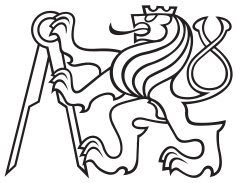
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Multiplatformní aplikace v rozšířené realitě pro vizualizaci satelitů Země

Vojtěch Pospíšil

Vedoucí: Ing. David Sedláček, Ph.D.
Obor: Otevřená informatika
Studijní program: Počítačové hry a grafika
Květen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pospíšil** Jméno: **Vojtěch** Osobní číslo: **478156**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Studijní obor: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Multiplatformní aplikace v rozšířené realitě pro vizualizaci satelitů Země

Název bakalářské práce anglicky:

Multiplatform augmented reality application for Earth satelite visualization

Pokyny pro vypracování:

Prostudujte vhodné multiplatformní frameworky pro rozšířenou realitu (AR) s ohledem na využití AR pro sledování reálného objektu (ne jenom plochý předmět). Navrhněte vzhled objektu, který se bude pomocí AR rozšiřovat (např. glóbus).
Pomocí rozšířené reality vizualizujte aktuální polohy umělých satelitů Země a významných objektů tzv. pozemního segmentu.
Implementujte načítání informace o satelitech z webové služby, která tato data poskytuje. Stažená data si bude aplikace uchovávat pro příští spuštění, aby nemusela vždy přistupovat k internetu.
Navrhněte a implementujte rozhraní pro výběr zobrazených satelitů, dle skupin (např. GPS, GLONASS, Sentinel,..) a také dle jména, případně dalších parametrů. Pro významné satelity (po dohodě s vedoucím) vyberte vhodné modely pro vizualizace.
Při návrhu a implementaci uživatelského rozhraní postupujte dle metodiky UCD (User Center Design) a aplikaci otestujte s cílovou skupinou uživatelů.

Seznam doporučené literatury:

[1] Augmented Reality Game Development, Micheal Lanham, Packt Publishing, 2017
[2] Virtual Reality & Augmented Reality in primary education: Robin De Lange, Maarten Lodewijk, Nesse van der Meer, 2017
[3] Practical Augmented Reality: Steve Aukstakalnis, 2017, Addison Wesley

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. David Sedláček, Ph.D., katedra počítačové grafiky a interakce FEL

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

Datum zadání bakalářské práce: **17.02.2020** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce: **19.02.2022**

Ing. David Sedláček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. | Datum převzetí zadání

Podpis studenta

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu Ing. Davidu Sedláčkovi, Ph.D. za odborné vedení, za pomoc a cenné rady při zpracování této práce. Zároveň bych také rád poděkoval své rodině za podporu v průběhu celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 10. května 2020

Abstrakt

Tato bakalářská práce se zabývá vývojem multiplatformní mobilní aplikace s využitím prvků rozšířené reality pro vizualizaci poloh satelitů Země a prvků pozemního segmentu GPS. Cílem práce je seznámit čtenáře s možnostmi multiplatformního mobilního vývoje, frameworky pro vývoj aplikací využívajících rozšířené reality a s daty a procesy potřebnými k výpočtu pozic satelitů. Na základě analýzy těchto částí byla vyvinuta aplikace umožňující zmíněnou vizualizaci. Tato aplikace byla následně otestována s uživateli a na základě zpětné vazby patřičně upravena.

Klíčová slova: rozšířená realita, AR, mobilní aplikace, React Native, Viro React, vizualizace, satelity

Vedoucí: Ing. David Sedláček, Ph.D.

Abstract

This thesis deals with the development of a multiplatform mobile application visualizing the positions of Earth satellites and GPS ground segment using augmented reality. The thesis aims to acquaint the reader with the possibilities of multiplatform mobile development, frameworks for the augmented reality application development, and with the data and processes needed to calculate satellite positions. Based on the analysis of these parts, an application was developed to enable the mentioned visualization. The application was then tested with users and modified accordingly based on user feedback.

Keywords: augmented reality, AR, mobile application, React Native, Viro React, visualization, satellites

Title translation: Multiplatform augmented reality application for Earth satellite visualization

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Tato práce se zaměřuje na porovnání dostupných možností vývoje mobilních multiplatformních aplikací využívajících rozšířenou realitu. Porovnává jednotlivé technologické možnosti a vybírá z nich ty nejvhodnější pro vývoj na nejrozšířenější mobilní platformy reprezentované operačními systémy Android a iOS. Rozšířené reality je využito k netradiční vizualizaci satelitů Země a prvků takzvaného pozemního segmentu GPS. Základem této vizualizace je skutečný globus, který je doplněn prvky rozšířené reality pro znázornění pozic jednotlivých satelitů, jejich orbit a prvků pozemního segmentu GPS. Tato vizualizace je doplněna informacemi o jednotlivých objektech a může sloužit například jako učební pomůcka pro žáky základních a středních škol. Své místo nalezne také mezi nadšenci o vesmír a lidskou techniku v něm, popřípadě mezi zájemci o rozšířenou realitu, protože se jedná o jedinou aplikaci na principu propojení reálného globusu a vizualizace pozic satelitů kolem něj.

1.1 Motivace a cíle

Technologický pokrok se dotýká téměř každé oblasti lidského působení a jednou z nich je právě i vzdělávání. Se zvyšující se dostupností mobilních zařízení a nárůstem jejich výkonu, se otevírají nové možnosti pro jejich využití jako netradičních učebních pomůcek. Pokud navíc dojde ke spojení s virtuální (virtual, VR) či rozšířenou (augmented, AR) realitou, která má mezi populací stále jakousi herní pověst, může vzniknout netradiční učební pomůcka, která žáky uchvátí mnohem více než například obyčejná mapa. Zároveň například vizualizace v této podobě může informace zobrazovat v širší perspektivě a žáci s ní mohou interagovat, což může podnítit ještě větší zájem. Hlavním cílem této práce je tedy vytvoření multiplatformní mobilní aplikace využívající prvků rozšířené reality pro vizualizaci satelitů Země a pozemního segmentu GPS. Tato vizualizace bude využívat globus jako podkladový objekt pro zmíněnou vizualizaci. Kromě samotné vizualizace satelitů bude aplikace sloužit i jako zdroj informací o jednotlivých prvcích. S tím souvisí i další cíle mezi které

patří informování čtenáře o možnostech vývoje multiplatformních aplikací a frameworků umožňující tyto aplikace doplnit rozšířenou realitou. A dále také přiblížení problematiky sledování a počítání pozic satelitů Země a s tím souvisejících datových zdrojů, formátů a matematických modelů.

1.2 Struktura práce

Tento text je dělen do šesti kapitol kde kapitola 1 je tento úvod. Kapitola 2 se zabývá průzkumem technologií týkajících se vývoje multiplatformních mobilních aplikací a frameworků pro rozšířenou realitu. Dále rozebírá také potřebné technologie pro vizualizaci a poskytování informací o satelitech, tedy zdroje pro získání potřebných dat, jejich formát a následné matematické zpracování. Kapitola 3 se zabývá návrhem samotné aplikace. Rozebírá detailně cílovou uživatelskou skupinu, její požadavky a návrh uživatelského rozhraní. Tento návrh je následně zpracován a realizován do výsledné aplikace. Funkční detaily a popis jednotlivých prvků je rozebrán v kapitole 4 týkající se samotné implementace. Výsledná aplikace byla otestována s referenční uživatelskou skupinou. Metodika, průběh a výsledky tohoto testování jsou zpracovány v kapitole 5. Poslední kapitola 6 se zabývá zhodnocením celé práce a nastíněním možných budoucích rozšíření.

Kapitola 2

Analýza

V této kapitole se budu zabývat analytickou částí práce. Konkrétně se jedná o analýzu vhodných technologií a postupů pro multiplatformní vývoj mobilních aplikací, analýzu dostupných frameworků pro rozšířenou realitu na mobilních zařízeních a následně o přiblížení technologií pro poskytování informací o satelitech Země. Zejména o služby poskytující data pro tyto účely, formát těchto dat a jejich následné zpracování pomocí matematických modelů.

2.1 Rozšířená realita

Rozšířená realita je označení pro reálný obraz světa doplněný, jinými slovy rozšířený, o virtuální objekty. Tyto objekty mohou být využity jak konstruktivně, to znamená že nějakým způsobem doplňují prostředí, tak i destruktivně k zakrytí či pozměnění části skutečného světa [32]. Rozdílem oproti virtuální realitě je tedy to, že zatímco ve virtuální realitě je vše, co vidí uživatel kompletně virtuální, v rozšířené realitě je vždy nějaká část založena na skutečném vnímání okolí uživatele, které je až následně doplněno virtuálními objekty.

Základním principem je tedy snímání okolního světa pomocí kamery připojené k výpočetnímu zařízení, kterým může být jak počítač, tak v dnešní době díky nárůstu výpočetního výkonu i mobilní telefon. V této scéně mohou být detekovány předem definované znaky a v případě mobilních telefonů jsou využívána i data z dostupných senzorů určující například pozici a natočení telefonu ve scéně. Na základě těchto detekcí jsou poté do prostoru vloženy v reálném čase virtuální objekty (text, 2D či 3D objekty, video, zvuky).

Trh rozšířené reality má nakročeno ke zdárné budoucnosti, kdy odborníci odhadují nárůst trhu ze zhruba 4.32 miliardy USD v roce 2017 na celkových 149 miliard USD v roce 2025 [2]. Jedná se tedy o trh s velkým potenciálem růstu a tedy i potenciálně velkou cílovou skupinou.

2.1.1 Dělení

Jedním z možných dělení rozšířené reality je dělení dle způsobu aktivace akcí spouštěných v aplikaci. Akcí je myšleno například zobrazení 3D modelu, okna s informacemi, spuštění videa, apod. Tímto dělením se nadále budu zabývat. Dále AR můžeme dělit například podle využité technologie (mobilní telefon, Head-Up display, projekce, ...) či lokace kde se rozšíření využívá (vnitřní, venkovní) [6].

Marker-based AR Tento druh využívá předem definovaných speciálních značek tzv. markerů, které se využívají jako spouštěče akcí v rozšířené realitě. Tyto speciální značky (viz obrázek 2.1) musí být předem zaregistrovány v příslušném software a mít přiřazenou akci, která se spustí při jejich detekci. Jako značky většinou využíváme unikátní dvourozměrné obrazce, které lze snadno generovat a případně do nich umístit dodatečné informace. Často je značka využívána jako reference pro zobrazení 3D objektu, který si uživatel následně prohlíží a díky interakci s markerem (otáčení, přemisťování) může pohybovat i virtuálním objektem.

Obrázek 2.1: Příklady několika různých markerů [4]

Rozpoznávání vlastností objektů V tomto případě se místo detekce markerů využívá detekce obrázků, 3D objektů, či dalších vlastností které jsme schopni programově detekovat. Tento způsob by mohlo být možné řadit do kategorie marker-based jelikož jde o stejný princip, pouze nedetekujeme marker ale jiný objekt.

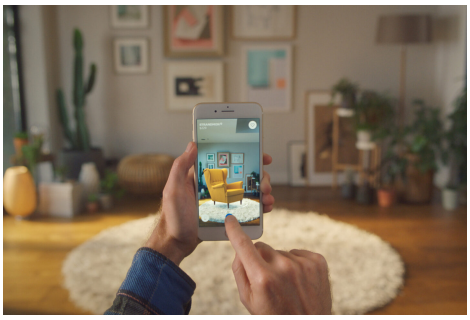
Marker-less AR Tato kategorie označuje všechny ostatní druhy, které nepoužívají jako základ detekci markerů pro pozice objektů v rozšířené realitě. Hlavním principem na pozadí tohoto druhu rozšířené reality je tzv. SLAM (Simultaneous Localization And Mapping) technologie, která je založená na principu snímání okolí, které je využito k tvorbě virtuální mapy prostředí a současné lokalizaci uživatele v něm.

Rozšířená realita založená na lokaci Ke spouštění předem definovaných akcí můžeme využít i data ze senzorů zařízení. Vhodným příkladem jsou právě data z GPS senzoru zkombinovaná dále například s daty z digitálního kompasu a akcelerometru. Tento druh rozšířené reality je oblíbený zejména v rámci mobilních her, kde nejznámějším příkladem je PokemonGO. Hlavní interakce v těchto hrách spočívá v dostavení se na konkrétní místo a provedení akcí v rozšířené realitě.

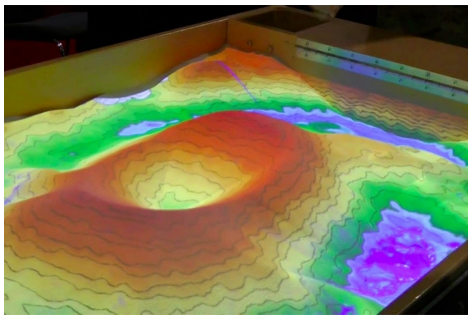
Poddruhem by se dala označit rozšířená realita založena na využití tzv. beacons (majáků). Jedná se o zařízení vysílající signál, který je například telefon schopen detekovat a na základě identifikace detekovaného zařízení spouštět předem definované akce. Nejčastěji je toto řešení používáno pro audio/video průvodce v muzeích a galeriích [19].

Rozšířená realita založená na uživatelském vstupu Tento způsob je založen čistě na uživatelské interakci, kde si z předem daného seznamu například modelů některý vybere a umístí ho sám do scény, kde je poté ukotven a je možné si ho prohlížet zasazený do skutečného světa [14]. Využití je hlavně ve vizualizaci 3D objektů jako je třeba nábytek (viz obrázek 2.2a), auta nebo zvířata.

Rozšířená realita založená na projekci Ne zcela zřejmou kategorií je projekční rozšířená realita. Jedná se o promítání na objekty v reálném světě. Při projekci je využíváno rysů ploch, na které se promítá, například budov, či oblíbené jsou instalace miniaturních pískových hřišť (viz obrázek 2.2b) . Další možností je tvorba 3D hologramů pomocí laserové technologie a umělé mlhy.



(a) : IKEA Place [14]



(b) : Projekční AR

Obrázek 2.2: Ukázky AR aplikací

2.1.2 Využití

Oblasti využití rozšířené reality jsou prakticky neomezené. Dalo by se říci, že záleží pouze na představivosti tvůrců. Důkazem toho je i rozmanité spektrum ve kterém našly tyto aplikace uplatnění dodnes. Kvůli velkému množství uvedu jen výčet několika z nich.

Marketing a e-shopy Aplikací využívající rozšířenou realitu se dá doplnit prodej téměř libovolného zboží. Ať se jedná například o sběratelské kartičky, kde se po naskenování kartičky může například objevit hráč právě z té karty, provádějící nějaký jemu ikonický pohyb. Při prodeji nábytku je možné z katalogu vybrat kusy zboží které nás zajímají a

zkusit, jak by vypadaly umístěné u nás doma bez toho, aniž by bylo nutné je kupovat. Nebo je možné si před zrcadlem zkoušet oblečení bez nutnosti si ho oblékat [23].

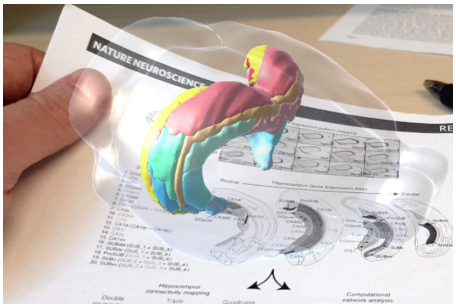
Armáda Jako je tomu ve spoustě oblastí lidského působení i u rozšířené reality se stala jedním z prvních uživatelů armáda. Jedno z možných využití je v případě tzv. head-up displejů (viz obrázek 2.3a). Jedná se o zařízení umožňující zobrazení informací přímo v zorném poli uživatele (pilota, řidiče). Tyto displeje se poslední dobou rozšiřují i například do osobních automobilů. Nicméně není to jediný způsob využití, dále lze najít využití například v podobě simulací cvičení či vizualizace rozmístění jednotek (viz obrázek 2.3b) [22].

(a) : HUD stíhacího letounu F/A-18

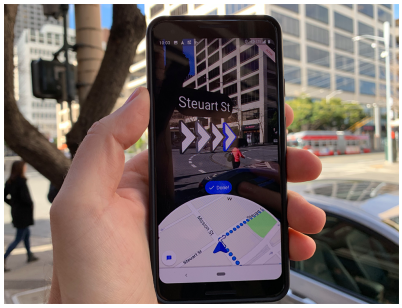
(b) : Simulace cvičení v AR [11]

Obrázek 2.3: Využití AR v armádě

Vzdělání Ve školství je rozšířená realita používána jako rozšíření učebních pomůcek. Učební pomůcky tak mohou být doplněny markery v podobě kódů, nebo vybraných obrázků, které po naskenování umožní zobrazení doplňujících informací. Nicméně nemusí se jednat pouze o doplnění učebnic (obrázek 2.4a), ale i samostatné aplikace sloužící jako vizualizace různých probíraných kapitol například ve strojírenství či biologii [5]. Do této kategorie by se později dala řadit i tato práce.



