je vhodné dávať priamo na obrazovku sú:

- veľmi často používané ovládacie prvky
- ovládacie prvky vyžadujúce rýchly prístup

V AR scéne by sme sa mali vyvarovať používaniu textu v UI prvkoch. Náležitejšími sú ikony, ktoré rozhodne popisujú daný prvok. V 3D scéne sa môže používateľ od objektu veľmi vzdialiť alebo naopak byť veľmi blízko. UI elementy by však mali byť vždy dostatočne veľké aby ich mohol používateľ stlačiť.

Využívanie gest je prirodzenejšie, no vyberanie kurzorom dáva viac priestoru pre sledovanie scény. Pridaním lúča k vybranému efektu môžeme pridať používateľovi pocit váhy objektu. Ťažší objekt bude pri pohybe viac ohýbať lúč, kdežto menší naopak.

Ďalším spôsobom interakcie v scéne môže byť aj samotná poloha. Podľa proximity k nejakému objektu môžeme zareagovať pútavou udalosťou. Postavy v scéne môžu reagovať na polohu zariadenia otáčaním hlavy pri priblížení.

Prechod z AR do 2D aplikácie by mal byť vynútený jedine používateľom. Keď je používateľ pohltený v aplikácií, tak to pôsobí veľmi rušivým dojmom.

Modely môžu obsahovať lesklé povrchy, no mapa okolia je vytvorená iba z doposiaľ získaných obrazových dát. Preto by lesklé plochy mali byť malé aby boli uveriteľné.

2.4 Požiadavky aplikácie

Stanovenie požiadaviek na tvorenú aplikáciu je veľmi dôležité. Dáva to vývojárom možnosť ľahko overiť, či vytvárajú správne produkt pre používateľa z hľadiska funkčnosti. Zároveň nefunkčné požiadavky stanovujú minimálne nároky na zariadenia, na ktorých má aplikácia fungovať.

2.4.1 Funkčné požiadavky

F-01 Používateľ môže vymeniť vizualizáciu budovy za nasledujúcu

Nasledujúci model budovy sa po interakcii načíta a vymení so súčasným modelom. V prípade, že je súčasný model najnovší, môže byť cyklicky načítaný najstarší model.

F-02 Používateľ môže vymeniť vizualizáciu budovy za predchádzajúcu

Predchádzajúci model budovy sa po interakcii načíta a vymení so súčasným modelom. V prípade, že je súčasný model najstarší, môže byť cyklicky načítaný najnovší model.

F-03 Používateľ môže zobrazený model na scéne posúvať

Posun modelu má byť umožnený aj inak ako posunom kamery.

F-04 Používateľ môže zobrazený model na scéne otáčať

V prípade, kedy používateľ nemá dosť miesta na pohyb kamerou okolo modelu, môže využiť možnosť otáčania modelu v aplikácii.

F-05 Používateľ môže zobrazený model na scéne škálovať

Ak chce používateľ lepšie vidieť nejaký detail alebo je model budovy príliš veľký, môže využiť možnosť škálovania modelu v aplikácii.

F-06 Používateľ môže zobraziť/skryť grafické rozhranie pre úpravy

Používateľ musí mať možnosť ľahko skryť alebo zobraziť rozhranie pre úpravy daného modelu.

2.4.2 Nefunkčné požiadavky

N-01 Aplikácia je navrhnutá nezávisle od platformy zariadenia

Zo zadania plynie požiadavka na multiplatformné zameranie.

N-02 Aplikácia je spustiteľná na zariadeniach s OS Android

Minimálne požiadavky na verziu OS sú >=6.0.1

N-03 Aplikácia je spustiteľná na zariadeniach s OS iOS

Minimálne požiadavky na verziu OS sú >=11.0

N-04 Aplikácia má jednoduché a intuitívne užívateľské rozhranie

Užívateľské rozhranie aplikácie by malo byť jednoduché a minimálne.

N-05 Aplikácia dokáže načítať a zobraziť rôzne formáty 3D modelov

Aplikácia má podporovať najbežnejšie formáty pre 3D grafický obsah. Minimálne kritéria na podporované formáty sú: .dae, .gltf, .obj + .mtl, .stl.

Kapitola 3

Návrh

V tejto časti práce sa budeme venovať návrhu aplikácie a vhodným postupom pri vytváraní rozhrania pre AR aplikácie. Začneme popisom návrhu spracovania modelov, kde sa budeme venovať hlavne samotnému načítavaniu a následnej výmene modelov. Ďalšia sekcia rozoberá prácu s modelom a spôsob interakcie. Na záver navrhneme ako by malo vyzerať UI vo vytvorenom prototype.

V analytickej časti sme najprv stanovili čo ponúka framework A-Frame v sekcii 2.1. Neskôr sme stanovili v sekcii 2.4 funkčné a nefunkčné požiadavky. Z analýzy vyplýva, že zvolením frameworku A-Frame a knižnice AR.js by sme mali splniť všetky nefunkčné požiadavky zo sekcie 2.4.2, s výnimkou nefunkčnej požiadavky **N-04**. Požiadavke **N-04** sa budeme venovať v sekcii 3.3 a čiastočne 3.2. Jediným problémom by eventuálne mohlo byť využitie funkcií, ktoré nie sú podporované väčšinou mobilných prehliadačov. Preto navrhujeme počas implementácie striktne kontrolovať podporu HTML5, CSS3 a JavaScript volaní. [29]

3.1 Návrh spracovania modelov

Ako načítavať modely a neskôr ich medzi sebou vymieňať rozoberieme v ďalších častiach. Kam však ale spomínané modely zobraziť. Modely budeme ukladať na značky. Najpoužívanejšou značkou je tzv. „Hiro marker“. Ide o dobrú testovaciu značku. V hotovej aplikácii budeme zrejme potrebovať značiek viac. Preto navrhujeme využívať maticové vzory. Jedná sa o značky podobné známym QR kódom. Nie je však potrebné aby boli tak komplexné. Budú stačiť matice o veľkosti $3x3$, ktoré nie sú symetrické.