(a) : Využití AR v textu [40]



(b) : Google Maps AR [20]

Obrázek 2.4: Příklady využití AR pro širokou veřejnost

Zábavní průmysl Pro většinu lidí se jedná o oblast se kterou se setkávají nejvíce, díky tomu že do této kategorie spadají mobilní a počítačové hry. Na rozdíl od virtuální reality, kde jsou hry právě většinou pro počítače a žádají si vysoce výkoné počítače s drahými speciálními doplňky, díky čemuž nejsou tak dostupné široké veřejnosti, se v případě rozšířené reality jedná převážně o mobilní hry, kde se využívá geolokace, zpracování dat ze senzorů telefonu a prvků rozšířené reality pro hry jako již zmíněné PokemonGO (obrázky 2.5) a další.

(a) : Část hry založená na geolokaci **(b) :** Část hry v AR

Obrázek 2.5: Mobilní AR Hra Pokémon GO

Další oblastí je například lékařství, architektura, turismus a mnoho dalších odvětví, která časem budou zajisté přibývat.

2.2 Technologie

V této části se zaměřím na rozbor dostupných technologií a principů multiplatformního vývoje a frameworků pro rozšířenou realitu. Následně zdůvodním výběr konkrétních technologií pro tuto práci a blíže je představím.

2.2.1 Platforma

V dřívějších dobách byla virtuální či rozšířená realita omezena díky velké výpočetní náročnosti pouze na použití v osobních počítačích. Nicméně s technologickým pokrokem se nese ruku v ruce i zvyšování výkonu jak počítačů, tak hlavně mobilních telefonů, a tudíž je možné používat rozšířenou či virtuální realitu právě zde. Nicméně dle upadající frekvence představování periferií pro VR na telefonu a naopak se vzrůstajícím počtem těchto headsetů registrovaných ve službě Steam lze soudit, že u virtuální reality je kvalita zobrazení na prvním místě před cenou a virtuální realita zůstane devizou počítačů [21]. Oproti tomu použití rozšířené reality, kde je předpoklad snímání okolí uživatele, je právě z tohoto důvodu pravděpodobnější na mobilním zařízení.

Vizualizace má probíhat v rozšířené realitě na mobilním telefonu, tak se zaměřme pouze na mobilní platformu . V současné době je trh s mobilními telefony zastoupen primárně dvěma operačními systémy, kterými jsou Android od společnosti Google a jeho protějšek od Apple, systém iOS. Na celkovém trhu se tyto systémy podílejí ze zhruba 99 % (přesněji v tabulce 2.1). Budeme se tedy zabývat pouze jimi.

Datum	Android	iOS	Ostatní
Prosinec 2019	74,13	24,79	1,08
Leden 2020	74,30	24,76	0,94
Únor 2020	73,30	25,89	0,81
Březen 2020	72,26	27,03	0,71
Duben 2020	70,68	28,79	0,53

Tabulka 2.1: Podíl mobilních operačních systémů na trhu (v %) [25]

2.2.2 Multiplatformní vývoj

Po zaměření vývoje na platformy Android a iOS vyvstává otázka jakou techniku vývoje zvolit. V některých případech se může hodit vytvořit dvě nativní aplikace (tj. aplikaci navrženou a programovanou přímo pro danou platformu). Na první pohled jednoduché řešení s sebou však nese i komplikace. Jelikož se jedná vývoj dvou aplikací je nutné udržovat dva různé zdrojové kódy v různých jazycích (např. Swift – iOS; Kotlin - Android). S tím roste jak časová, tak i finanční náročnost následných oprav či možností rozšíření, protože je vše nutné provádět dvakrát pro každou platformu zvlášť. Nicméně výhodou těchto aplikací je bezesporu jejich rychlost právě díky využití nativních funkcí [30].

Další možností by bylo vytvořit klasickou webovou aplikaci, kde by se data posílala na server, tam by docházelo k jejich zpracování a následnému vrácení uživateli. To ovšem v případě interaktivní vizualizační aplikace není úplně vhodný postup a nebudeme se jím tedy vůbec zabývat.

Možným řešením by mohly být tzv. hybridní aplikace. Hybridní proto, protože se jedná o aplikaci napsanou s pomocí webových technologií (primárně HTML, CSS a JavaScript), ale tato aplikace běží lokálně na zařízení uživatele a využívá webWiew komponenty (kontejner umožňující využití jádra prohlížeče) pro zobrazení HTML a jeho doplnění ostatními jazyky. Tato aplikace ovšem není webovou stránkou a jde stáhnout v obchodě s aplikacemi, stejně jako by byla nativní.

Poslední, a i zvolenou možností je vytvoření nativní aplikace s pomocí jednoho univerzálního jazyka. Na rozdíl od hybridní aplikace zde není použita webView komponenta pro vykreslení HTML prvků, ale veškeré ovládací prvky jsou přeloženy na nativní ovládací prvky daného operačního systému. Na samotném zařízení se využívá vícevláknového zpracování a některá vlákna se tak starají o interpretaci zvoleného jazyka a následné volání nativních API.

2.2.3 Univerzální jazyky a frameworky

Tyto jazyky zažívají v posledních letech růst právě díky jednoduššímu vývoji a jejich založení na jazycích a technologiích které jsou většině vývojářům známé například z webového prostředí. Pro výběr toho správného musíme zvážit dostupnost knihoven, dokumentace a také osobní preference vůči danému jazyku.

Jednou z nejpopulárnějších možností je **React Native**[1]. Jedná se o open-source framework vyvíjený Facebookem od roku 2015. Je založený na populární JavaScriptové knihovně React oblíbené z webového prostředí. Díky tomu, a také díky Facebooku v zádech, má tento framework velmi rozšířenou dokumentaci a vývojářskou podporu, kterou si zaslouží především díky své rychlosti a flexibilitě. Pro webového vývojáře je tato technologie výhodná díky využití stejných principů jako v samotném Reactu s výjimkou, že v React Native nemanipulujeme přímo s DOM, ale proces v pozadí interpretuje Javascript v kombinaci s XML, dohromady známé jako JSX. Pomocí implementovaného mostu mezi Javascriptem a konkrétní platformou zajistíme, že aplikace bude používat opravdu nativní prvky platformy.

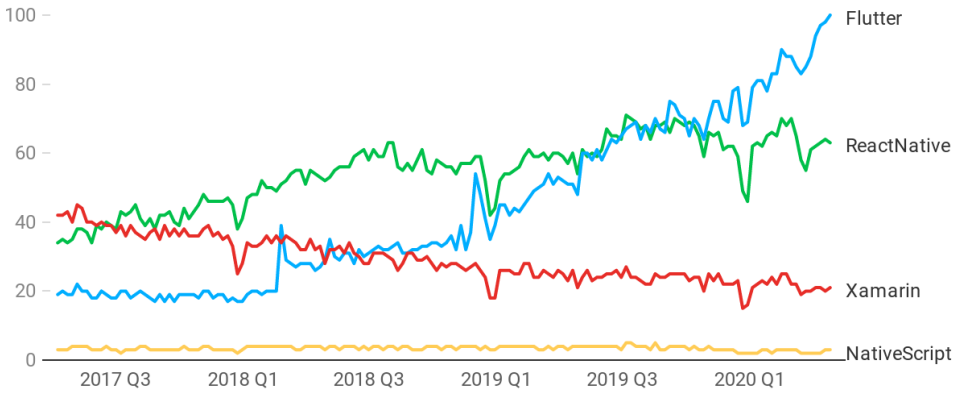
Alternativou ke zmíněnému je například **NativeScript**[2]. Opět jde o open-source framework, nicméně tento není založen na Reactu, ale na populární konkurenci Angular v kombinaci s Vue.js. Schopnosti tohoto frameworku jsou srovnatelné s React Native, ten má ovšem mnohem rozšířenější bázi uživatelů. Záleží tedy na našich preferencích, či specifických požadavcích při rozhodování mezi těmito možnostmi.

Další možností je open-source alternativa od Google, a to v podobě frameworku **Flutter**[3]. Ze zatím zmíněných se jedná o nejmladšího zástupce,

[1] `https://facebook.github.io/react-native/`
[2] `https://www.nativescript.org/`
[3] `https://flutter.dev/`

první verze vyšla v roce 2017. Dalším odlišujícím znamením je použití programovacího jazyka Dart. Jedná se objektově-orientovaný jazyk se syntaxí podobnou jazyku C s možností transkompilace do JavaScriptu. Výhodou Flutteru oproti React Native je absence tzv. bridge (mostu) mezi JavaScriptem a cílovou platformou. Dart je přímo překládán do strojového kódu, a tudíž je rychlejší. I přes jeho nevelké stáří se dostává do popředí a vývojáři se k použití uchylují čím dál tím více, viz graf na obrázku 2.6.

Dalším z největších zástupců je opět open-source řešení, tentokrát od Microsoftu a to **Xamarin**[4]. Přestože se jedná o open-source řešení, zdarma je dostupný pouze v Community verzi pro jednotlivce a menší týmy. Pro ostatní se tedy jedná o placenou technologii. Xamarin je založen na jazyce C# společně s .NET frameworkem a rozšířením XML – XAML. Při sestavení samotné aplikace se právě tyto jazyky přeloží do nativních jazyků pro každou platformu (Kotlin pro Android a Swift pro iOS). Výhodou je kompletní vývojový ekosystém Microsoftu a perfektní výkon srovnatelný s nativními aplikacemi. Nicméně jsou tu i zápory v podobě větší velikosti aplikací, nutnosti přepsat některé části kódu do jazyka pro konkrétní platformu a také již zmíněná licenční politika.



Obrázek 2.6: Zájem v průběhu času dle Google Trends

Framework, který můžeme použít ve specifických případech je **Unity**[5]. Jedná se o multiplatformní herní engine s podporou osobních počítačů, Mac, mobilních telefonů, herních konzolí a dalších více než 25 platforem. [26] Vešekeré skripty jsou psané za pomocí jazyka C#. Nejedná se o open-source řešení, nicméně pro jednotlivce a menší týmy je framework dostupný zdarma, větší týmy musí využít placenou verzi. Vzhledem k tomu, že se jedná o herní engine je jeho určení vcelku jasné, v našem případě by pro vizualizaci byl použitelný ovšem například pro aplikace zobrazující pouhá data z databáze atd. by se jednalo o zbytečně veliký moloch.

Výše zmíněné frameworky rozhodně nejsou jediné. Nicméně dle uživateských reakcí na webu StackShare se jedná o výběr z těch nejpopulárnějších [45].

[4] `https://dotnet.microsoft.com/apps/xamarin`
[5] `https://unity.com/`

2.2.4 SDK pro rozšířenou realitu

Zkratka SDK označuje Software Development Kit (česky možno přeložit jako sada vývojových nástrojů). Jedná se o sadu nástrojů, knihoven, dokumentace a návodů umožňující vývojářům vývoj pro konkrétní platformu.

Za dvě hlavní SDK pro rozšířenou realitu by se mohlo považovat ARCore tvořené Googlem a ARkit od Apple, jelikož se jedná o SDK právě od vlastníků dané platformy.

Zaměřím se nejdříve na popis **ARCore**[6], tedy pohledu Google na AR. Toto SDK vychází z předchozích pokusů Googlu s rozšířenou realitou, konkrétně jde o Google Tango, platformu vyvíjenou od roku 2014 do roku 2018 kdy se přetransformovala do ARCore. I přesto že jde o SDK od samotného Google ne každý telefon s operačním systémem Android je podporován. Podporovány jsou pouze telefony které projdou certifikací Googlu, a tedy splňují určité kvality obrazu, mají dostatečený výkon a senzory pro zprostředkování plnohodnotného AR zážitku. Toto SDK nám poskytuje primárně tři klíčové schopnosti a to:

- sledování pohybu (motion tracking) pro zpracování pozice v reálném světě,
- rozpoznávání prostředí (environmental understanding) pro detekci povrchů a vlastností okolí,
- odhad osvětlení (light estimation) pro odhad okolního osvětlení.

V základu tedy ARCore dělá dvě věci, sleduje pozici telefonu a tvoří si vlastní model okolního světa. Právě zmíněné sledování pohybu umožňuje skrze kameru telefonu identifikovat body v prostoru, sledovat jak se mění v čase a v kombinaci s vnitřními senzory telefonu správně detekovat, jak se zařízení pohybuje v reálném světě. Díky rozeznávání vlastností prostředí (sledování ploch, detekce osvětlení atd.) a zmíněné detekci pohybu si ARCore tvoří vlastní model okolního světa díky kterému je možné do něj umisťovat objekty které tvoří rozšířenou realitu. Jde o různé 3D modely či popisky, které díky sledování pohybu zůstávají ukotvené na místě a je možné je prohlížet jako skutečné objekty.

Výhodou ARCore může být i to, že primárně se sice jedná o SDK pro platformu Android, nicméně existují cesty jak jej používat i na zařízeních s iOS.

Odpovědí Apple je **ARkit**[7]. Jde o SDK pro tvorbu AR aplikací pro platformu iOS. Díky menšímu počtu různých zařízení s tímto operačním systémem je podpora mnohem lepší, jde o všechna zařízení s čipem A9 a novějším (nejstarším podporovaným zařízením je iPhone 6s z roku 2015). Podpora

[6] `https://developers.google.com/ar`
[7] `https://developer.apple.com/augmented-reality/`

nejnovější verze ARKit3 je menší, podmínkou je procesor A12 a novější což splňují zařízení vydaná v době iPhone XS a déle.

Schopnosti toho SDK jsou na stejné úrovni jako u ARCore. Existují různé nezávislé testy ukazující, v jaké oblasti vyniká konkrétní SDK, ale ani jeden nad druhým nijak nepřevažuje [27].

Možností, která nepochází od ani jednoho z tvůrců hlavních mobilních platforem je například **Vuforia**[8]. Jedná se o platformu pro tvorbu AR aplikací zaměřující se primárně na rozpoznávání a následné akce spojené s tím. Schopnosti detekce nekončí u specializovaných markerů či obrázků, Vuforia umožňuje detekci jednoduchých 3D primitiv, textu, či větších částí prostředí. Pomocí Vuforia Object Scanneru je možné provést naskenování 3D objektu a poté jej používat pro detekci. Detekce může probíhat ať už za pomoci lokální databáze tak i přes proprietární cloud. Tato platforma je sice nabízena jako doplněk pro Unity zdarma, nicméně je to podmíněno zobrazením vodoznaku Vuforie a je zde limitován počet rozpoznávaných markerů a cloudových rozpoznání. Pro odstranění těchto limitací je nutné přejít na placenou verzi.

Další možností pro rozšířenou realitu společně s Unity enginem je platforma **AR Foundation**[9], která je přímo od tvůrců Unity. AR foundation nám poskytuje jednotné API nad ARCore a ARkit a umožňuje nám tak vyvíjet aplikace mnohem jednodušeji. Díky tomu že se jedná o jedno API nad dvěma SDK je jasné, že budou dostupné pouze funkce, které jsou dostupné v obou zmíněných SDK. Rozsahem je tedy AR Foundation menší, nicméně podstatné funkce nám poskytuje a pro tvorbu většiny aplikací bude dostačovat. Vzhledem k tomu že se jedná o součást Unity enginu platí pro něj úplně stejné podmínky jako pro využívání enginu samotného.

Jestliže jsem zmiňoval React Native jako jeden z nejrozšířenějších frameworků bylo by vhodné uvést možnosti pro AR v něm. Pokud se nechce vývojář omezit pouze na jednu platformu, či použít specifický kód pro specifickou platformu zbývá jediná možnost v podobě **ViroReact**[10]. Jde opět o open-source platformu pro vývoj jak rozšířené, tak i virtuální reality. ViroReact přináší do React Native nové komponenty pro vykreslování virtuálních scén či objektů v okolním prostředí a velké množství dalších podpůrných komponent. Velkou výhodou je dostupnost testovací aplikace umožňující jednoduché spuštění vyvíjené aplikace na skutečném zařízení bez nutného sestavování aplikace před každým spuštěním. Vývojáři připravili také několik ukázkových aplikací společně s několika tutoriály.

Pro React Native dále existují i moduly přidávající podporu ARKitu i ARCore, nicméně tyto moduly jsou v experimentálním stádiu a nepodporují veškeré funkce samotných SDK.

[8] `https://developer.vuforia.com/`
[9] `https://unity.com/unity/features/arfoundation`
[10] `https://viromedia.com/viroar`

2.2.5 Zvolené technologie

Ve volbě technologií, které budou zastřešovat tuto aplikaci, mi byla ze strany zadání ponechána víceméně volná ruka a pouze mi byla dodána následná doporučení:

- multiplatformnost výsledné aplikace,
- využití open source technologií v co největší míře.

Za sebe jsem si k tomu přidal následující:

- technologie je aktivně vyvíjená,
- rozsáhlá dokumentace k dispozici,
- existence počátečních tutoriálů.

Aktivitu vyvíjeného projektu jsem zvolil po předchozích vývojových zkušenostech, kdy lze narazit na problémy které člověk není schopen sám vyřešit, a tak je vhodné mít možnost zeptat se buď přímo samotných vývojářů projektu nebo alespoň někoho z komunity uživatelů. S tímto bodem se pojí i požadavek na dokumentaci, která pokud existuje v kvalitní podobě tak ulehčuje vývoj ve velké míře. Poslední bod souvisí s absencí zkušeností s mobilním vývojem i se samotným vývojem AR aplikací. Shrnutí všech požadavků a jejich následné vyhodnocení je v tabulce 2.2.

Požadavek	ReactNative	NativeScript	Flutter	Xamarin	Unity
Multiplatformnost	Ano	Ano	Ano	Ano	Ano
Open source	Ano	Ano	Ano	Ne	Ne
Aktivní vývoj	Ano	Ano	Ano	Ano	Ano
Dokumentace	Dostatečná	V menší míře	Dostatečná	Dostatečná	Dostatečná
Tutoriály	Dostatečné	V menší míře	V menší míře	Dostatečné	Dostatečné
Jazyk	JS	JS	Dart	C#	C#

Tabulka 2.2: Shrnutí požadavků na zastřešující technologii

Následný výběr SDK už záleží na zvolené technologii a v případě více možností na základě požadavků, které určuje samotný rozsah aplikace. Po zběžné analýze jsem došel k tomuto seznamu:

- detekce tzv. Image Targetů,
- vykreslování grafických 3D primitiv (krychle, koule, křivka v prostoru),
- vykreslování 3D modelů ve volně dostupných formátech,
- detekce 3D objektů (např. krychle, koule).

Mnou přidané požadavky jsou stejné jako v předchozím případě. Shrnutí a vyhodnocení požadavků je opět přehledně uvedeno v tabulce 2.3.

Detekce Image Targetu bude využita k vizualizaci pro uživatele, kteří nemají k dispozici globus. Pokud uživatel globusem disponuje využije se

detekce 3D objektu, v mém případě konkrétně koule. Vykreslení objektů a primitiv bude využito jak pro doplnění chybějícího globusu, tak hlavně pro vykreslení satelitů a lokací pozemních segmentů. Křivka v prostoru bude sloužit pro naznačení orbit jednotlivých satelitů.

Požadavek	**ARCore**	**ARKit**	**Vuforia**	**AR Found.**	**Viro React**
Multiplat.	Ano	Ne	Ano	Ano	Ano
Open Source	Ano	Ne	Ne	Ne	Ano
Aktivní	Ano	Ano	Ano	Ano	Ano
Dokumentace	⟵ Dostatečná ⟶				
Tutoriály	Ano	Ano	Ano	Méně	Méně
ImageTarget	Ano	Ano	Ano	Ano	Ano
3D primitiva	Ano	Ano	Ano	Ano	Ano
3D formáty	⟵ obj, glTF, FBX ⟶				
Detekce 3D	Ne	Ano	Ano	Ne	Pouze iOS
Frameworky	Unity, Xamarin, Flutter	Všechny zmíněné	Unity	Unity	React Native

Tabulka 2.3: Shrnutí požadavků na AR SDK

Výběr samotné technologie se díky požadavku na open source zúžil na React Native, NativeScript a Flutter. Po spojení s možnostmi vývoje AR aplikací vychází, že NativeScript prozatím nepodporuje ARCore a tedy celou platformu založenou na operačním systému Android. Zbývá ReactNative a Flutter. V době tvorby této práce existuje podpora ARCore pro Flutter pouze v omezené verzi, kde není možné vykreslovat 3D objekty a žádná jiná multiplatformní prozatím neexistuje. Jedinou zbývající možností, která je právě naší zvolenou, je **React Native** společně s **Viro React**.

2.3 React Native

Jak bylo zmíněno React Native je open source framework tvořený Facebookem a komunitou umožňující tvorbu multiplatformních aplikací konkrétně pro platformy Android, iOS, web a UWP. Za počátek tohoto frameworku lze považovat prohlášení Marka Zuckerberga z roku 2012 ve kterém zmiňuje, že největší chyba jakou Facebook udělal bylo přílišné spoléhání na HTML oproti nativním mobilním aplikacím. Vývojářům slíbil brzké oznámení nové

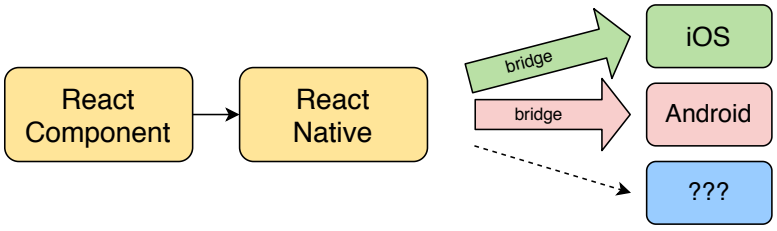
technologické možnosti pro vývoj na mobilních zařízeních. Nicméně vývoj trval několik měsíců a první zmínka o React Native se objevila až v roce 2015 na konferenci React.js, kde byl zmíněn v panelu *A Deep Dive into React Native* [13]. Projekt se stal velmi úspěšným, což lze dokázat právě i angažovaností komunity v jeho tvorbě. V posledních letech se pravidelně objevoval v Top 10 open source projektech na GitHubu a tak je tomu i v letošním roce [39].

2.3.1 Princip fungování

Hlavním principem stojícím na pozadí fungování React Native je Virtuální DOM. Pro pochopení se podíváme do světa prohlížečů, ve kterém je využívána část React Native, knihovna React.js.

Virtuální DOM je článek mezi vývojářovým popisem vzhledu aplikace a výslednou interpretací na cílovém zařízení. Pro změnu vzhledu aplikace je potřeba upravit DOM prohlížeče. Nicméně měnit pokaždé celý DOM je výpočetně náročné, a proto se React stará o porovnání Virtuálního DOMu ve své paměti s DOMem prohlížeče a provede pouze takové změny, které jsou nezbytně nutné k zajištění shody jejich vzhledu.

Provádění pouze nezbytných změn má za následek snížení výpočetní náročnosti překreslení aplikace. Navíc ještě přidává abstraktní vrstvu mezi vývojářův popis a výsledný vzhled aplikace. Této abstrakce můžeme využít, a právě to dělá React Native. Místo vykreslování do prohlížeče, jako je tomu v případě React.js, je využito tzv. mostu (bridge, obrázek 2.7), který propojuje programátorův popis aplikace přímo s nativními prvky cílové platformy [46]. Ve výsledku se tedy prvky aplikace zapsané pomocí frameworku zobrazují jako nativní prvky platformy.



Obrázek 2.7: Schéma převodu na nativní komponenty

O zobrazení prvků u kterých není možnost převedení na nativní prvky se stará virtuální stroj využívající JavaScriptCore. V případě iOS zařízení se jedná o interpret z prohlížeče Safari. U ostatních platforem, které nemají prohlížeč Safari je JavaScriptCore přidán do aplikace při kompilaci aplikačního balíčku.

2.3.2 JSX

Při vývoji webových aplikací je zvykem rozdělovat soubory dle využité technologie a tyto soubory následně nalinkovat. Příkladem může být klasická webová trojice HTML + CSS + JS. V .css souborech jsou styly pro jednotlivé prvky, v .js souborech je přidaná logika v JavaScriptu a nakonec .html soubory, které obsahují jednotlivé prvky značkovacího jazyka doplněné o nalinkovaný vzhled a logiku z předešlých souborů. U React Native je tento přístup odlišný, soubory dělíme dle jejich účelu.

Právě v tomto nám pomáhá deklarativní značkovací jazyk JSX[11] (JavaScript XML). Tento jazyk nám umožňuje zápis HTML elementů a jejich vykreslení na DOM přímo v JavaScriptovém souboru bez nutnosti využívání funkcí typu *createElement()*, *appendChild()*. Pro názornou ukázku se můžeme podívat na část kódu v obrázku 2.8, která vytvoří jednoduchý nadpis v React Native nejprve za pomoci JSX a poté bez jeho využití.

```
const e1 = <h1>JSX is the best!</h1>;
const e2 = React.createElement('h1', {}, 'JSX is useless!');
```

Obrázek 2.8: Nadpis za využití JSX a bez

Další vlastností JSX je možnost vložení JavaScriptových výrazů do kódu pomocí uzavření do složených závorek. Výrazem může být proměnná, vlastnost či jakýkoliv jiný JavaScriptový výraz. Právě toho je využíváno u komponent s dynamickým obsahem, který je do statické části komponentu dosazen právě pomocí výrazu. Opět se podíváme na ukázku v obrázku 2.9, kde máme příklad jednoduchého pozdravu do kterého je jméno dosazeno v závislosti na osobě která zdraví.

```
const greeting = <div>Helo, I~am {person.name}</div>;
```

Obrázek 2.9: Vložení výrazu do JSX

Z důležitých vlastností je ještě možno zmínit potřebu mít právě jeden rodičovský element a také požadavek na uzavření veškerých elementů, který vyplývá právě ze základu v podobě XML.

2.3.3 Komponenty

Základním stavebním prvkem v React Native, potažmo i v React.js jsou tzv. komponenty. Komponenty umožňují rozdělit UI do nezávislých, znovupoužitelných prvků. Koncept komponent je v podstatě stejný jako koncept funkcí v JavaScriptu. Přijmou nějaké vstupní hodnoty a na výstupu vracejí React elementy, popisující vzhled komponentu na cílovém zařízení.

[11] `https://reactjs.org/docs/introducing-jsx.html`

V zásadě existují dva způsoby, jak tvořit komponenty. Nejjednodušší možností je obyčejná JavaScriptová funkce. Druhou možností je využití ES6 třídy. Oba tyto zápisy jsou z pohledu Reactu ekvivalentními, nicméně využití ES6 třídy je robustnější a je také preferováno i později v dokumentaci. Porovnání zápisů jednoduchého komponentu lze vidět na obrázku 2.10.

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
}
class Welcome extends React.Component {
    render() {
        return <h1>Hello, {this.props.name}</h1>;
    }
}
```

Obrázek 2.10: Definice komponentu pomocí funkce a ES6 třídy

Hlavní funkcí každé komponenty je metoda *render()*. Tato metoda nám umožňuje využívat ve vzhledu aplikace nejen základní DOM elementy, ale právě vlastní komponenty, které shlukují základní elementy a jejich stylováním tvoří nové, složitější elementy. Pokud React „spatří“ komponentu v kódu přepošle ji její JSX atributy označované jako props. Přes klíčové slovo props jsou v komponentu také přístupné. Tyto atributy využíváme právě k univerzalizaci komponent umožňující jejich znovupoužití. Komponenta na obrázku 2.11 níže (v návaznosti na předchozí ukázky) by tedy po vykreslení vypadala jako nadpis „Hello Bob!“.

```
const element = <Welcome name="Bob" />;
```

Obrázek 2.11: Ukázka použití vlastní komponenty

Každá komponenta musí dodržovat pravidlo které říká, že v žádném případě nesmí měnit své atributy (vlastnosti, props) a musí k nim přistupovat pouze v režimu pro čtení. Pro primitivní komponenty to je v pořádku, ale u složitějších by bylo záhodno, aby si komponenta dokázala udržet svůj vnitřní stav (state).

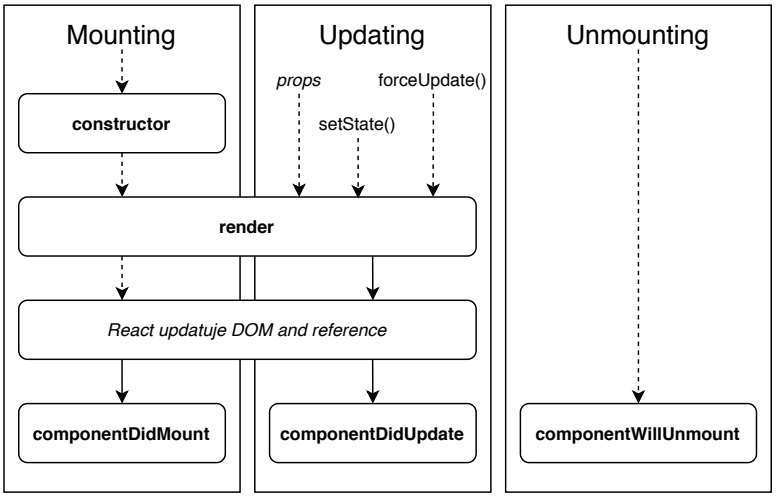
Využívání stavu komponent je klíčovou funkcí díky tomu, že komponent provede překreslení právě po změně svého stavu. O toto překreslení se stará React sám a vývojářovým úkolem je pouze změna stavu v závislosti na požadavcích. Ke stavu přistupujeme přes klíčové slovo state, nicméně stav nikdy nesmí být modifikován na přímo, ale přes speciální metodu *setState()*. Existuje výjimka kdy je možné ke stavu přistupovat na přímo a tou je počáteční nastavení stavu v konstruktoru.