3.1.1 Načítavanie modelov

Z analýzy vyplýva, že A-Frame poskytuje silný nástroj na spravovanie zdrojov. Navyše vždy čaká definovaný čas, pokiaľ nenačíta všetky zdroje, až potom začne vykresľovať scénu. Predvolená hodnota čakania na načítanie sa zdá byť

dostatočná. Vďaka knižnici three.js, ktorá je základom frameworku A-Frame, môžeme naplno zúžitkovať `FileLoader` a načítať tak veľké množstvo rôznych formátov. Avšak najvýhodnejším formátom je formát .gltf, ktorý v sebe dokáže niesť ako informácie o materiáli, tak i o animácii objektu. Čo sa materiálu týka, môže byť definovaný niekoľkými textúrami pre fyzikálne založené vykresľovanie (PBR). Volíme ho ako ideálny formát pre našu aplikáciu.

3.1.2 Zmena modelov

Pre zmenu modelu potrebujeme poznať všetky modely, ktoré spolu súvisia. V `<a-assets>` nájdeme úplne všetky modely a zdroje, ktoré v aplikácii budeme využívať. Nejakým spôsobom však musíme modely anotovať, aby sme odlíšili modely týkajúce sa jednej budovy od modelov druhej. Lexikálne by sme mohli deliť modely podľa unikátneho názvu budovy. Unikátnym identifikátorom pre budovu z konkrétneho obdobia by mohlo byť spojenie názvu budovy s rokom, ku ktorému sa viaže.

Pri využití takejto anotácie jednoducho vytriedime iba podstatné modely. Ďalším krokom je vytvorenie komponentu, ktorý sa bude starať o indexovanie v množine modelov. Tento komponent bude zabezpečovať zmenu modelu za nasledovný, resp. predchádzajúci, ak si to používateľ vyžiada. Týmto budú splnené funkčné požiadavky **F-01** a **F-02**.

3.2 Návrh interakcie s modelom

Sekcia 2.3 stanovila základné pravidlá tvorenia UI a UX. Obsahuje, samozrejme, i mnoho informácií o tom ako navrhovať interakcie pre AR aplikácie. Na základe zvolenej technológie sa dostávame do väčšej limitácie, kvôli potrebe udržania značky v scéne.

Hlavným cieľom aplikácie je pozorovať vizualizácie budov a ich rôzne podoby v histórii. Preto neočakávame náhle pohyby používateľa. Chceme aby si vychutnal pohľad na dobové modely naplno a preto navrhujeme využitie celého displeja a minimalizáciu vstupov z displeja. Vhodným riešením je použitie kurzora, resp. terča. Terč ukazuje na objekty, no potvrdenie stlačenia stále musí používateľ nejak vynútiť. Prichádzajú do úvahy 2 možnosti kliknutia:

- dotykom na displeji
- sledovaním cieľa určitú dobu

Obe možnosti majú svoje výhody a nevýhody. Kliknutie dotykom, znamená, že si používateľ musí zakryť časť obrazu. Keďže používame terč, stačí sa dotknúť na kraji a používateľ to zvládne i jednou rukou. Kliknutie sledovaním, nemusí byť intuitívne, preto je vhodné užívateľovi nejak naznačiť, že prebieha výber. Nevýhodou môže byť neustále klikanie pri sledovaní nejakého objektu.

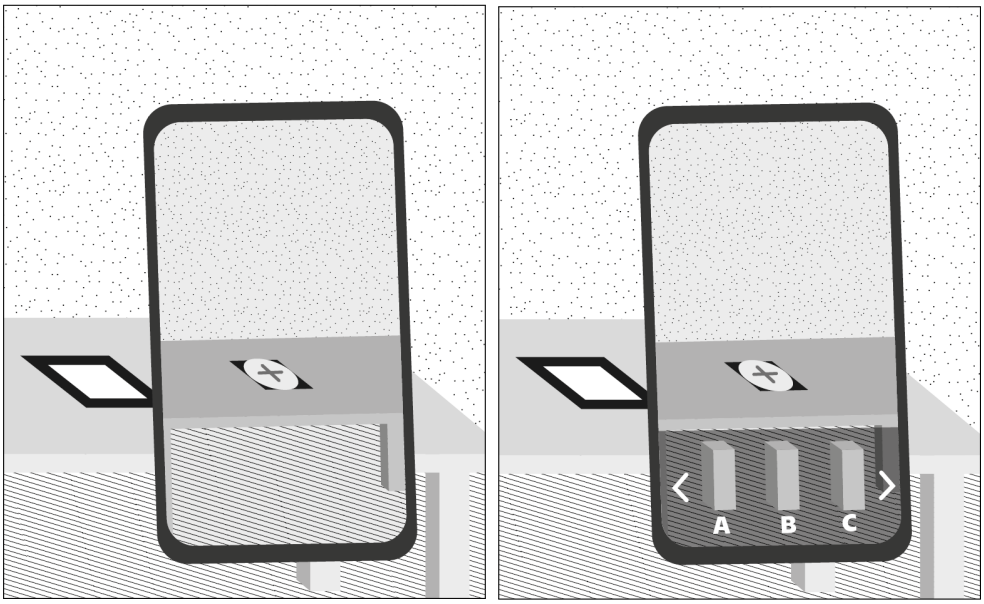
Funkčná požiadavka **F-06** vyžaduje zobrazenie/skrytie rozhrania pre úpravy. Navrhujeme, s využitím terča, skrývať a zobrazovať rozhranie pre úpravy kliknutím na model budovy.

Požiadavky **F-03**, **F-04** a **F-05** sa týkajú úprav modelu. Prvou menovanou je posun objektu v scéne nezávisle od kamery. Pretože sme zvolili AR sledovaním značiek, navrhujeme nepresúvať model v aplikácii ale fyzicky presunom značky. Chceme podnietiť používateľa k pohybu okolo modelu, preto navrhujeme rotovať objekt v nástroji pre úpravy iba krokovo, aby to využil iba v prípade nutnosti. Napr. kvôli nedostatku miesta. Zmenu veľkosti navrhujeme meniť rovnako krokovo, napr. úpravou o 10%.

3.3 Návrh rozhrania

V predchádzajúcej sekcii sme navrhli niekoľko riešení z pohľadu UX. V tejto sekcii sa budeme venovať nastoleným požiadavkám z pohľadu UI. Cieľom je navrhnúť UI, tak aby spĺňalo požiadavku **N-04**.

Prvá vec, ktorú používateľ uvidí po zapnutí aplikácie by mala byť možnosť pridať na scénu budovu. Navrhujeme to používateľovi jasne prezentovať zobrazením tlačidla s veľkou ikonou +, ako to je možné vidieť na obrázku 3.1a. Po kliknutí na ikonu, by sa mala otvoriť galéria s modelmi. Na obrázku 3.1b



(a) Úvodná scéna po zapnutí aplikácie (b) Galéria s modelmi

Obr. 3.1: Úvodný stav aplikácie

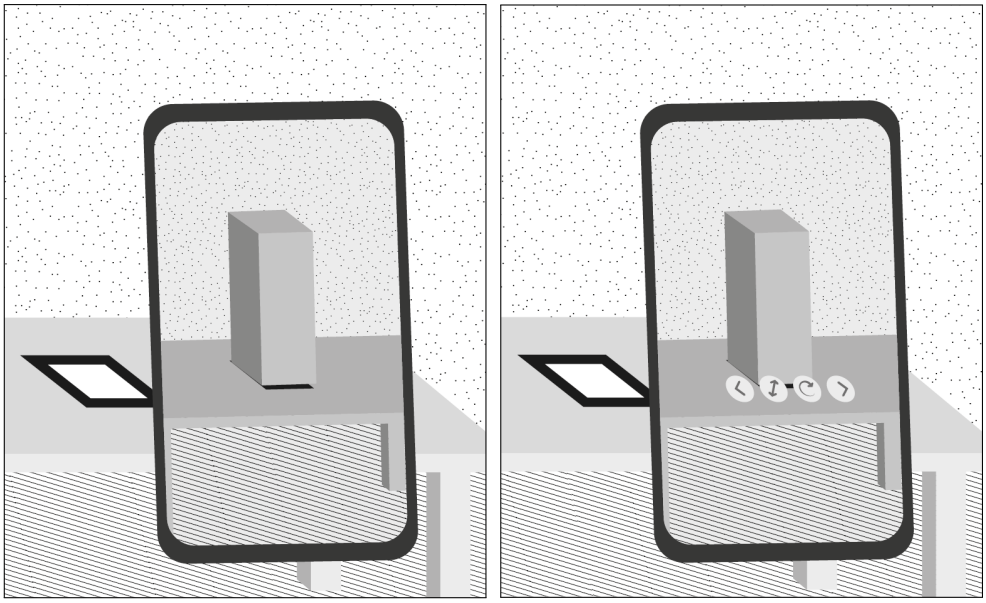
ilustrujeme, že používateľ by mal mať možnosť listovať v galérii. Po vybraní budovy, by sa mal objekt vložiť do scény.

Používateľ si môže objekt ľubovoľne prezerať zo všetkých strán. Po kliknutí na model budovy, by sa mu mal zobraziť základný panel úprav. Panel úprav by mal pozostávať z ikon zreteľne komunikujúcich ich funkciu:

Menu základných úprav

- Zvolenie predchádzajúceho modelu
- Zmena veľkosti modelu
- Zmena rotácie modelu
- Zvolenie nasledujúceho modelu

Návrh základného panelu je zobrazený na obrázku 3.2b.



(a) Scéna s vloženým modelom (b) Model s otvoreným panelom úprav

Obr. 3.2: Prezentačný mód a mód s panelom úprav

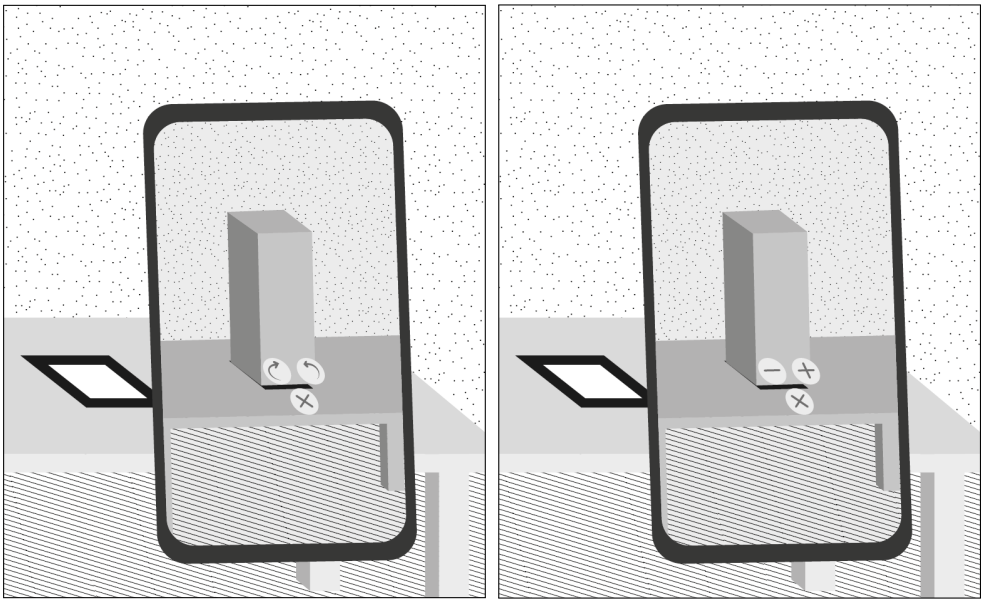
Po zvolení jednej z modifikácií zo základného panelu úprav sa otvorí špecifické rozhranie pre danú modifikáciu. V tomto menu bude rozhranie reagovať na základe funkcionalít popísaných v predchádzajúcej sekcii. Návrhy špecifického menu môže čitateľ vidieť na obrázku 3.3. Opäť je podstatné aby ikony udávali jasne akú funkciu vykonajú.