```
this.setState({
    date: new Date()
});
```

Obrázek 2.12: Ukázka volání *setState()* metody

Využití této funkce má výhody v podobě dávkového zpracování, které se kladně podepisuje na výkonu a také tzv. status merge což se dá přeložit jako sloučení stavu. Znamená to, že můžeme nastavit pouze jednu konkrétní proměnnou a změní se pouze její stav, ostatní proměnné stavu zůstanou nezměněny. Možné volání je vidět na obrázku 2.12. V tomto případě se provede přiřazení nového data do stavové proměnné *date* a všechny ostatní proměnné zůstanou nezměněny.

Poslední důležitou částí komponent, kterou zmíním je životní cyklus komponenty a s tím spojené tzv. lifecycle metody. Průběh životního cyklu je vidět na obrázku 2.13. K těmto metodám se přistupuje pomocí předem definovaných názvů uvozených podtržítkem a právě ty jsou dostupné v každé komponentě. S jejich využitím lze provádět nastavování vlastností komponent, reagovat na změny atributů a stavu a provést „úklid“ při odebírání komponent.



Obrázek 2.13: Životní cyklus React komponentů

2.3.4 Využívání modulů

Při vývoji není nutné pokaždé vynalézat kolo a je možné si práci usnadnit díky komunitě vývojářů a jimi tvořenými knihovnami[12]. Knihovnou se rozumí seskupení funkcí, tříd, v našem případě i komponent, které byly vývojářem vytvořeny a uvolněny pro použití ostatním. Například se může jednat o komponentu která zajišťuje speciální druh Select Boxu, třídu funkcí umožňující přístup k nějakému API nebo třeba kolekce matematických funkcí.

Jednotlivé knihovny můžeme spravovat ručně, v případě většího množství se již jedná o nezanedbatelné množství práce a vhod přijde software, který se o knihovny bude starat sám. Jedná se o takzvaného správce balíčků (anglicky Package Manager). Tento specializovaný software se stará o instalaci,

[12] V JavaScriptové terminologii se knihovnou rozumí modul a naopak.

aktualizování, pomáhá řešit závislosti mezi balíčky a mnoho dalšího. Pro JavaScript, potažmo React Native, jde například o npm, yarn, rnpm [17].

Správa jednotlivých balíčků probíhá na základě tzv. závislostí. Tyto závislosti bývají udržovány v samostatném souboru, pro npm i yarn se jedná o soubor *packages.json* v kořeni projektu. V tomto souboru je vždy uveden název knihovny a její verze, která je v projektu požadována. Instalační skript poté projde tento seznam a jednotlivé knihovny nainstaluje. To přináší výhodu při práci v týmu, kdy nemusíme zálohovat veškeré použité knihovny na týmové úložiště. Pouze uvedeme názvy s potřebnými verzemi a každý člen týmu si pomocí správce balíčků už veškeré knihovny doinstaluje.

2.3.5 Vývoj

Vývojář má v zásadě dvě cesty, kterými se může v případě vývoje s React Native vydat. První a jednodušší, je vývojové prostředí Expo[13]. Druhou, určenou pro zkušenější vývojáře, je React Native CLI[14].

Název Expo ve skutečnosti označuje dvě různé aplikace. Jednou z nich je eponymní aplikace, kterou je možné stáhnout zdarma z obchodu s aplikacemi na příslušné platformě. Díky této aplikaci je možné spustit projekt na cílovém zařízení pouze zadáním adresy či skenem QR kódu. Druhou částí je sada nástrojů pro příkazový řádek, které umožňují inicializaci a spouštění vývojového serveru. Tento server slouží ke kompilaci aplikace a komunikaci s testovací aplikací. Výhodou této možnosti je tedy její jednoduchost. Nicméně tato jednoduchost je i kamenem úrazu této možnosti. Expo totiž nepodporuje úplně všechny nativní komponenty a v případě některých knihoven tedy tato možnost není použitelná.

Knihovnou která nepodporuje Expo je například zvolený framework pro rozšířenou realitu Viro React.

V našem případě je tedy nutné využít možnosti druhé, a to instalace vývojových prostředí (XCode pro iOS a Android Studio pro Android) a následné konfigurace React Native. Tímto procesem však provádí tutoriál dostupný z dokumentace, měl by ho tedy zvládnout i začátečník [36]. Přestože je tato možnost robustnější a umožňuje vývojáři více možností konfigurace nese se s ní i jedna podstatná nevýhoda. Díky potřebě vývojového prostředí není možné kompilovat verze pro iOS na operačním systému Windows, kde není XCode dostupné. Je tedy potřeba mít k dispozici dvě vývojové platformy.

[13] `https://expo.io/`
[14] CLI = Command Line Interface (Příkazový řádek)

2.4 Viro React

Druhou důležitou volbou bylo zvolení frameworku pro rozšířenou realitu. Na základě zmíněných rozhodnutí byl zvolen právě Viro React. Samotná technologie se nazývá pouze Viro, nicméně je poskytována ve dvou variantách. Jednou je framework Viro Core určený pro psaní nativních aplikací v Javě pro platformu Android. Druhou je právě Viro React, který je knihovnou pro React Native a podporuje rozšířenou realitu v podobě ARCore pro Android i ARkitu pro iOS. Viro React však není knihovnou pouze pro rozšířenou realitu. Podporuje také virtuální realitu pro Apple Cardboard, Android Cardboard, Daydream a Gear VR [44]. Původně Viro React nebyl opensource, ale od verze 2.17.0 se tomu tak stalo a zdrojové kódy jsou tedy dostupné komunitě pod MIT licencí.

Framework se skládá ze dvou hlavních částí. První je výkonný nativní 3D renderovací engine a druhou je značné množství React komponent, které jsou využívány jak pro vykreslování scén, tak i pro tvorbu jejich obsahu.

2.4.1 Vlastnosti

Zaměříme-li se pouze na vlastnosti související s rozšířenou realitou je hned v počátku důležité zmínit, že na pozadí jsou využívány SDK ARCore a ARKit. Pro podporu Viro React na cílovém zařízení je nutné, aby toto zařízení podporovalo právě zmíněné SDK [1, 16]. Na rozdíl od tvorby klasických 3D aplikací, nebo aplikací pro virtuální realitu je u AR aplikací požadována interakce se světem v okolí uživatele. Tato interakce probíhá právě skrze několik speciálních komponent a metod, které pomáhají vývojáři při tvorbě aplikací využívajících rozšířenou realitu v různých podobách.

Z komponent které jsou pro AR vývoj užitečné můžeme jmenovat například *ViroARScene*, jedná se o kontejner, který spravuje graf scény vykreslovaný nad reálným světem. Další často využívanou komponentou je *ViroARPlane*, která se stará o pozicování dalších komponent na základě detekovaných rovin. Důležitou roli v této práci zastává *ViroARImageMarker*. Tato komponenta umožňuje umisťovat objekty do scény relativně vůči detekovanému obrazu.

Z poskytovaných vlastností stojí za zmínku *6DOF*[15] *pohyb kamery*, jehož detekce má za následek, že objekty umístěné ve scéně zůstávají ukotvené na své lokaci. Toho by nebylo možné docílit bez možnosti mít jako pozadí AR scény obraz z kamery. To je označováno jako *video pass through*. Ze snímaného videa je také následně detekována světelná intenzita a teplota ambientního osvětlení, kterou je možné využít pro realistické osvětlení přidaných objektů.

[15] DOF = Degree Of Freedom, stupně volnosti

2.4.2 Vývoj

Tvůrci uvádí jako jednu z hlavních výhod vlastní testovací aplikaci. Tato aplikace je dostupná z AppStore i PlayStore pod názvem Viro Media. Výhodou této aplikace je, stejně jako při použití Expo (které ovšem Viro React nepodporuje), že vývojář nepotřebuje mít nainstalované prostředí Xcode či Android studio. Jediné co musí vývojář provést je spustit vývojový server na svém počítači a z aplikace se na něj odkázat přes síť. Další výhodou této aplikace je podpora tzv. Live Reloadingu při kterém dochází k automatickému načtení aplikace kdykoliv jsou změny v některém ze souborů projektu uloženy. Nevýhodou této testovací aplikace je nutnost specifického názvu aplikace. Testovací aplikace funguje pouze pokud vyvíjená aplikace má název „ViroSample“. Další podstatnou nevýhodou je nekompatibilita s některými dalšími knihovnami pro React Native.

Pokud není možné použít testovací aplikaci, lze postupovat standardní cestou s nainstalovaným vývojovým prostředím. V tomto případě stačí přidat Viro React jako závislost do React Native projektu a v závislosti na vyvíjené platformě projít několik dalších kroků doporučených vývojáři [15].

2.5 Souřadné systémy

V této a následující kapitole bych se rád zaměřil na úvod do problematiky výpočtu pozic satelitů. Chceme-li se bavit o pozici libovolného objektu, setkáme se s prvním problémem tohoto tématu a tím je existence rozličného množství souřadných systémů ve kterých lze pozici vyjádřit. Mezi formáty a jednotkami jednotlivých souřadných systémů mohou být značné rozdíly a proto je nutné si nejprve ujasnit jaké budeme používat. Tento výběr je potřeba uzpůsobit i s ohledem na to s jakými souřadnými systémy pracují matematické modely, které nám umožňují vypočítat polohu satelitu či jiného vesmírného tělesa v konkrétní čas na základě sady dat, která jednoznačně popisuje orbitu objektu.

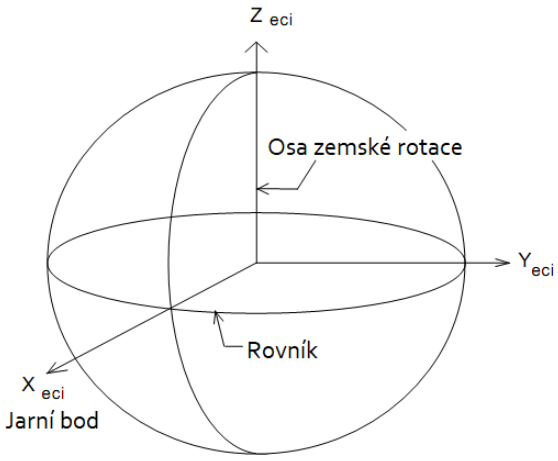
Souřadný systém (ekvivalentně soustava souřadnic) umožňuje jednoznačně vyjádřit polohu bodu v prostoru za pomocí čísel. Pro určení polohy v n-rozměrném prostoru je potřeba n čísel, které následně tvoří uspořádané n-tice jež označujeme jako souřadnice. Mezi souřadnými systémy je možno libovolně přecházet pomocí tzv. transformace souřadnic. Z matematického pohledu se jedná o bijekci, což znamená že jeden bod jednoho souřadného systému odpovídá právě jednomu bodu souřadného systému jiného.

Souřadných systémů existuje velké množství, a proto si popíšeme pouze některé. Konkrétně ty, které budou využity v této práci. Nicméně je důležité zmínit jednu poznámku. V této oblasti neexistuje dostatečná česká terminologie, takže se ji ani nebudu pokoušet zavádět a budu používat příslušné

anglické pojmy s patřičným českým vysvětlením. Veškeré informace pro tuto kapitolu byly čerpány z knihy *Global positioning systems, inertial navigation, and integration* [10].

2.5.1 Earth Centered Inertial

Earth Centered Inertial (ECI) souřadný systém je variantou kartézské soustavy souřadnic a sestává tedy ze tří na sebe kolmých os s počátkem v těžišti Země (obrázek 2.14). Rovina XY je rovina tvořená rovníkem. Osa Z je kolmá k této rovině a prochází oběma póly, odpovídá tedy ose zemské rotace. Osa Y prochází skrze tzv. vernal equinox, některými autory překládán jako jarní bod [12]. Slunce během pohybu po své dráze – ekliptice, protne rovníkovou rovinu právě dvakrát za jednu rotaci. Při cestě směrem "dolů" protíná rovinu na jaře a nastává den jarní rovnodennosti. O 180 stupňů dále nastává druhé protnutí a s tím i den podzimní rovnodennosti. Třetí osa, osa X je tvořena doplněním těchto dvou os na kartézskou soustavu souřadnic. Slovo inerciální v názvu označuje, že tato soustava souřadnic zůstává fixní vůči rotaci Země, tedy spolu se Zemí nerotuje kolem osy Z.



Obrázek 2.14: ECI souřadný systém

2.5.2 Earth-Centered, Earth-Fixed

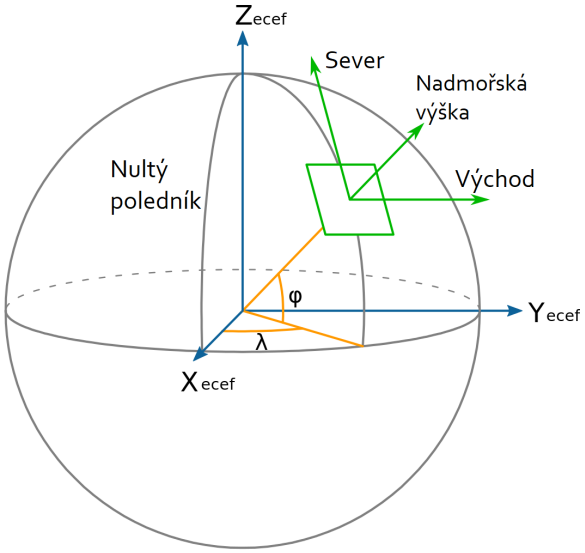
Souřadný systém *Earth-Centered, Earth-Fixed* (ECEF), někdy zmiňovaný jako geocentrický souřadný systém je další z rodiny kartézských souřadných systémů. Stejně jako ECI souřadný systém má počátek v zemském těžišti a rovina XY je určena rovníkem. Osa Z v tomto případě prochází skutečným severním pólem a neodpovídá tedy úplně přesně ose zemské rotace. Osa X prochází bodem, ve kterém se protíná nultý poledník s rovníkem. Osa Y je opět tvořena doplněním na kartézskou soustavu. Toto nadefinování má za následek rotaci souřadného systému spolu s rotací Země, což znamená že

fixní bod například na zemském povrchu má stále stejné souřadnice. Tomuto souřadnému systému odpovídají modré osy na obrázku 2.15.

2.5.3 Zeměpisné souřadnice

V zeměpisných souřadnicích je poloha bodu udávána pomocí tří údajů. Zeměpisné šířky, délky a nadmořské výšky. Zeměpisná šířka udává úhlovou vzdálenost od rovníku a pohybuje se v rozsahu 0-90 stupňů. Směr vzdálenosti je udáván v závislosti, jestli se jedná o severní nebo jižní polokouli. Zeměpisná délka udává úhlovou vzdálenost od nultého poledníku a pohybuje se v rozsahu 0-180 stupňů. Dále rozlišujeme západní a východní polokouli. Někdy jsou obě tato rozlišení nahrazována záporným znaménkem v jednom ze směrů. Na obrázku 2.15 je zeměpisná šířka označena jako φ a délka jako λ, zároveň jsou tyto hodnoty naznačeny ve směru ve kterém se běžně označují kladnými hodnotami. Pro jednoznačné určení místa na planetě by stačily tyto dva údaje pokud bychom řekli že daný bod leží na zemském povrchu, nicméně pro názornost se většinou udává třetí údaj, kterým je nadmořská výška, kterou bez jejího udání nemůžeme z předchozích dvou údajů spočítat.

Tyto souřadnice jsou využívány hlavně díky systému GPS, který je přístupný i široké veřejnosti a díky tomu umožňuje jednoznačně identifikovat místa na Zemi.

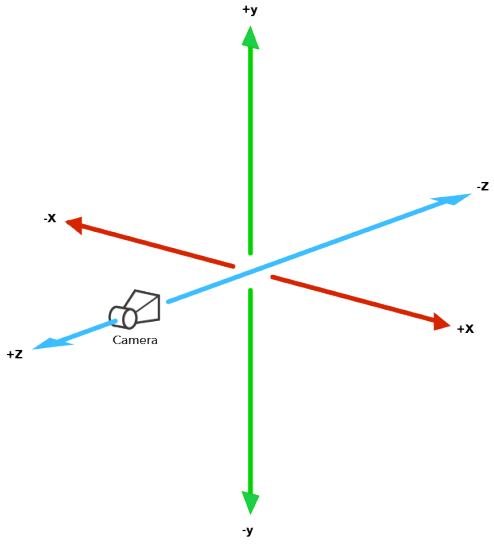


Obrázek 2.15: ECEF (modře), ENU (zeleně) a zeměpisný (žlutě) souřadný systém

2.5.4 Souřadnice ve Viro Scéně

Viro React používá pravotočivý kartézský souřadný systém, kde pohled kamery míří v záporném směru osy Z. Po prvotní detekci okolí je počátek

umístěn v pozici kamery a při pohybu scénou je tento počátek neměnný. Směry ostatních os lze vidět na obrázku 2.16.



Obrázek 2.16: Souřadnice ve Viro scéně

Pro komplexnější pozicování je vhodné využít graf scény, který můžeme tvořit pomocí speciálních komponent *ViroNode*. Tento komponent nemá žádnou viditelnou podobu, slouží pouze jako obal pro prostorové transformace vůči svému rodiči. Pomocí hierarchie těchto komponent docílíme konstrukce grafu scény.

2.6 Výpočty pozic satelitů

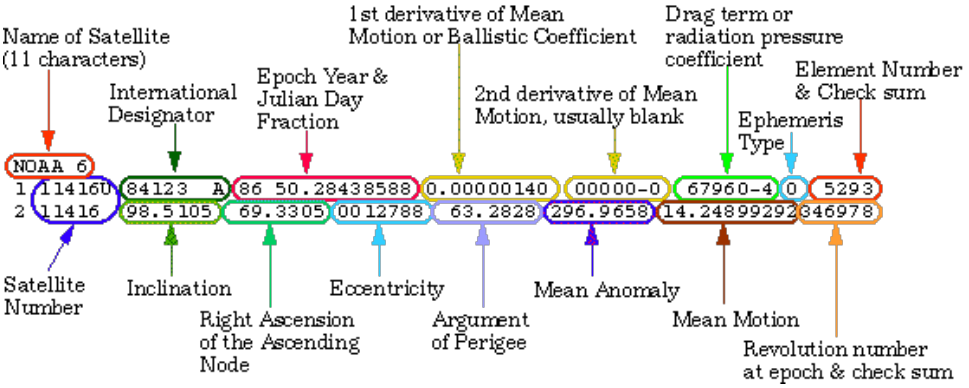
Vzhledem k neustálému pohybu a velké rychlosti satelitů, dochází každou chvílí k podstatné změně pozice. Z tohoto důvodu není možné poskytovat data o polohách satelitů přímo ve formátu jejich polohy, ale jsou poskytována data, která po dosazení do matematických modelů umožňují vypočítat pozici satelitu v daný časový okamžik. Tato data už nemusí být konstantně aktualizována, ale stačí aby k aktualizaci došlo v případě, že by odchylka, vypočítané polohy od té skutečné, přesáhla nějakou mez. Pro vzdálenější satelity, které jsou pomalejší stačí tato data aktualizovat jednou za několik dní, pro rychlejší či důležitější se může jednat o několik aktualizací denně [18].

2.6.1 Dvouřádkové elementy dráhy

Dvouřádkový element dráhy (Two Line Element set, TLE) je textový formát reprezentující elementy dráhy a další údaje, které jsou potřebné k výpočtu pozice kosmického tělesa právě ve dvou řádcích. Tyto elementy dráhy jsou

souborem minimálně šesti veličin a časového údaje, které společně jednoznačně definují dráhu kosmického tělesa v daný časový okamžik. Popis jednotlivých položek je vidět na obrázku 2.17, překlad není uváděn protože pro některé prvky není ustálený český překlad. Reprezentace těchto dat je specifická pro takzvané Simplified Perturbation matematické modely (viz kapitola 2.6.3).

Formát TLE dat vychází z historicky používaných děrných štítků. Z těchto děrných štítků si formát zachovává pevné délky jednotlivých záznamů – případná čísla musí být doplněna uvozujícími mezerami na příslušnou délku, ale narozdíl od nich nezahrnuje informace, které se dají z těch ostatních dopočítat. Jedná se tedy o opatření kvůli úspoře dat při případném přenosu po síti či jejich ukládání.



Obrázek 2.17: Popis formátu TLE

2.6.2 Poskytovatelé TLE

Podklady pro data jsou sbírána z United States Space Surveillance Network (SSN) pod správou Vesmírných sil Spojených států amerických. Tato síť zaznamenává objekty kroužící kolem Země a objekty které získají dostatečnou věrohodnost pro publikaci uveřejní v katalogu satelitů SATCAT. Nízká věrohodnost objektu může být dána například spojením objektu se skupinou objektů v jeho blízkém okolí nebo nemožností spárovat objekt s některým ze známých vesmírných startů. V době psaní práce se v tomto katalogu nachází zhruba 45000 objektů [33].

Pro přístup k datům v katalogu byla zřízena webová službu space-track.org na které uživatel po registraci dostane přístup k REST API poskytující jak přístup ke katalogu SATCAT, tak právě i ke zmíněným záznamům TLE. Využití tohoto API je limitované pouze počtem dotazů, kdy strop je 20 za minutu a 200 za hodinu.

Další možností kde získat TLE záznamy je webová stránka CelesTrak. Na této stránce jsou záznamy tříděny do přehledných kategorií jako například vesmírné mise, vědecké satelity, GPS a mnoho dalších. Zároveň poskytuje

i rychlý přístup do katalogu SATCAT a také interaktivní vizualizaci. Tato stránka ovšem nenabízí žádné API, pouze poskytuje tyto záznamy jako textové soubory.

2.6.3 Simplified Perturbation modely

Výpočty související s pozicemi satelitů tedy využívají data ve zmíněném formátu TLE. Z těchto dat je možné spočítat pozici směrem do budoucna i do historie od okamžiku zveřejnění dat. S větší časovou odchylkou však klesá přesnost výpočtů díky vnějším silám jako je tlak kosmického záření, odpor zbytků atmosféry nebo i díky vnitřním silám v podobě manévrů samotných satelitů.

Výpočty probíhají s využitím zjednodušených matematických modelů tzv. Simplified Perturbation Models. Jedná se o set pěti konkrétních modelů SGP, SGP4, SDP4, SGP8 a SDP8 (SGP označuje Simplified General Perturbation – vhodné pro objekty s periodou menší než 225 minut, SDP značí Simplified Deep Space Perturbations – pro periodu nad 225 minut) používaných pro výpočet tzv. orbitálních stavových vektorů. Jde o vektory pozice a rychlosti, které společně s časem (epochou) jednoznačně identifikují trajektorii obíhajícího tělesa [24].

Nejpoužívanějším modelem je SGP4. Ten se vyznačuje chybovostí maximálně okolo 1 km ve výchozí epoše a roste rychlostí zhruba 1-3 km za den [43]. Tento model bude využit i v této práci díky implementaci v knihovně satellite.js

2.7 Podobné aplikace

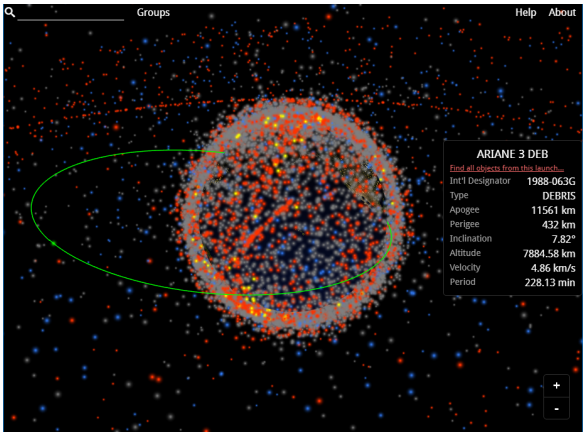
Před samotným vývojem aplikace jsem provedl analýzu již dostupných řešení. Existuje několik aplikací vizualizujících orbity satelitů v prohlížeči i na mobilních zařízeních. Tato vizualizace probíhá nad 3D modelem Země, popřípadě nad její mapou. Dále je dostupných několik aplikací, které pro vizualizaci využívají rozšířenou realitu. V případě satelitů se však jedná o vizualizaci stylem zobrazení toho, co je možné vidět na obloze a nejedná se o rozšíření globusu jako v případě této práce. Poslední kategorií jsou právě aplikace, které nějakým způsobem rozšiřují globus pomocí prvků rozšířené reality.

2.7.1 Vizualizační aplikace

Vizualizačních aplikací lze nalézt mnohem více než těch využívajících rozšířenou realitu, zmíním tedy pouze několik vybraných. Zejména ty které jsem

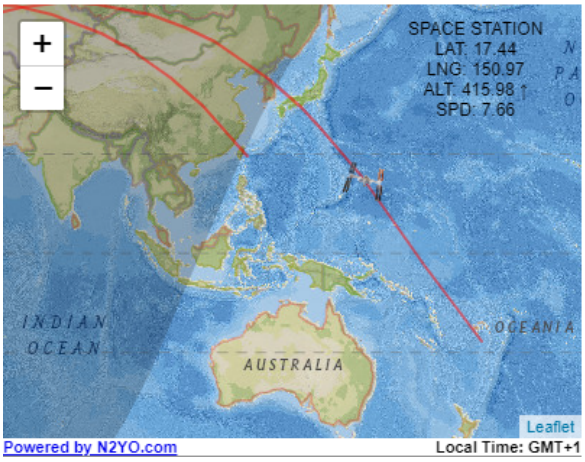
používal v průběhu vývoje pro kontrolu vizualizačních schopností, nebo pro inspiraci jaké informace uživateli zobrazit.

Na prvním místě se jedná o webovou aplikaci stuffin.space[16]. Jedná se o vizualizaci veškerých těles z katalogu SATCAT nad virtuálním globusem v reálném čase (obrázek 2.18). Spolu se schematickým znázorněním těchto objektů je možné zobrazit si orbitu jednotlivých těles a výpis základních informací. V základu jsou objekty zobrazeny všechny, ale vizualizaci je možné pomocí filtrů omezit na několik vybraných skupin. Další stránkou pracu-



Obrázek 2.18: Služba stuffin.space

jící na podobném principu je web N2YO.com[17]. V tomto případě se jedná o vizualizaci nad mapou Země (obrázek 2.19). Vizualizační schopnosti jsou však nahrazeny mnohem větším počtem zobrazovaných informací včetně predikce přeletů nad konkrétním místem. Tato stránka také poskytuje API pro získávání TLE dat, nicméně jedná se pouze o replikaci dat ze space-track.org.



Obrázek 2.19: Vizualizace ze služby N2YO.com

[16] `http://stuffin.space/`
[17] `https://www.n2yo.com/`

Pokud jsem se v kapitole 2.6.2 zmínil o stránce celestrak.com jako o zdroji dat, je nutno zmínit i jejich vizualizaci. Jde o vizualizaci nad 3D modelem Země společně se základními informacemi. Na rozdíl od aplikace stuffin.space je zde možné pracovat s rychlostí času a posouvat se tak do konkrétních okamžiků. Bohužel z uživatelského hlediska mi nepřišla tak přívětivá jako ostatní.

Jednou z mobilních aplikací je například Orbitrack[18]. Jedná se o aplikaci od tvůrců N2YO.com. Na rozdíl od jejich webové služby je tato aplikace placená a má možnost vizualizovat satelity nad 3D modelem Země. Stejně jako na webové službě je zde velké množství informací a například i modelů jednotlivých satelitů. Jako bonusem navíc je zde například mapa noční oblohy, která využívá „rozšířené reality“ ve smyslu využití údajů z akcelerometrů pro rozpoznání natočení telefonu. Ukázka je na obrázku 2.20.

Obrázek 2.20: Aplikace Orbitrack [29]

2.7.2 Aplikace s využitím AR

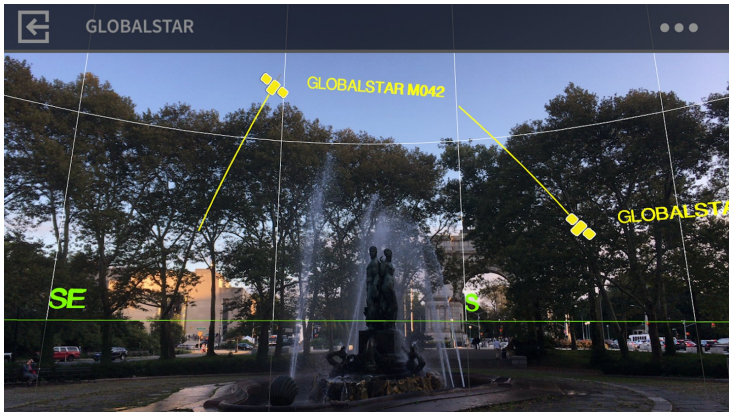
Aplikací, které využívají rozšířenou realitu, už lze nalézt méně, nicméně v omezené míře existují jak na platformě Android, tak na iOS. V placené i bezplatné verzi. Žádná z aplikací však nefunguje na principu stejném, nebo alespoň vzdáleně podobném cíli této práce, tedy vizualizace satelitů v blízkém okolí uživatele na základě nějakého reálného objektu.

Jednou z aplikací která využívá rozšířenou realitu v pravém slova smyslu, rozšiřuje tedy obraz z kamery je například Satellite AR (viz obrázek 2.21). V této aplikaci se do snímaného obrazu zobrazují polohy zvolených satelitů a symbolické naznačení jejich orbit.

Na stejném principu pracují i ostatní aplikace sledující satelity. Navíc jde

[18] `http://southernstars.com/products/`