Menu zmeny veľkosti

- Zmenšenie veľkosti
- Zväčšenie veľkosti

Menu zmeny rotácie

- Rotácia okolo osy Z v smere hodinových ručičiek
- Rotácia okolo osy Z v protismere hodinových ručičiek



(a) Menu pre rotáciu modelu (b) Menu pre škálovanie modelu

Obr. 3.3: Model s otvoreným menu modifikácie

Kapitola 4

Realizácia

Nasledujúca kapitola popisuje ako sme vypracovali prototyp. Na začiatku sa venujeme komunitným komponentom. Predovšetkým tomu aké sú dostupné a vhodné pre náš prototyp. Potom rozoberieme aké vlastné komponenty bolo potrebné implementovať. Posledná sekcia je venovaná implementácii interakcii a tvorbe užívateľského rozhrania.

Prototyp je vytvorený, samozrejme, frameworkom A-Frame. Vychádzajúc z návrhu, knižnicou poskytujúcou AR rozšírenie je AR.js. Ako je pre A-Frame aplikácie typické, základom je graf scény v HTML. Graf scény obsahuje zoznam zdrojov zapísaných pod elementom `<a-assets>`, aby sa A-Frame manažér zdrojov postaral o ich načítanie pred vykresľovaním. Ďalej graf obsahuje základ scény v podobe značiek a kamery. Ďalšie rozšírenia aplikácie, ktorými sú vlastné komponenty a aplikačná logika, sú implementované v jazyku JavaScript.

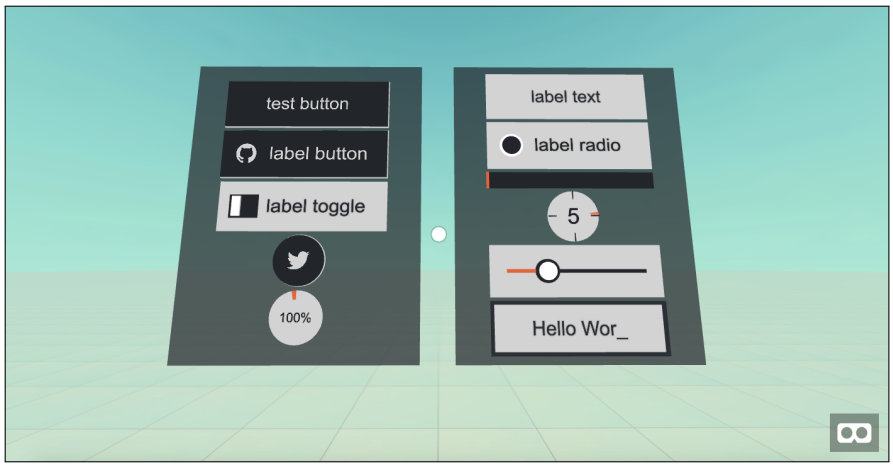
4.1 Komunitné komponenty

Komunita okolo frameworku A-Frame je pomerne aktívna a poskytuje množstvo užitočných verejných komponentov. Ako sme vylíčili už v sekcii 2.1.8, A-Frame je mladý a moderný framework, s rozrastajúcou sa komunitou. To však so sebou občas prináša problémy pri vývoji. V čase písania tejto práce je najaktuálnejšou verziou A-Frame 0.9.0. Počas implementácie sme vyskúšali niekoľko rôznych komunitných komponentov. Veľkým problémom komunitných komponentov je zlá podpora. Niektoré komponenty fungujú správne iba s niektorými konkrétnymi verziami.

Inak to nebolo ani s balíkom komponentov, ktoré sme využili pre tvorbu GUI elementov a prácu s kurzorom. Použili sme ho v troch verziách a v každej fungoval inak. Vo verzii 0.7.0 nefungoval vôbec. Pre verziu 0.8.2 sme museli dopísať vlastný komponent pre spracovanie kliknutia kurzorom a nakoniec vo verzii 0.9.0 nefungovali niektoré komponenty z balíka. Preto sme zvolili verziu 0.8.2 a v nej sme pokračovali s vývojom aplikácie.

4.1.1 aframe-gui

`aframe-gui` je balík GUI primitív a komponentov, na ktorom stojí interakcia vytvoreného prototypu. Ponúka rôzne tlačidlá, posuvníky, prepínače, kurzory, či ukazovatele postupu. Z knižnice sme využili dva typy kurzorov `<a-gui-cursor>`. Jednotlivé typy boli spomínané už v časti 3.2. Keďže sú vzájomne ľahko vymeniteľné implementovali sme oboje. Následne budeme testovať jednotlivé verzie v užívateľskom testovaní, aby sme zistili, ktorá verzia je vhodnejšia. Ďalším primitívom, ktoré vytvára prakticky celé rozhranie je tlačidlo `<a-gui-icon-button>`. Posledným využitým komponentom z tejto knižnice je `gui-interactable`. Týmto komponentom môžeme označiť všetky entity, na ktoré môže používateľ klikať. Parametrom môžeme nastaviť názov funkcie, ktorá sa má po kliknutí zavolať. [30]



Obr. 4.1: Dostupné primitíva z balíka `aframe-gui`

4.1.2 AR.js

Rozšírenie AR.js využívame pre vytvorenie AR aplikácie, registráciu značiek a výpočet polohy objektu. AR.js vrámci ECS architektúry dopĺňa do scény systém `arjs`, ktorý zabezpečuje získanie obrazu cez JavaScript API volanie `getUserMedia`. Toto volanie je najväčším obmedzením v prehliadačoch, pretože mnoho starších prehliadačov ho nepodporuje. V prípade iOS je podporované iba v prehliadači Safari, pretože operačný systém nedáva dostatočné oprávnenia prehliadačom tretích strán. Systém ďalej spracováva obraz, podľa nastavených hodnôt. Môžeme nastaviť mód spracovania, ktorý nastavuje rozlíšenie. To je vhodné pre zníženie hardvérových nárokov. Ďalej môžeme určiť, či sa má detegovať farebný obraz alebo stačí čiernobiely. V prípade používania klasických značiek, je vhodné nastaviť registráciu na čiernobielo. Zlepší sa tým

kvalita registrácie a tým stabilita aplikácie. Okrem priameho záznamu z môže systém AR.js spracovať aj nahrávku, či iba statický obraz.