většinou o aplikace specializující se na vyhledávání lokalit satelitů pro satelitní televizi, tudíž je jejich vizualizační rozsah značně omezen na tuto oblast. Často se také jedná o aplikace vizualizující rozpoložení hvězd na obloze.



Obrázek 2.21: Aplikace Satellite AR [34]

Aplikací rozšiřujících globus je naprosté minimum. Narazil jsem pouze na jedinou a tou je aplikace pro smart hračku AR Globe od společnosti Neobear. Tato hračka spočívá v kombinaci speciálního globusu a aplikace, která je naučena rozpoznávat obrázky na povrchu globusu. Nad těmito předem definovanými místy zobrazí například doplňující 3D model, nebo nějaké dodatečné informace viz obrázek 2.22. Vzhledem k tomu že se jedná o komerční produkt, nejsou nikde dostupné kódy ani informace o samotném principu fungování aplikace.

Obrázek 2.22: Smart hračka Neobear AR Globe [31]

Kapitola 3

Návrh

V této kapitole bych se rád zaměřil na návrh aplikace před začátkem samotného vývoje. Cílem je specifikace cílové uživatelské skupiny vyvíjené aplikace a také specifikace požadovaných funkcí. V návaznosti na tyto požadavky pak proběhne návrh uživatelského rozhraní a rozčlenění do jednotlivých React komponent.

3.1 Cílová skupina

Díky následování principů User-Centered Designu je důležité analyzovat skupinu uživatelů, která bude využívat vyvíjenou aplikaci. Na základě definice cílové skupiny bude lehčí přiblížit jejich požadavky na tuto aplikaci a v závislosti na nich přizpůsobit například uživatelské rozhraní aplikace a úroveň odbornosti zobrazovaných informací.

Cílovou skupinou této aplikace budou převážně studenti základních a středních škol, kde se bude aplikace dát využít jako podpora výuky například přírodopisu či fyziky. Ve všech případech se bude jednat o doplnění probíraných témat souvisejících s vesmírem a lidského působení v něm.

Pokud jsou cílovou skupinou mladší studenti je také vhodné počítat s tím, že jim budou při práci například na domácích úkolech asistovat jejich rodiče, není tedy vhodné aplikaci tvořit s přílišným zaměřením na nejmladší cílovou skupinu.

Další početnou skupinou může být mnoho zájemců o vesmírné technologie. Na rozdíl od mladších dětí tuto skupinu zajímá větší množství informací a nemíří do aplikace pouze za účelem podívání se na jakýsi obecný přehled. Těmto uživatelům může aplikace pomoci například při pozorování přeletů satelitů, je tedy vhodné mít nejenom pro ně připravené potřebné informace.

3.2 Požadované funkce

Další částí která zásadně ovlivní návrh výsledné aplikace jsou požadavky na její funkčnost. V této fázi stačí brát v potaz pouze funkčnost ze strany uživatele. Vývojářské požadavky totiž vyplynou z nich a budu se jimi více zabývat v kapitole ohledně realizace samotné aplikace.

3.2.1 Vizualizace satelitů

Hlavní funkcí má být vizualizace satelitů a bude to také prioritizováno ze strany uživatele. Aplikace by měla být schopná zobrazit libovolné satelity z katalogu SATCAT. Pro ulehčení využívání by měl být k dispozici výběr zobrazeného satelitu na základě rozdělení satelitů do kategorií, například na základě vesmírné mise, projektu a podobně. Ne každý satelit spadá do speciální skupiny, musí být tedy možnost zvolit i tyto satelity. Pro vizualizaci oběhu je žádoucí, aby bylo možné vykreslit orbity jednotlivých satelitů. Samotná vizualizace bude probíhat nad globusem který uživatel bude mít na stole před sebou a bude se tedy řídit jeho rotací a pozicí. Pokud uživatel nemá k dispozici globus bylo by vhodné mít připravenou variantu vizualizace, která se bez globusu obejde.

Do tohoto bodu by se dalo zařadit i zobrazení informací o satelitech. Tyto informace by měly být jednoduše dostupné a vysvětlené, aby i neznalému uživateli bylo jasné co který údaj znamená.

3.2.2 Vizualizace pozemního segmentu

S vizualizací pozemního segmentu je to podobné jako se satelity. U tohoto bodu je však ještě více důležitá přesnost vizualizace. Vzhledem k tomu že se segmenty nacházejí na konkrétních lokacích, měl by uživatel být schopný z vizualizace tuto lokaci rozpoznat. Pro výběr zobrazených částí je opět vhodné mít připravený seznam jednotlivých prvků roztříděný do přehledných kategorií a toto rozdělení do kategorií reflektovat i ve vizualizaci.

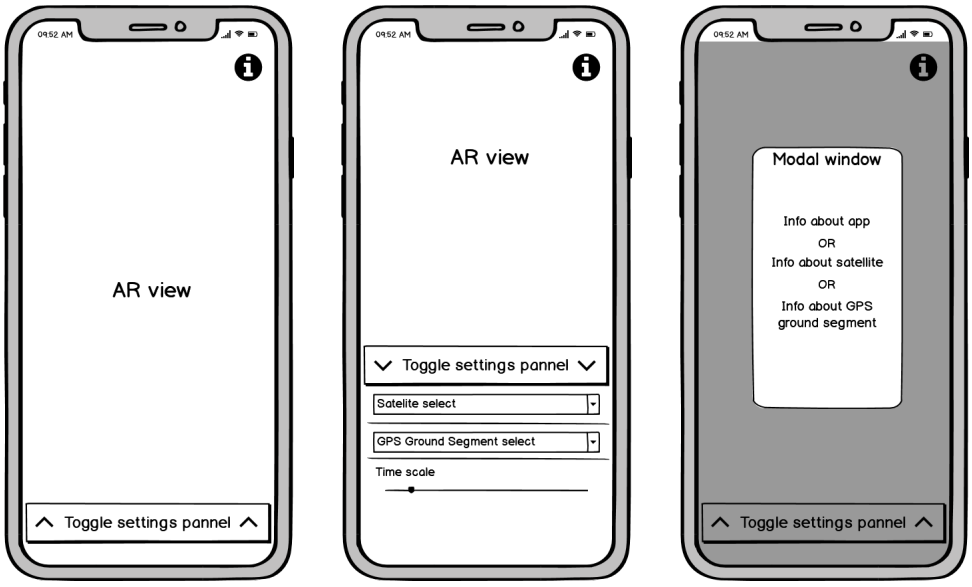
3.2.3 Detekce pozice a rotace cíle vizualizace

Jak již bylo zmíněno cíle vizualizace by měly být dvou typů. Jedním typem bude globus, nad kterým se po detekci budou pohybovat zvolené satelity a na jeho povrchu budou naznačeny prvky pozemního segmentu GPS. Další možností je vizualizace bez globusu. Půjde tedy například o vytištěný obrázek pohledu na Zemi z horního pohledu. Po detekci se zobrazí virtuální globus který bude řízen právě vytištěným obrázkem.

U těchto cílů musí být možnost detekovat jejich pozici a rotaci v prostoru. Na základě detekovaných dat bude upravována vizualizace. Jinými slovy bude možné otáčet globusem, či obrázkem pro rotaci virtuálního globusu a celé vizualizace.

3.3 Uživatelské rozhraní

Vzhledem k cílovým skupinám a požadovaným funkcím bylo zvoleno spíše jednoduché a funkční rozhraní než těžkopádné s mnoha okny. Veškerá práce se bude odehrávat v jednom hlavním okně, které bude mít jako pozadí právě záběr z kamery. Tuto oblast následně bude překrývat buď vysouvací panel s výběrem zobrazovaných prvků nebo modální okna s informacemi a nastavením.



(a) : Domovská obrazovka **(b) :** Modální okno **(c) :** Panel nastavení

Obrázek 3.1: Návrh uživatelského rozhraní

Na obrázku 3.1a je možné vidět návrh zmíněného hlavního okna aplikace. Většinu plochy tvoří pohled z kamery, který bude případně doplněn prvky rozšířené reality. V horním levém rohu je tlačítko, pod kterým se budou skrývat informace o aplikaci a návod k použití, aby byl uživateli přístupný co nejjednodušeji. Ve spodní části obrazovky je vidět záhlaví výsuvného panelu, které bude sloužit k jeho vysunutí.

Po vysunutí panelu s nastavením se dostáváme do situace na obrázku 3.1b. Jsou zde možnosti k výběru zobrazovaných prvků a zároveň i další možná nastavení. Elementy pro výběr prvků budou umožňovat jak výběr z předem připravených kategorií, tak i na základě ID satelitu. Dále bude možný výběr

pozemního segmentu tak aby bylo možné zobrazit opravdu všechny prvky které chceme vizualizovat.

Poslední možnou situací v aplikaci bude zobrazení modálního okna jako je tomu na obrázku 3.1c. V tomto případě modální okno překryje veškeré prvky, které byly do té doby na obrazovce zobrazeny. Obsah modálního okna se bude měnit dle situace ve které bylo toto okno zobrazeno. Může se jednat například o nápovědu k celé aplikaci, nebo okno s informacemi ohledně jednotlivých satelitů.

3.4 Koncepce aplikace

V této části bych chtěl popsat jak celá aplikace funguje - pro lepší pochopení návaznosti komponent a zároveň z jakých komponent se bude skládat. Následně komponenty které bude potřeba naprogramovat přiblížím podrobněji.

Základním prvkem celé vizualizace bude virtuální globus, který bude zobrazen po detekci cíle. Tento globus bude nahrazovat skutečný globus v případě planárního cíle, popřípadě bude objekt globusu pouze poskytovat zpětnou vazbu pro uživatele kde je skutečný globus detekován, pokud by detekce nebyla stoprocentně přesná. Virtuální globus bude na základě předaných ID satelitů spravovat jejich vykreslení včetně orbit. Vykreslení bude probíhat po získání dat z webového API za pomocí naprogramované knihovny. Zároveň bude virtuální globus řídit otevírání dalšího prvku v podobě informačního okna, které na základě uživatelova výběru zobrazí informace o konkrétním satelitu. Tyto prvky budou spolu s ostatními uzavřeny v jedné hlavní scéně. Zastřešující scéna se bude starat o správu zvolených dat ze select komponentů a o předání těchto dat do ostatních komponent.

3.4.1 Komponenty

Už úvodní tutoriál pro React Native zmiňuje že vše, co je vidět na obrazovce zařízení, je v nějakém smyslu komponentem. Krom toho podporuje a doporučuje tvorbu vlastních, které nám pomohou členit aplikaci do menších, přehlednějších a znovupoužitelných částí. Proto je rozvržení do komponent důležitou částí návrhu a budu se jím zabývat. Díky nulové zkušenosti s Reactem před touto prací může být toto dělení neoptimální, nicméně z mého pohledu dávalo smysl. Zaměřím se tedy pouze na prvky, které nejsou dostupné v samotném React Native a bude je potřeba vytvořit jako vlastní (ve smyslu vytvořené) komponenty.

Globe Jedná se o hlavní komponentu která bude zajišťovat většinovou část aplikace v podobě reprezentace globusu nad planárním nebo 3D cílem. Bude se starat o vizualizaci jednotlivých satelitů, jejich orbit a správu dat

potřebných ke správné vizualizaci. Na základě těchto dat bude následně provádět jejich vykreslení včetně orbit. Stejná situace nastává i v případě pozemního segmentu GPS.

Info modal Jedná se o rozšíření modálního okna které sice v React Native existuje, ale toto bude mít obsah závislý na zvoleném satelitu o němž bude třeba zobrazit informace. V ideálním případě by se mělo dát komponentu předat ID konkrétního satelitu a veškerá data si obstará samostatně, správně je naformátuje a vrátí modální okno se všemi potřebnými informacemi.

Slide panel Tento komponent bude pouhým kontejnerem na další prvky se specifickým chováním. Po kliknutí na jeho záhlaví vyjede do určité výše na obrazovce a po dalším kliknutí zajede. Spolu s ním se budou pohybovat i veškeré prvky v něm, a musí být tedy schopný pojmout libovolné množství obsahu například pomocí scrollovací oblasti.

Multi Category Select Box Odstranění zmíněného nedostatku základního select boxu v podobě výběru pouhého jednoho prvku bude vyřešeno právě tímto komponentem. Jeho schopností bude umožnění výběru libovolného množství prvků přehledně roztříděných v kategoriích. Zároveň zde musí být možnost, jak tyto zvolené prvky zobrazovat a spravovat je, zrušit jejich volbu. Pokud by uživatel vyhledal prvek, který se nenachází v žádné kategorii, mohla by zde být možnost tento prvek přidat a následně jej zvolit.

Kapitola 4

Implementace

Tato kapitola se zabývá popisem implementace vyvíjené aplikace. Budou zde popsány využité technologie pro programování, využité externí knihovny a také nastíněno řešení nejdůležitějších částí implementace. Aplikace byla vyvíjena s ohledem na principy User-Centered Designu, počítala tedy s návrhem zmíněným v předchozí kapitole a jeho iterativním vylepšováním na základě zpětné uživatelské vazby.

4.1 Použité nástroje

Jedním z hlavních programátorských nástrojů je bezesporu integrované vývojové prostředí (IDE). Pro samotný JavaScript použitý v této práci jich existuje velké množství, ze kterého si vývojář může vybrat to, které se mu osvědčilo nejvíce. Často doporučovaným je *WebStorm*[1] od české společnosti JetBrains s.r.o. Dle tvůrců samotných je označováno jako nejchytřejší JavaScript IDE. Zahrnuje spoustu funkcí jako například kvalitní debugger, podporu dalších vývojových nástrojů či integraci několika verzovacích systémů. Z mého pohledu se jednalo o příliš těžkopádné řešení a obrátil jsem se na open source IDE od Microsoftu, *Visual Studio Code*[2]. Jedná se o velmi oblíbené, dle některých anket dokonce nejpoužívanější vývojové prostředí [38]. Jeho výhoda spočívá jednak v open source MIT licenci, ale hlavně v jeho lehkosti ohledně dostupných funkcí. V základu editor přichází pouze s našeptávačem IntelliSense známého z ostatních Visual Studio produktů, debuggerem a podporou verzovacího systému Git. Veškeré ostatní funkce si každý uživatel může přidat na základě rozšíření dostupných ať už z internetu tak převážně ze zabudovaného Marketplace, obchodu s rozšířeními. Tato rozšíření jsou poskytována zdarma a umožňují do editoru přidat podporu dalších jazyků, technologií, měnit jeho vzhled a přidávat další užitečné funkce.

[1] `https://www.jetbrains.com/webstorm/`
[2] `https://code.visualstudio.com/`

Dalším důležitým prvkem je verzovací systém. Díky předchozím zkušenostem byl zvolen Git a díky nabídce neomezeného množství privátních repozitářů byl jako hosting zvolen GitHub [7].

Díky nepoužívání React Expo bylo nutné mít nainstalované i Android Studio a s ním distribuované nástroje, nicméně pro samotný vývoj využíváno nebylo.

Testování aplikace od samého počátku probíhalo na fyzickém zařízení. Při použití emulátoru by nebylo možné testovat využití rozšířené reality, pouze samotné uživatelské rozhraní, proto jsem tuto možnost vůbec nepoužil. Testovacím zařízením byl mobilní telefon Xiaomi Mi 9 s operačním systémem Android verze 10.

4.2 Použité knihovny

V následující kapitole uvádím výčet těch nejdůležitějších využitých knihoven třetích stran, které byly dostupné s patřičnou licencí a usnadnily mi tvorbu této práce. Celkový přehled je uveden v tabulce 4.1 společně se zmíněnými licencemi a zdroji k těmto knihovnám.

async-storage Tato knihovna poskytuje asynchronní, nešifrované, persistentní, key-value úložiště pro ReactNative. Toto úložiště je vyžito jako cache pro získaná TLE data čímž je ulehčeno zátěži na space-track.org API.

react-native-slider Samotný slider v React Native byl označen jako *deprecated*, tedy zastaralý, a v oficiálních tutoriálech je doporučován právě tento. Jedná se o klasický slider komponent pro výběr hodnoty z předem daného rozsahu. Je rozšířen o snadné možnosti stylování a úpravu chování. V aplikaci je slider využit jako posuvník ovládající zrychlení času.

ESDoc Pro generování dokumentace na základě JSDoc komentářů byla využita knihovna ESDoc. Na rozdíl o samotného JSDoc podporuje ES6 syntaxi a je tedy vhodnější pro tuto práci. Pomocí tohoto nástroje byla vytvořena přiložená dokumentace. Jedinou nevýhodou je již pozastavený vývoj a s tím i nulová podpora ze strany tvůrců. Díky tomu je v dokumentaci několik chyb kvůli nerozpoznání JSDoc tagů i přes jejich správný formát. Samotný ESDoc nemá podporu pro React Native metody a JSX. Tento problém je vyřešen pluginy, které podporu přidávají. Jedná se o *esdoc-ecmascript-proposal-plugin*, *esdoc-jsx-plugin* a v poslední řadě jde o plugin který slouží k propojení ostatních pluginů *esdoc-standard-plugin*.

Jetifier V Android vývoji nastala změna v podobě upgradu stávající Android Support Library, která již není podporována. Nová knihovna nahrazující původní je označována jako AndroidX. V této knihovně došlo k několika

modifikacím z níž hlavní je změna jmenných prostorů starých knihoven. Jetifier umožňuje modifikovat strukturu modulů které ještě AndroidX nevyužívají tak, aby v celé aplikaci bylo právě sjednocené linkování, které využívá AndroidX. Od verze React Native 0.60 je tento balíček již součástí samotného React Native, nicméně v tomto projektu je díky podpoře Viro React používána verze 0.59.

react-native-flash-message Pro potřeby jednoduchých oznámení jsem využil této knihovny, která tento úkol zajišťuje pomocí vyjíždějících či plovoucích oznámení na okraji obrazovky. Použití je navíc velmi snadné díky jejímu globálnímu využití skrze dvě funkce – *showMessage()* a *hideMessage()* pro zobrazení a případné skrytí zprávy. Tyto zprávy jsou v aplikaci zobrazovány při detekci cílů nebo v okamžiku výskytu chyby.

react-native-modal V tomto případě jde o rozšíření běžného React Native modálního okna o lepší animace a možnosti stylování. V aplikaci zastřešuje veškerá modální okna.

react-native-sectioned-multi-select Sectioned Multi Select je vylepšený Select Box, ve kterém je možné vybírat více možností, třídit jednotlivé prvky do kategorií ve kterých lze vyhledávat. Select se otevírá jako modální okno nad aplikací, je tedy přehlednější pro práci s více prvky. Zároveň také spravuje zobrazení již vybraných prvků a jejich případné odebrání. V aplikaci je tento select použitý pro výběr satelitů a pozemních prvků. Na základě stylu zobrazených vybraných prvků bylo vytvořeno zobrazení ručně přidaných satelitů dle jejich identifikátoru.

react-native-sliding-up-down-panel Sliding Up&Down panel dělá přesně to co slibuje jeho název. Jedná se o kontejner pro ostatní prvky, který v zavřeném stavu je pouze lištou ve spodní části obrazovky. Po kliknutí vyjede do poloviny obrazovky a spolu s ním i jeho obsah. Mezi těmito stavy se dá přepínat jak klikáním na záhlaví tak pomocí „slide“ pohybu prstem. Použit byl jako kontejner pro výběry zobrazených elementů a nastavení.

react-native-splash-screen Splash screen je speciální název pro obrazovku, která se zobrazuje před načtením aplikace při spuštění. Využít ji lze například k zobrazení informací o aplikaci při načítání dat na pozadí, v době kdy by aplikace nereagovala na uživatelské akce. V tomto případě je zde však z důvodu licenčních podmínek Viro React.

react-native-vector-icons Tato knihovna se stará o zpřístupnění vektorových ikon z několika zdrojů. Ikony se v aplikaci použijí jako komponent Icon, kde vlastnost *name* určuje vzhled ikony. Seznam jednotlivých ikon je dostupný na GitHubu vývojářů. V aplikaci jsou tyto ikony použity na různých místech, jednak je to povinná závislost Sectioned Multi Selectu, dále jsou použité jako tlačítka pro zobrazení nápovědy, nebo jako různé indikátory.

react-viro Samotný Viro React se do React Native projektu integruje právě díky této závislosti. Jediným problémem je, že podporuje pouze React Native maximálně ve verzi 0.59.x což může činit problém s kompatibilitou jiných knihoven.

satellite.js Satellite.js je javascriptovou knihovnou založenou na knihovně pythonové, která poskytuje veškeré potřebné funkce k použití SGP4/SDP4 modelů pro propagaci TLE dat. Zároveň také tato knihovna poskytuje velké množství funkcí pro převody mezi souřadnými systémy, čehož je využíváno hojně při zobrazování dat uživatelům.

Název	Zdroj	Licence
Async Storage	`https://github.com/react-native-community/async-storage`	MIT
Slider	`https://github.com/react-native-community/react-native-slider`	MIT
ESDoc	`https://github.com/esdoc/esdoc`	MIT
Jetifier	`https://github.com/mikehardy/jetifier`	MIT
Flash Message	`https://github.com/lucasferreira/react-native-flash-message`	MIT
Modal	`https://github.com/react-native-community/react-native-modal`	MIT
Sectioned Multi Select	`https://github.com/renrizzolo/react-native-sectioned-multi-select`	MIT
Sliding Panel	`https://github.com/Abhijeet-Ashapure/react-native-sliding-up-down-panel`	-
Splash Screen	`https://github.com/crazycodeboy/react-native-splash-screen`	MIT
Vector Icons	`https://github.com/oblador/react-native-vector-icons`	MIT
Viro React	`https://github.com/viromedia/viro`	MIT
Satellite.js	`https://github.com/shashwatak/satellite-js`	MIT
Timetravel	`https://github.com/pbock/timetravel`	MIT
React	`https://github.com/facebook/react`	MIT
React Native	`https://github.com/facebook/react-native`	MIT

Tabulka 4.1: Použité knihovny

Timetravel Timetravel je malou knihovnou umožňující simulaci času. Velmi

užitečná je možnost zrychlit čas oproti času reálnému. Veškerý čas v aplikaci se právě řídí tímto časem. Díky tomu je právě možné čas libovolně zrychlovat a zpomalovat.

React a React Native Stejným způsobem je do projektu přidána závislost i na React a React Native, což umožňuje jejich jednoduchou aktualizaci a změnu verzí.

4.3 Správa dat a Space-Track API

V této sekci bych rád přiblížil správu dat v aplikaci a popsal způsob jakým jsou získávána podkladová data pro veškeré výpočty ohledně pozic satelitů.

4.3.1 Space-Track API

Cílem Space-Tracku pro vytvoření tohoto API bylo nahrazení dosavadního stahování kompletních TLE záznamů v podobě textových souborů a také odstranění parsování webových stránek kde byla data zobrazována. Bylo tedy vytvořeno rozhraní využívající REST architekturu, které bylo doplněno i grafickým rozhraním v podobě webové služby.

API je vývojářům dostupné zdarma pod podmínkou registrace, po které vývojář obdrží svůj API klíč, pomocí kterého se v požadavcích identifikuje. Limitujícím faktorem je omezení na počet dotazů na 20 za minutu respektive 200 za hodinu a počet dotazů pro automatizované skripty, který je omezen v závislosti na typu dotazu [37].

V dokumentaci na webu je uvedeno množství ukázkových dotazů, které zahrnují použití téměř všech možností, které API poskytuje. Mohou to být dotazy na samotný katalog SATCAT, dotazy pro získání TLE záznamů, popřípadě doplňkové dotazy typu získání všech startovacích lokací, nejnověji přidaných záznamů atd. V případě této práce je využíván pouze jeden druh dotazů, a to je získání nejnovějších TLE záznamů pro satelity s konkrétním ID.

4.3.2 Získání dat

Veškerá práce se Space-Track API je v aplikaci zastřešena v rámci třídy SpaceTrack a její metody *getTLEs()* která jako parametr přijímá list ID satelitů, pro které chceme nejnovější TLE záznamy získat. Samotná třída se nestará o kontrolu záznamů v cache aplikace a vrátí tedy TLE záznamy pro všechna poskytnutá ID.

Dle dokumentace by se samotné získání dat mělo dělit do dvou částí. První je autentifikace a až poté samotné požadavky na data. Odpovědí autentifikačního POST požadavku by měl být soubor s cookies, kterým se následně vývojář identifikuje v dotazech na data. Nicméně v React Native se mi nepodařilo zprovoznit toto získání cookies žádným způsobem, jelikož veškeré knihovny a požadavky vyžadovaly moduly, které jsou dostupné pouze v desktopových prohlížečích. Možností byla ruční správa cookies, ale této možnosti jsem se prozatím vyhnul.

Druhou možností je provést obě operace v jednom dotazu. V tomto případě není výsledkem cookie, ale rovnou samotná data. Touto cestou jsem se vydal a sestavil následující dotaz, který je blíže popsán v tabulce 4.2.

```
https://www.space-track.org/basicspacedata/query/class/tle_lat
est/ORDINAL/1/NORAD_CAT_ID/.../orderby/TLE_LINE1~ASC/format/tle
```

Část dotazu	Popis
https://www.space-track.org/	základní URL
basicspacedata/	požadovaný controller
query/	požadovaná akce
class/tle_latest/ORDINAL/1/	selekce pouze nejnověji přidaného TLE záznamu pro každý objekt
NORAD_CAT_ID/.../	seznam ID satelitů oddělených čárkou např.: „25544, 00032, . . . “
orderby/TLE_LINE1 ASC/	seřazení dle prvního řádku TLE záznamů vzestupně dle abecedy
format/tle	formát výstupu budou „tradičně“ formátované TLE v textové podobě

Tabulka 4.2: Popis dotazu na SpaceTrack API

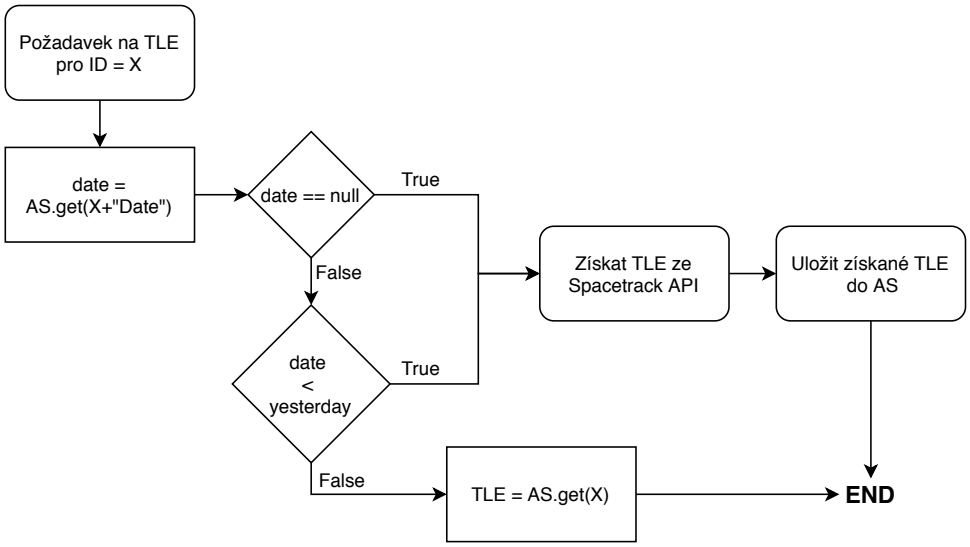
4.3.3 Správa dat

Požadavky na získání dat pramení z komponentu Globe, který se stará o správu veškerých vizualizovaných prvků a spravuje tak i seznam satelitů které mají být zobrazené. Pokud by však aktualizace veškerých TLE záznamů probíhala s každým aktualizováním tohoto seznamu jednalo by se o značnou zátěž na API, která by při větším množství uživatelů značně přesáhla limity stanovené poskytovatelem.

První optimalizací je tedy separace nově zvolených satelitů a jejich oddělení od již zobrazených. V případě že by se aplikace používala pouze jednou denně

by toto mohlo dostačovat, získaly by se pouze TLE záznamy pro nově vybrané satelity a ostatní by zůstaly stejné. Stále je zde však možnost opakovaného odebírání a přidávání těch samých satelitů například z důvodu porovnání a v tom případě by se opakovaně tyto záznamy získávaly z internetu.

Na řadu zde přichází druhá optimalizace v podobě Async Storage. Jedná se o key-value storage s omezením, že hodnota je limitována na datový formát string. To pro ukládání TLE záznamů nepřináší žádnou přítěž. Při prvotním stažení dat z internetu nebo při aktualizaci je do tohoto úložiště uložen pod klíčem ID daného satelitu string s TLE daty. Aby bylo možné data udržovat aktuální je v tento okamžik také uložen pod klíčem formátu ID satelitu + „Date“ současný UNIX timestamp. Pokud aplikace vyžaduje TLE data konkrétního satelitu, zkontroluje se nejprve Async Storage. Pokud se zde záznam nachází je zkontrolováno jeho stáří a poté je buď použit nebo je záznam získán z internetu. Přehledně tuto situaci zobrazuje následující diagram na obrázku 4.1.



Obrázek 4.1: Proces získání TLE dat

4.4 Detekce cílů vizualizace

Velkou část této práce tvoří detekce cíle, nad kterým se samotná vizualizace bude zobrazovat (obrázek 4.2). V ideálním případě jde o globus, nicméně v detekci nejsou použity metody počítačového vidění pro například detekci koule, ale jedná se o detekci globusu na základě předem definovaných fotek. Pro uživatele kteří nedisponují konkrétním globusem je možnost využití vytištěného obrázku jako cíle vizualizace.

Problémem v této části aplikace je velký rozdíl mezi přístupem k této problematice na jednotlivých platformách. V mých silách byl pouze vývoj a

testování na platformě Android z důvodů zmíněných v kapitole 2.3.5 v části o kompilaci pro zařízení Apple. Celá sekce se tedy bude týkat implementace pro platformu Android a v posledním bodě nastíním možná řešení pro zařízení Apple.

(a) : Planární cíl

(b) : 3D cíl

Obrázek 4.2: Dva druhy cílů

4.4.1 Planární cíl

Původně byl tento cíl používán primárně pro testovací účely, nicméně dávalo mi smysl ponechat tuto možnost i ve finální verzi aplikace. Samotným cílem je v aplikaci mapa Země v pohledu na severní pól (obrázek 4.3). Tento pohled byl zvolen pro intuitivnější znázornění rotace.

Detekce obrázku je ve Viro React jednoduchou záležitostí. Stačí použít komponent ViroARImageMarker, kterému se jako vlastnost předá název předem nadefinovaného cíle (tracking targetu). Tento target se skládá z odkazu na obrázek, definice jeho orientace a jeho fyzické velikosti. Po detekci nadefinovaného cíle se vyrenderuje na jeho pozici obsah, jehož rodičem v hierarchii komponent je právě konkrétní *ViroARImageMarker*. V této práci jsem ovšem použil jiný princip. Při každé změně detekovaného cíle je volána funkce *_onAnchorUpdated()* do které je předána *anchor* (objekt reprezentující cíl a událost která se odehrála) a název cíle na kterém se tato událost odehrála.

Z této kotvy lze přečíst detekovanou pozici v prostoru scény a také rotaci oproti výchozí nadefinované pozici. V případě planárního cíle jsou tyto detekované vlastnosti přímo nastavovány celému komponentu Globe a starají se tak o pozicování a správnou rotaci celé vizualizace.

Obrázek 4.3: Planární cíl vizualizace

4.4.2 3D cíl

Zařízení s operačním systémem Android v době psaní této práce neumožňují detekci 3D objektů v prostoru díky absenci této funkcionality v ARCore. Bylo tedy nutné vymyslet, jak detekovat globus pomocí jiných metod. Jediným možným řešením bylo nahrazení detekce 3D objektu detekcí několika různých obrázků.

Prvotní nápad byl s vyfocením globusu z několika úhlů kolem celého jeho obvodu a z toho následně rozpoznat část mířící k uživateli a podle toho nastavovat rotaci objektu. Pro pozici by stačilo pouze posunout globus o jeho poloměr směrem od uživatele na místě detekovaného obrazu. Tento nápad ovšem nefungoval díky špatné detekci jednotlivých fotek globusu. Po vyhodnocení jednotlivých snímků skrze nástroj společnosti Google pro detekci kvality ImageTargets, byly ohodnoceny průměrně 25 body ze 100 možných (Google doporučuje obrázky s hodnotou nad 75 bodů). Provedl jsem tedy doporučené úpravy dle autorů jiné AR knihovny podle jejich tutoriálu [28]. Výsledek těchto úprav je vidět na obrázcích 4.5 a 4.4, kde první zmíněný je neupravený a výřezy jsou po proběhlých úpravách. Po aplikaci těchto úprav jsem se dostal na hodnocení okolo 40 bodů což se stále jevilo jako velmi nedostačující. Na jednotlivých snímcích nebylo zřejmě dostatečné množství bodů, které by právě detekce zachytávala a bylo od tohoto způsobu upuštěno.

Druhou verzí bylo tedy vyfocení pouze menších částí globusu, na kterých bylo větší množství informací. Jde tedy o různá menší geografická území typu ostrov, kontinent, apod. Nicméně tyto cíle nebylo možné detekovat z větší vzdálenosti.

Obrázek 4.4: ImageTargets v podobě menších územních celků

Třetí a také finální verze byla upravena dle výsledků předchozího testování. Rozhodl jsem se nefotit celý globus z různých úhlů kolem obvodu, ale vyfotit oblasti globusu na kterých bylo více dat k detekci. Tento postup už zajistil průměrné hodnocení fotek okolo 60 bodů. Zkoušel jsem i úpravy kontrastu a podobně, nicméně nejlepších výsledků dosahovaly snímky odpovídající skutečnému vzhledu.

Obrázek 4.5: Finální ImageTarget

Název	Rotace [°]
africa.jpg	15
atlantic.jpg	340
australia.jpg	135
china.jpg	90
europe.jpg	15
hawaii.jpg	196
indonesia.jpg	140
northAmerica.jpg	260
southAmerica.jpg	300

Tabulka 4.3: Rotace ImageTargetů

Rotaci celého objektu není možné určovat pomocí jediného detekovaného obrázku, protože někdy detekce zareaguje velmi brzy, někdy má trochu prodlevu. To samé platí při oddetekování cíle. Pro každý ImageTarget jsem si nadefinoval jeho rotaci, poledník uprostřed snímku, a ze seznamu veškerých v současné době detekovaných cílů je počítána průměrná rotace (tabulka 4.3). S pozicí se děje to samé, akorát se průměr nepočítá z detekcí za celou dobu běhu, ale pouze z několika prvních. Poté se objekt ukotví ve scéně a má pevně danou lokaci. Při počítání průměrné pozice z n posledních detekcí se často

vyskytovaly chyby díky prodlevě detekce a virtuální globus se nacházel daleko od skutečného.

Problémem tohoto řešení je využívání vlastnosti *trackingMethod* u jednotlivých anchors. Jedná se o propagovanou vlastnost poskytovanou ARCore frameworkem [3]. Může nabývat tří stavů: *notTracking* pro target který ještě nebyl nikdy spatřen, *tracking* pro target který je aktivně sledován a *lastKnownPose* pro target který byl někdy detekován, ale nyní už není. Díky této vlastnosti je možné rozlišit, jestli je právě target vidět nebo ne. Jak ale bylo zmíněno, jedná se o vlastnost ARCore frameworku a na zařízeních Apple tedy *není* dostupná.

Princip fungování

Detekce planárního i 3D cíle probíhá ve funkci, která je volána při aktualizaci každého cíle. Parametry této funkce jsou *anchor* - Viro React objekt reprezentující detekovaný cíl a *targetName* - jméno detekovaného ImageTargetu. Výsledná vypočtená pozice a rotace je z funkce nastavena do stavu celé komponenty, což umožňuje snadnou a automatickou aktualizaci při jakékoliv detekované změně. Pseudokód této detekce je možné vidět v následujícím algoritmu 1.

Algoritmus 1 Aktualizace pozice a rotace vizualizace