Ďalej rozšírenie ponúka dve primitíva. Nesmú sa však používať vzájomne, pretože nimi určíme ako bude systém pracovať s nájdenou značkou:

<a-marker> Kamera je umiestnená do stredu súradnicového systému a počíta sa vzdialenosť značiek od kamery

<a-marker-camera> Pri detekcii značky sa nastaví stred scény na značku a počíta sa vzdialenosť kamery od značky

Zatiaľ čo v prvom prípade môžeme mať v scéne viacero značiek, v druhom môže byť vždy iba jedna. Výhodou je intuitívnejšia práca so scénou, no za cenu možnosti mať iba jednu značku definujúcu celú scénu.

V ukážke kódu 4.1 je vyrobená „Hello world“ scéna v AR, spojením A-Frame a AR.js, iba pomocou niekoľkých riadkov HTML. [24]

```
1  <!doctype HTML>
2  <html>
3    <head>
4      <script src="https://aframe.io/releases/0.9.0/
           aframe.min.js"></script>
5      <script src="https://cdn.rawgit.com/jeromeetienne/
           AR.js/1.6.2/aframe/build/aframe-ar.js"></script>
6    </head>
7    <body style='margin : 0px; overflow: hidden;'>
8      <a-scene embedded arjs>
9        <a-marker preset="hiro">
10           <a-box position='0 0.5 0' material='color:
                 yellow;'></a-box>
11       </a-marker>
12       <a-entity camera></a-entity>
13     </a-scene>
14   </body>
15 </html>
```

Kód 4.1: Ukážka „hello world“ AR aplikácie

4.2 Vlastné komponenty

V práci sme implementovali niekoľko nových komponentov. Kvôli nefunkčnému volaniu funkcii po kliknutí na vlastné elementy mimo balíka `aframe-gui`, sme vytvorili triviálny komponent doplňujúci toto volanie. Ďalší komponent bol vytvorený pre správu modelov. Posledný komponent, ktorý sme vytvorili sa stará o rotáciu GUI.

4.2.1 cursor-listener

V balíku `aframe-gui` sa nachádza komponent, ktorý slúži na to, aby sa objekty dali zvoliť kurzorom. V prípade, že používateľ klikne na daný objekt, môžeme nastaviť funkciu, ktorá sa má zavolať. Tu prichádza problém, pretože komponent síce pridal objekt medzi tie, ktoré kurzor môže trafiť lúčom, no nereagoval na kliknutia. Preto bol vytvorený tento komponent, ktorý funkcionalitu dopĺňa.

```
1 init: function () {
2     this.el.addEventListener('click', function (evt) {
3         let clickActionFunction = window[this.
              getAttribute('cursor-listener').call];
4         if (typeof clickActionFunction === "function")
              clickActionFunction(this.id);
5     }, false);
6 }
```

Kód 4.2: Inicializácia komponentu `cursor-listener`

4.2.2 assets-set

Tento komponent sa stará o správu zoznamu modelov jednej budovy. Drží si index aktuálneho modelu a ak si to užívateľ vyžiada, zmení model na nasledujúci, resp. predchádzajúci. Na vytváranie zoznamov slúži anotácia pomocou priradenia triedy HTML elementu. Potomkovia `<a-assets>` s rovnakou triedou patria jednej budove. Komponent má v schéme uložený index na aktuálny model. Keď je zavolaná funkcia `next()`, komponent presunie index, zistí unikátne `id` novo-zvoleného modelu a zmení zdroj modelu v entite na scéne. Funkcia `previous()` funguje analogicky.

```
1 next: function () {
2         let modelSet = document.getElementsByClassName(
              this.el.id + ' asset');
3         let i = this.data.index;
4         i = (i + 1) % modelSet.length;
5         this.el.setAttribute('assets-set', { index: i
              });
6         let id = '#' + modelSet[i].id;
7         this.el.setAttribute('src', id);
8     }
```

Kód 4.3: Implementácia funkcie `next()`

4.2.3 face-front

Ak sa používateľ rozhodne otáčať modelom, očakáva, že GUI prvky nebudú rotáciou ovplyvnené. V grafe scény sú ale GUI prvky potomkami modelu,

preto pri rotácii modelu dochádza rovnako k rotácii GUI. Komponent má na starosti negovanie tejto rotácie. A-Frame využíva súradnicový systém tak ako to býva zvykom v hrách. Osa y je kolmá na zem. Preto musíme GUI rotovať iba po tejto osy. Samotnou rotáciou by sme nedosiahli požadovaného efektu, pretože model a GUI nemajú spoločný stred rotácie. Takže je potrebné GUI zároveň posunúť so zachovaním rovnakého odstupu. Miesto kam je potrebné ikony presunúť vypočítame jednoducho goniometricky na jednotkovej kružnici a vynásobíme posunom.

```
1 let model = this.el.parentEl;
2 let modelRot = model.getAttribute('rotation');
3 let guiRot = this.el.getAttribute('rotation');
4 let rot = modelRot.y;
5 this.el.setAttribute('rotation', {x: guiRot.x, y: -rot,
      z: guiRot.z});
6 let xCoord = Math.sin(rot * Math.PI / 180) * -this.data
    .offset;
7 let zCoord = Math.cos(rot * Math.PI / 180) * this.data.
    offset;
8 let oldCoord = this.el.getAttribute('position');
9 this.el.setAttribute('position', xCoord + ' ' +
    oldCoord.y + ' ' + zCoord);
```

Kód 4.4: Úprava rotácie a polohy GUI pri rotácii modelu

4.3 Interakcia

Na začiatku scenára práce s aplikáciou v sekcii 3.3, sme navrhovali, že používateľ pridá budovu do scény z galérie budov. Zatiaľ však galéria budov nie je k dispozícii, preto zatiaľ implementáciu a testovanie týchto krokov vynechávame. Aplikácia začína priamo v stave, kde na značke je pridaná maketa budovy.