```
1:  function _onAnchorUpdated(anchor, targetName)
2:      if targetName is being detected then
3:          if targetName is flatTarget then
4:              copy rotation and position from anchor
5:              return
6:          else
7:              if targetName not in trackedTargets then
8:                  add targetName to trackedTargets and keep newest n
9:      else if targetName is not being detected then
10:         if trackedTargets include targetName then
11:             remove targetName from trackedTargets
12:     updateVisualizationRotation( )
13:     updateVisualizationPosition(anchor.position)
14: function updateVisualizationRotation( )
15:     angle ← average from defined rotations for each trackedTargets
16:     set rotation around Y axis by angle
17: function updateVisualizationPosition(position)
18:     if updateCount < maxUpdateCount then
19:         increase updateCount
20:         set position as weighted average of current position and position
21:         move position in -Z direction by globe radius
```

4.4.3 Řešení pro zařízení Apple

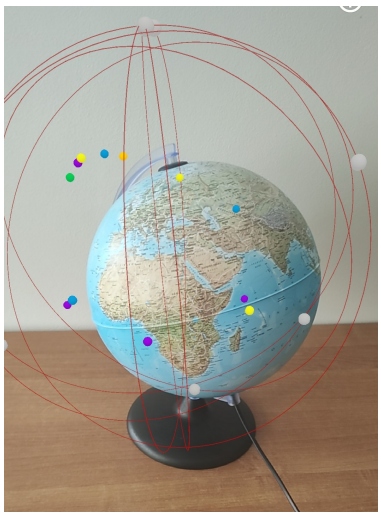
Aby detekce byla funkční i na zařízeních firmy Apple bylo by nutné napsat kód specifický pro každou platformu uvnitř *_onAnchorUpdated()* metody. V případě Androidu by bylo použito stávajícího řešení, pro zařízení Apple by se musely provést následující dvě úpravy. Pro planární cíl by se neověřovalo, jestli je target ve stavu tracking, ale pokaždé kdy by se provedl update cíle by se zkontrolovalo, jestli má nadefinovanou pozici a rotaci a tyto hodnoty by se předaly do komponenty Globe.

Úpravy v případě 3D cíle by byly větší. ARKit ve verzi 2.0+ umožňuje detekci 3D objektů, na základě cílových objektů které lze vytvořit dle dostupné dokumentace [35]. V takovém případě by se provedlo naskenování globusu a následně by neprobíhala detekce obrazových cílů, ale právě 3D objektu. Tato detekce by zároveň určovala pozici a rotaci ve scéně.

4.4.4 Zobrazení virtuálního globusu

Pro planární cíl se jedná o kompletní 3D model Země, který levituje nad detekovaným obrázkem (obrázek 4.2a).

V případě globusu jako cíle je situace o něco složitější. V dřívějších verzích skutečný globus byl překryt také 3D modelem země. Dle uživatelské zpětné vazby však působil příliš rušivě. Po odstranění tohoto modelu a ponechání pouze samotného globusu se při nepřesné detekci vizualizace stávala chaotickou. Navíc díky tomu že v současné verzi ARCore ani ARkit neumí pracovat s detekcí hloubky v obraze, byly satelity a orbity, jež by měly být globusem zakryty, viditelné.



(a) : Žádné pomůcky

(b) : Průhledný globus

Obrázek 4.6: Porovnání verzí vizualizace nad globusem

Řešením bylo přidání dvou objektů. Jedním je koule s průhledným materiálem, který ovšem zakrývá virtuální objekty za sebou a druhým je koule s texturou reprezentující pouze obrysy jednotlivých kontinentů, a navíc je ji nastavena průhlednost. Tato koule pro uživatele reprezentuje místo, kde aplikace „vidí“ skutečný globus a je poté jednodušší se s rozšířením reality vypořádat. Porovnání těchto metod je na obrázku 4.6.

4.5 Vizualizace satelitů

V této kapitole přiblížím situaci ohledně vizualizace samotných satelitů. Objasním postupy používané pro získání dat pro vizualizaci, transformace souřadnic, samotné vykreslení a nakonec i propojení se zobrazením podrobných informací pro uživatele.

4.5.1 Podkladová data

Veškeré satelity zastřešuje komponent Globe, nicméně každý satelit je reprezentován samostatným objektem SatelliteObject. Jedná se o kontejner nad TLE daty, která jsou zde reprezentována díky knihovně satellite.js jako tzv. Satellite Record. Dále podpůrnými daty odvozených z TLE, popřípadě speciálních konstant potřebných k vykreslení a několika metodami.

Hlavní metodou by se dala označit metoda *updatePosition()*, která jako parametr obdrží UNIX timestamp okamžiku ve kterém má být pozice vypočtena. Díky Satellite Record v objektu a obdrženému timestampu je možné zavolat funkci *propagate()* knihovny satellite.js. Tato funkce zpřístupňuje použití SGP4 modelu a vrací ECI souřadnice a vektor rychlostí v konkrétní okamžik. Tato data jsou uložena pro pozdější zobrazení informací nicméně pro vizualizaci nejsou ECI souřadnice přímo použitelné.

Prvním problémem je příliš velký rozsah ECI souřadnic, které jsou udávány v kilometrech. Jelikož souřadnice ve Viro scéně mají jako základní jednotku metry není možné tyto souřadnice na sebe namapovat přímo. Pro zmenšení rozsahu, ale zachování poměru jsou hodnoty děleny konstantou.

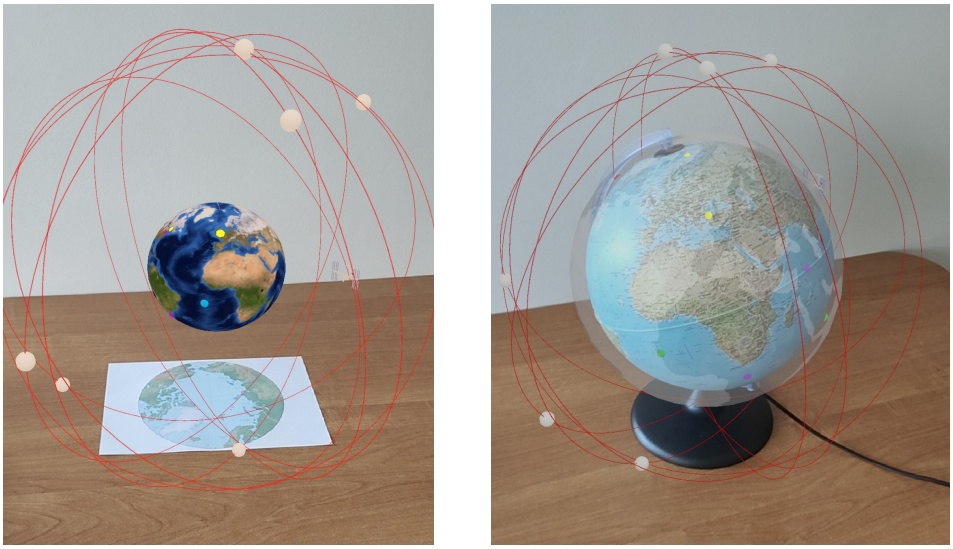
Dalším problémem je jiné značení os těchto souřadných systémů. Tento problém je vyřešen patřičnou změnou jednotlivých souřadných os, která může být reprezentována transformační maticí v podobě jednotkové matice s přeházenými řádky (sloupci), nebo jako v našem případě pouhou změnou pořadí hodnot v poli.

Poslední problém vzniká díky neustálé rotaci Země. Jak bylo zmíněno ECI souřadný systém se zemí nerotuje a konkrétní místo na Zemi má v průběhu času jiné souřadnice. Tuto rotaci je tedy potřeba jakýmsi způsobem kompenzovat. Tato kompenzace je možná za využití souřadného systému ECEF,

který se od ECI liší pouhou rotací kolem osy Z. Danou rotaci lze spočítat jako úhel mezi osou X ECEF a ECI systému. Tento úhel následně udává potřebné natočení Země. V našem případě je využita záporná hodnota která udává otočení satelitů. Otočení právě satelitů je nutné díky potřebě řídit orientaci Země podle detekovaného cíle vizualizace.

4.5.2 Vykreslení

Vykreslování samotných satelitů probíhá, díky vlastnostem Reactu, při každé aktualizaci stavu komponentu. O tyto pravidelné aktualizace se stará timer, který každých n milisekund zavolá funkci *updatePosition()* na každém satelitu s timestampem konkrétního okamžiku. Na základě těchto nových pozic je poté list veškerých zobrazených SatelliteObject objektů pomocí funkce map převeden na list 3D objektů s patřičnou pozicí ve scéně. Vzhled 3D objektu není pevně definován, ale záleží na hodnotě ve statickém seznamu třídy SatelliteObject. Na základě této hodnoty se vybere buď specifický 3D model, nebo je satelit znázorněn pomocí generické koule. Na obrázku 4.7 je v obou případech vidět satelit reprezentovaný symbolicky koulí a také skutečným modelem - konkrétně ISS.



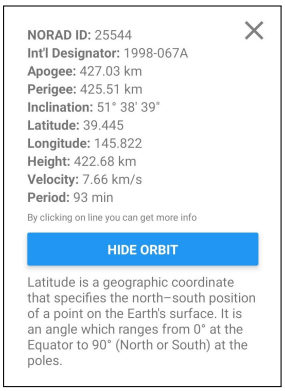
(a) : Vizualizace nad planárním cílem **(b) :** Vizualizace nad globusem

Obrázek 4.7: Výsledná vizualizace

4.5.3 Zobrazení informací

Zobrazení informací k jednotlivým satelitům je zprostředkováno pomocí modálního okna (obrázek 4.8). Toto modální okno lze otevřít kliknutím na konkrétní satelit. Inspirací pro zobrazované informace byla stránka N2YO.com, kde mi soubor informací přišel dostatečně obsáhlý, ale zároveň se nejednalo

o žádné údaje, které by byly těžko pochopitelné pro mladší uživatele. Přesto jsou tyto hodnoty doplněny možností rychlého vysvětlení konkrétního pojmu po kliknutí na něj.



Obrázek 4.8: Info Modal komponent

V modálním okně je dostupné tlačítko umožňující skrýt orbitu konkrétního satelitu. Tato možnost je zde z důvodu výchozího nastavení, kdy se orbita automaticky vykresluje pro každý zvolený satelit.

Problémem tohoto řešení je občas špatná detekce kliknutí na satelit i když mají jednotlivé modely nastaveno aby se renderovaly jako úplně poslední objekty ve scéně díky vlastnosti *renderingOrder*. Někteří uživatelé doporučovali nastavení vlastnosti *highAccuracyEvents* na hodnotu `true` [8]. Nicméně ani toto nepřineslo změnu do detekce. Jediná spolehlivější cesta je provádět dotyk pomaleji a prst tak podržet na příslušném objektu o něco déle, než by člověk intuitivně udělal.

4.5.4 Orbity

Jak bylo zmíněno aplikace je schopná vykreslovat orbity jednotlivých satelitů a to za pomoci komponentu ViroPolyline. Jedná se o 2D křivku v 3D prostoru, která prochází všemi body předanými ve vlastnosti points. S tím že není uzavřená - je tedy potřeba první bod zopakovat i na konci seznamu aby šlo o uzavřenou elipsu. V globálním nastavení aplikace je uživateli dána možnost měnit průhlednost orbit.

O získání bodů skrze které bude křivka reprezentující orbitu procházet se stará SatelliteObject a metoda *getPointsForOrbit()*. Parametry této funkce je počet segmentů tvořící výslednou křivku a počáteční UNIX timestamp orbity. Proces získání bodů ilustruje následující algoritmus 2.

Algoritmus 2 Získání bodů pro křivku reprezentující orbitu