Základom interakcie je kurzor, ktorý funguje ako používateľova „ruka“. Využívame kurzor s okrúhlym terčom, ktorý sa animáciou otvorí, ak sa nachádza nad objektom, na ktorý je možné kliknúť. Po kliknutí na model sa zavolá metóda `toggleGUI(eid)`, kde `eid` je identifikátor elementu. Tá najprv skontroluje, či element má medzi svojimi potomkami elementy rozhrania. Ak sa tam nenachádzajú zavolá metódu `createGUI(eid)`, ktorej ďalej posiela identifikátor elementu. Metóda vytvorí všetky časti a pridá GUI elementu modelu ako potomka. Ak GUI už existuje, tak zmení jeho stav viditeľnosti.

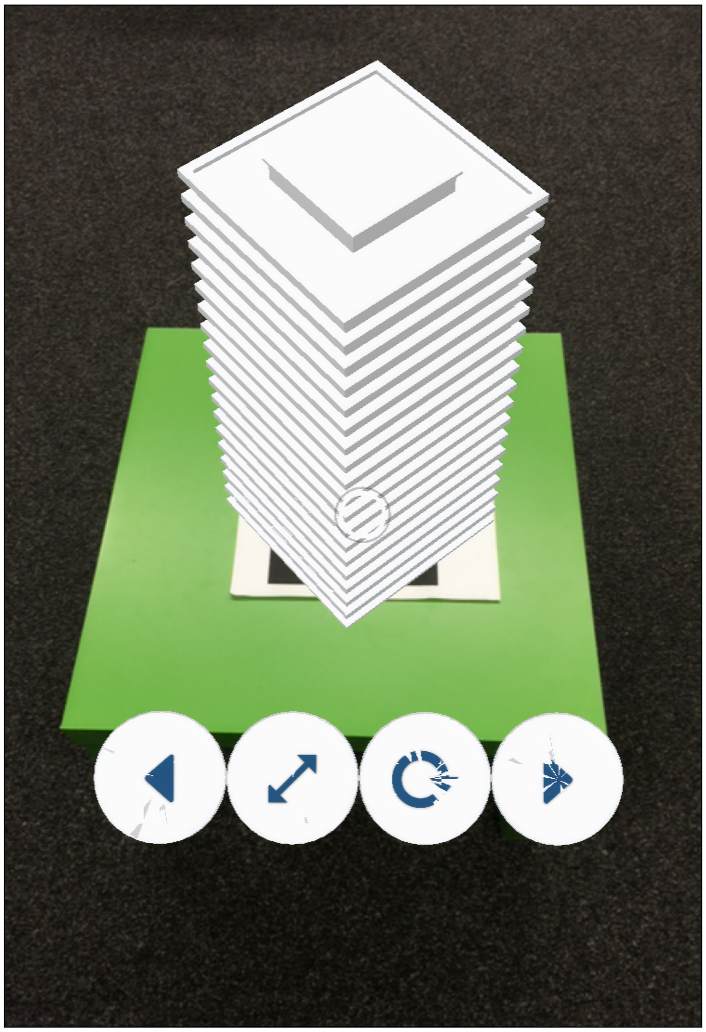
V základnom menu máme štyri prvky. Šípky vľavo a vpravo menia model na predchádzajúci, resp. nasledujúci. Kliknutím spustia funkciu, ktorá nájde model, ktorého sa interakcia týka. Následne je od komponentu `assets-set` vyžadovaná zmena modelu. Ďalšie dve ikony, ktoré slúžia na modifikáciu modelu,

nastavia parametre pre kontajnery prvkov rozhrania a spustia animáciu na zmenu rozhrania.

Po animácii už máme k dispozícii ikony, ktorými priamo môžeme ovplyvňovať model. Funkcie `rotateRight()` a `rotateLeft()` zmenia rotáciu modelu okolo osy y o 10^{o}. Funkcie `scaleDown()` a `scaleUp()` zmenia veľkosť modelu upravením hodnoty komponentu scale o hodnotu 0.1.

4.3.1 Užívateľské rozhranie

Samotný vzhľad užívateľského rozhrania sme vytvorili podľa návrhu. Stanovili sme si ako cieľ vytvorenie prototypu aplikácie, ktorej hlavným prvkom má byť pozorovanie budovy a jej premien v čase. Tomu odpovedá minimálne UI. Výsledný vzhľad prototypu môžeme vidieť na obrázku 4.2.



Obr. 4.2: Výsledný prototyp aplikácie

Kapitola 5

Testovanie

V každej aplikácii sa pri vývoji, i po nasadení, objaví nejaká chyba. Z toho dôvodu je veľmi potrebné aplikácie testovať. Môžeme testovať rôzne funkcionality a časti aplikácie, rôznymi metrikami. Testovanie AR aplikácii na mobilných zariadeniach môže byť obzvlášť náročné, ak testujeme aplikáciu priebežne. Po každej zmene, ktorú chceme testovať, je potrebné nahrať zmenu na zariadenie alebo webový server. Presunúť sa k mobilnému zariadeniu, spustiť novú verziu začať testovať. Samozrejme existujú spôsoby ako to uľahčiť. Pri vývoji aplikácie sme väčšinu funkcionalít testovali už pri vývoji vo VR, bez testovania na mobilnom zariadení. Je to jednou z výhod, ktoré framework A-Frame ponúka. Spustením lokálneho PHP servera je možné jednoducho upravovať graf scény, či dopísať skript a hneď to testovať vo vedľajšom okne. Podobne sa dá využiť aj webová služba glitch.com, ktorá je pomerne obľúbená v A-Frame komunite.

5.1 Testovanie kompatibility

Z požiadaviek N-02 a N-03 vyplynulo, že bolo potrebné otestovať zariadenia na spodnej hranici splniteľnosti. Problémom však bolo dostať sa k daným zariadeniam.

5.1.1 Android

Podarilo sa nám otestovať prototyp na zariadení Samsung Galaxy S5 s verziou 6.0.1 v prehliadači Chrome. Bohužiaľ z toho ale nemáme k dispozícii žiadne údaje z komponentu `stats`. Testovanie neprebiehalo v ideálnych svetelných podmienkach a beh aplikácie bol zjavne pomalší, odhadom pod 30 FPS. Ide však naozaj iba o odhad, ktorý je skôr iba dohad. Platí však, že prototyp bol funkčne spustený. Ďalšie testovanie bolo s omnoho novším zariadením Honor 8X, taktiež v prehliadači Chrome. Na tomto zariadení je OS s verziou 9.0. Testovanie prebiehalo za rovnakých podmienok ako v sekcii 5.1.2. Telefón

dosahoval stabilných 60 FPS, so zriedkavými poklesmi pod 55 FPS. Pri väčšom pohybe sa občas FPS prepadali k 40. Telefón pri testovaní nemal problém s registráciou značky.

5.1.2 iOS

Operačný systém iOS sme otestovali iba na jednom zariadení. Bol ním iPhone 6, čo je druhým najstarším mobilným zariadením spĺňajúcim požiadavku N-03. Najkritickejším zariadením z pohľadu tohto testu by mohol byť iPad mini, na tom sa nám však testovať nepodarilo. Spomínaný iPhone 6 bol testovaný v prehliadači Safari a za rovnakých svetelných podmienok ako Honor 8X. Počas pomalých pohybov telefón dosahoval cez 55 FPS, s občasnými prepadmi pod 50 FPS. Pri väčšom pohybe boli poklesy výraznejšie až pod 30 FPS. Nezaznamenali sme problémy s registráciou značky.

5.2 UX testovanie

V tomto teste sa chceme venovať rozboru rozdielov v kurzoroch. Máme malú vzorku 3 respondentov, takže nemôžeme závery zovšeobecniť. Respondenti odpovedali nezávisle od seba a ich odpovede boli pomerne podobné. Skreslenie mohlo vzniknúť na základe poradia testovaných variant.

Najprv testovali kurzor s klikaním dotykom. Sprvu respondenti, akoby prehliadli kurzor a skúšali klasické ovládanie gestami z plochých aplikácii. Neskôr koncept pochopili a osvojili si ho.

Druhým testovaným kurzorom bol kurzor s klikaním sledovaním cieľa. Používatelia už neboli tak zmetení a zvládli pracovať s prototypom rýchlejšie.

Aj napriek problematickému začiatku, v hodnotení po testovaní, označili za lepšiu variantu klikanie dotykom.

Kapitola 6

Záver

Cieľom tejto práce bolo analyzovať framework A-Frame a popísať jeho možnosti pre vytváranie mutliplatformných AR aplikácii. Ďalej bolo vyžadované vytvoriť na základe analýzy prototyp, ktorý umiestni do rozšírenej scény objekt, ideálne budovu. S touto budovou by malo byť možné interagovať. Nakoniec bolo požadované analyzovať pravidlá a doporučenia pri vytváraní užívateľského rozhrania a aplikovať ich na vytváraný prototyp.

Na začiatku bolo uvedené čo je rozšírená realita, ako sa dnes využíva a ako sa k nám vôbec dostala. Ďalej bol analyzovaný framework A-Frame, určením jeho návrhových vzorov a dostupných programovacích rozhraní. Následne boli porovnané možnosti využitia A-Frame pre AR. Stanovili sa funkčné a nefunkčné požiadavky na vytváranú aplikáciu. Na základe analýzy boli navrhnuté riešenia implementácie kritických častí aplikácie. Neskôr boli navrhnuté prvky rozhrania, vyplývajúce z analýzy pravidiel a doporučení pre tvorbu rozhrania pre rozšírenú realitu. Implementácia sa držala navrhnutých riešení, no bola zjednodušená pre účely testovania. Prototyp bol vytvorený frameworkom A-Frame a rozšírením AR.js. Na záver bol prototyp podrobený testovaniu. Testovania neboli síce dostatočného rozsahu, no načrtli pravdepodobné problémy prototypu.

6.1 Možnosti ďalšieho rozšírenia práce

Zaujímavým vylepšením by mohlo byť vytvorenie galérie budov, z ktorých by si používateľ vyberal. Ďalším možným vylepšením by mohlo byť pridanie anotácii k budovám. Používateľ by sa tak mohol dozvedieť niečo zaujímavé a nové. Možnosť zapnutia pomalej automatickej rotácie, by mohla zlepšiť zážitok z aplikácie a zároveň by mohla používateľa viac nabádať k tomu aby okolo modelu začal tiež hýbať.

Literatúra

[1] Milgram, P.; Takemura, H.; Utsumi, A.; aj.: Augmented Reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, 1994, [cit. 2019-05-03]. Dostupné z: http://etclab.mie.utoronto.ca/publication/1994/Milgram_Takemura_SPIE1994.pdf

[2] Sutherland, I. E.: A Head-mounted Three Dimensional Display. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), New York, NY, USA: ACM, 1968, s. 757–764, doi:10.1145/1476589.1476686. Dostupné z: http://doi.acm.org/10.1145/1476589.1476686

[3] Marcos, D.; McCurdy, D.; Ngo, K.: *Introduction – A-Frame*. A-Frame, 2019, [cit. 2019-05-07]. Dostupné z: https://aframe.io/docs/0.8.0/introduction/

[4] TARGO: Step inside Notre-Dame cathedral [VR/360]. 2019, [cit. 2019-05-04]. Dostupné z: https://www.youtube.com/watch?v=zLuVsFGXukc

[5] Webster, A.: Building a better Paris in Assassin's Creed Unity. *The Verge*, 2014, [cit. 2019-05-04]. Dostupné z: https://www.theverge.com/2014/10/31/7132587/assassins-creed-unity-paris

[6] RealityTechnologies.com: What is Augmented Reality (AR)? Ultimate Guide to Augmented Reality (AR) Technology. [cit. 2019-05-04]. Dostupné z: https://www.realitytechnologies.com/augmented-reality/

[7] Kato, H.: Return to the origin of Augmented Reality. [cit. 2019-05-04]. Dostupné z: https://www.youtube.com/watch?v=b33eqcVz7X8

[8] Milgram, P.; Kishino, F.: A Taxonomy of Mixed Reality Visual Displays. *IEICE Trans. Information Systems*, ročník vol. E77-D, no. 12, 12 1994: s. 1321–1329.