```
1:  procedure getPointsForOrbit(segmentCount, timestamp)
2:      period ← satellite.getPeriod()
3:      timestep ← period ÷ segmentCount
4:      points ← [ ]
5:      i ← 0
6:      for i = 0; i ≤ segmentCount do
7:          coordinates ←satellite position at time (timestamp+i∗timestep)
8:          add coordinates to points
9:      add points[0] to points              ▷ Repeat first point to close polyline
10:     return points
```

4.6 Vizualizace pozemního segmentu

Vizualizace pozemního segmentu byla jednodušší díky statickým datům – jednotlivé segmenty jsou na konkrétních lokacích ze kterých se nepohybují. Na rozdíl od satelitů však lokace těchto segmentů nejsou dostupné z žádného API, pouze jako seznam lokací na oficiálních internetových stránkách systému GPS [9] . V této kapitole popíšu proces získání souřadnic jednotlivých segmentů a jejich následnou vizualizaci.

4.6.1 Získání pozic

Nikde jsem nenašel konkrétní lokace jednotlivých prvků, pouze obecné informace typu město, u některých například konkrétní vojenská základna. Pro vizualizaci v rozšířené realitě to však bude více než dostačující. Pro převod souřadnic byla vytvořena třída CoordConverter se slovníky (dictionary, key-value storage) ve kterých jsou uloženy lokace měst v zeměpisných souřadnicích a metodou *convert()*, která za pomoci knihovny satellite.js převede zeměpisné souřadnice na ECI.

Pro tento převod je nutné mít kompletní trojici zeměpisné šířky, délky a nadmořské výšky. Každému městu tedy byla nastavena výška 800 metrů nad mořem. Knihovna neumožňuje přímý převod mezi zeměpisným a ECI souřadným systémem, bylo nutné použít prostředníka v podobě souřadného systému ECEF. Navíc pro převod z ECEF do ECI souřadnic je nutné dodat ještě hvězdný čas, který je díky knihovně možný získat z JavaScriptového objektu Date. Výsledné pozice je tedy pro zobrazení ještě nutné orotovat okolo zemské osy o úhel který je rozdílem mezi natočením os X systému ECI a ECEF v době převodu.

4.6.2 Vizualizace a poskytnutí informací

Prvky pozemního segmentu se dělí do několika skupin čehož je využito jak při výběru tak při vizualizaci jednotlivých prvků. Výběr je realizován pomocí Sectioned Multi Selectu ve kterém je navíc použito zakázání některých voleb pro zobrazení informací o barevném odlišení ve vizualizaci. Vykreslení jednotlivých zvolených prvků je provedeno stejným způsobem jako vykreslení satelitů s tím rozdílem, že nyní není funkce map použita na objekty, ale na ID jednotlivých prvků podle kterých jsou dohledány pozice a materiály (barvy) koulí reprezentujících jednotlivé lokality viz obrázek 4.7a.

Pro vysvětlení funkcí jednotlivých segmentů je vedle Selectu dostupná ikona nápovědy, která po kliknutí otevře modální okno s vysvětlením a opětovným připomenutím barevného rozdělení prvků do konkrétních skupin.

4.7 Obsah a funkce jednotlivých souborů

V této kapitole bych chtěl shrnout výslednou implementaci a popsat funkce jednotlivých souborů starajících se o funkčnost této aplikace.

App.js Třída App je vstupním bodem aplikace. Nachází se zde definice veškerého uživatelského rozhraní a propojuje zároveň komponenty na různých úrovních tak aby mezi nimi byla umožněna komunikace. Díky tomu že se zde nachází definice uživatelského rozhraní se stará také o správu zvolených prvků, zvolenou rychlost času a tyto informace distribuuje do potřebných komponent.

js/ARScene.js ARScene je komponent, který zajišťuje veškeré funkce rozšířené reality v aplikaci. V prvé řadě je to zobrazení videa z kamery telefonu jako pozadí výchozí obrazovky. Dále rendering všech virtuálních objektů a detekce cílů pro vizualizaci.

js/SatelliteObject.js SatelliteObject je třída zastřešující jednotlivé satelity, které jsou vizualizovány. Jedná se jak o kontejner pro data týkajících se satelitu, tak zároveň i o místo kde jsou veškeré funkce pro počítání pozice satelitu, získávání informací do výpisu pro uživatele a podobně.

js/SpaceTrack.js SpaceTrack je třída poskytující přístup k REST API space-track.org. V této aplikaci je využíváno pouze pro přístup k TLE záznamům satelitů, ale v případě rozšíření by například funkce pro práci s katalogem SATCAT byly právě zde.

js/components/Globe.js Tento komponent zastřešuje celou vizualizaci ať se jedná o vizualizaci nad planárním nebo 3D cílem. Konkrétně se stará se o aktualizaci pozic a vykreslení satelitů, vykreslení orbit a kompenzaci

rotace Země a vykreslení pozemních segmentů. Důležitou funkcí je také správa času pro možnost zrychlení či zpomalení vizualizace.

js/components/CustomInfoModal.js CustomInfoModal je komponent používaný pro zobrazení informací o konkrétním satelitu na základě předaného ID. Stará se o aktualizaci svých dat a jejich výsledné vykreslení.

utils/categoryScrape.py Tento pomocný skript slouží k tvorbě JSON slovníku jako datového podkladu Sectioned Multi Select pro výběr satelitů. Na základě předem definovaných adres stáhne data z webu celestrak.org, upraví je a roztřídí do předem zvolených kategorií. Skript je nutno spustit ručně v případě potřeby aktualizace těchto dat.

utils/coordsConverter.js Druhý pomocný skript se stará o konverzi souřadnic lokací na Zemi ze zeměpisných do ECI souřadnic pro možné vizualizování. Použit byl pro konverzi souřadnic lokací prvků pozemního segmentu GPS.

Kapitola 5

Testování

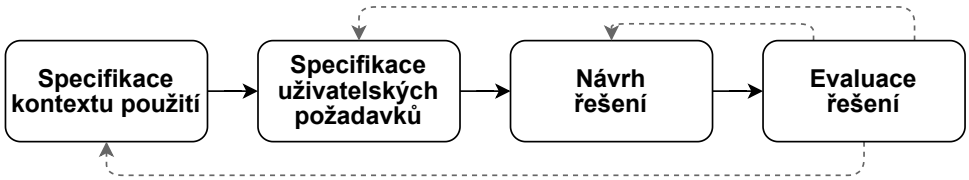
V následující kapitole bude popsáno testování výsledné aplikace vyvinuté v rámci této bakalářské práce. Celý vývoj využíval principů User Centered Designu, tedy iterativního vývoje s velkým důrazem na zpětnou vazbu uživatelů. Po dokončení vývoje byla provedena série Usability testů s uživateli odpovídající cílové skupině uživatelů této aplikace. Na základě zpětné vazby z testování byly vyvozeny důsledky a patřičné úpravy aplikace.

5.1 User-Centered Design

Definici tohoto pojmu si téměř každý autor vykládá podle svého, nicméně všechny mají spoustu částí společných. Dovolil jsem si vybrat následující, z mého pohledu jednodušší avšak plně vystihující definici celého procesu.

„User-Centered Desing (UCD) nastiňuje fáze životního cyklu návrhu a vývoje produktu, přičemž se soustředí na získání hlubokého porozumění tomu, kdo a jak bude výsledný produkt používat.“ [42]

Existuje mnoho principů a fází tohoto procesu z nichž ty nevíce zmiňované jsou: Specifikace kontextu použití produktu, Specifikace uživatelských požadavků, Návrh řešení a Evaluace. Možný průběh vývoje nastiňuje diagram 5.1. Z pohledu na tento diagram je vidět že není žádný předem daný vývojový cyklus, z evaluační části je možné se vrátit k libovolnému předchozímu kroku.



Obrázek 5.1: Proces User-Centered Designu

Tyto principy byly při vývoji uplatňovány a aplikace byla na základě zpětné vazby průběžně upravována. Příklad uvažované cílové skupiny a jejích požadavků je uveden v kapitole 3.1.

5.2 Finální Usability testy

Po dokončení vývoje aplikace bylo nutné otestovat její použitelnost na referenční skupině uživatelů. Toho bylo docíleno díky principům Usability testování. Základním principem tohoto testování je předložení typických úkolů souvisejících s využíváním produktu participantům, zatímco pozorovatel (v tomto případě samotný vývojář) sleduje jejich počínání a zaznamenává si jeho průběh a případné poznámky. Výsledkem tohoto testování je zjištění problémů týkajících se použitelnosti produktu a seznámení se se spokojeností uživatelů s aplikací [41].

5.2.1 Testovací skupina a průběh testování

Testovací skupina byla zvolena na základě předchozí