[9] ARGeo (https://stackoverflow.com/users/6599590/argeo): Are there any limitations in Vuforia compared to ARCore and ARKit? 2018, [cit. 2019-05-04]. Dostupné z: https://stackoverflow.com/questions/50811770/are-there-any-limitations-in-vuforia-compared-to-arcore-and-arkit

[10] Miesnieks, M.: How is ARCore better than ARKit? *Medium*, 2017, [cit. 2019-05-04]. Dostupné z: https://medium.com/6d-ai/how-is-arcore-better-than-arkit-5223e6b3e79d

[11] Miesnieks, M.: Why is ARKit better than the alternatives? *Medium*, 2017, [cit. 2019-05-04]. Dostupné z: https://medium.com/6d-ai/why-is-arkit-better-than-the-alternatives-af8871889d6a

[12] Song, S.; Chandraker, M.; Guest, C.: Parallel, real-time monocular visual odometry. In *Robotics and Automation*, 05 2013, ISBN 978-1-4673-5641-1, s. 4698–4705, doi:10.1109/ICRA.2013.6631246.

[13] Sung, D.: The history of augmented reality. *Pocket-lint*, 2011, [cit. 2019-05-04]. Dostupné z: https://www.pocket-lint.com/phones/news/108888-the-history-of-augmented-reality

[14] Kato, H.: *ARToolKit Documentation*. 2003, [cit. 2019-05-04]. Dostupné z: http://www.hitl.washington.edu/artoolkit/documentation/

[15] Valo Motion Oy: ValoClimb: Augmented Climbing Wall. 2018, [cit. 2019-05-03]. Dostupné z: http://valomotion.com/wp-content/uploads/2018/03/Augmented_Climbing_Wall_Games_Climball_2_cropped-e1520234130186.jpg

[16] Mitsuno, D.; Ueda, K.; Itamiya, T.; aj.: Intraoperative Evaluation of Body Surface Improvement by an Augmented Reality System That a Clinician Can Modify. *Plastic and Reconstructive Surgery –Global Open*, ročník 5, č. 8, 2017, [cit. 2019-05-04].

[17] BBC News: Nintendo shares up 50% on Pokemon Go. *BBC News*, 2016, [cit. 2019-05-04]. Dostupné z: https://www.bbc.com/news/business-36791275

[18] Williams II., D.: The History of Augmented Reality (Infographic). *HuffPost*, 2016, [cit. 2019-05-04]. Dostupné z: https://www.huffpost.com/entry/the-history-of-augmented_b_9955048

[19] Mr.doob (https://github.com/mrdoob): *three.js*. [cit. 2019-05-07]. Dostupné z: https://threejs.org/docs/

[20] W3Schools: *HTML DOM Document Objects*. 2019, [cit. 2019-05-10]. Dostupné z: https://www.w3schools.com/jsref/dom_obj_document.asp

[21] Murphy, W.: aframe-machinisma-testing. 2018, [cit. 2019-05-12]. Dostupné z: https://github.com/wmurphyrd/aframe-machinima-testing/

[22] A-Frame Community: A-FRAME Registry. [cit. 2019-05-13]. Dostupné z: https://aframe.io/aframe-registry/

[23] npm, Inc.: aframe-component - npm search. [cit. 2019-05-14]. Dostupné z: https://www.npmjs.com/search?q=aframe-component&page=1&ranking=optimal

[24] Etienne, J.: Augmented Reality for the Web. [cit. 2019-05-07]. Dostupné z: https://jeromeetienne.github.io/AR.js/

[25] Chenzlabs: GitHub - chenzlabs/aframe-ar. [cit. 2019-05-13]. Dostupné z: https://github.com/chenzlabs/aframe-ar

[26] Kato, H.; Billinghurst, M.: Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, Oct 1999, s. 85–94, doi:10.1109/IWAR.1999.803809, [cit. 2019-05-04].

[27] Apple Inc.: *Augmented Reality - System Capabilities - iOS - Human Interface Guidelines - Apple Developer*. 2019, [cit. 2015-05-07]. Dostupné z: https://developer.apple.com/design/human-interface-guidelines/ios/system-capabilities/augmented-reality/

[28] Google Developers: Best practices to design AR applications (Google I/O '18). [cit. 2019-05-13]. Dostupné z: https://www.youtube.com/watch?v=bNJJCREZgVM

[29] Deveria, A.: Can I use... Support tables for HTML5, CSS3, etc. [cit. 2019-05-14]. Dostupné z: https://caniuse.com

[30] Dubois (rdub80), R.: *aframe-gui*. 2018, [cit. 2019-05-07]. Dostupné z: https://github.com/rdub80/aframe-gui

Dodatok **A**

Zoznam použitých skratiek

3D Trojrozmerný

6DOF 6 Degrees of freedom

API Application programming interface

AR Rozšírená realita

CDN Content delivery network

COM Concurrent odometry and mapping

CORS Cross-origin resource sharing

CSS Cascading style sheet

DOM Document object model

ECS Entity-component-system

FPS Frame per second

GPS Globálny polohový systém

GUI Graphical user interface

HMD Head-mounted display

HTML Hypertext markup language

IMU Inertial measurement unit

OS Operačný systém

PBR Physically based rendering

QR Quick response

RFID Identifikácia rádiovou frekvenciou

SDK Software development kit

SLAM Simultánna lokalizácia a mapovanie

UI User interface

UX User experience

VR Virtuálna realita

XHR XMLHttpRequest

Dodatok B

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF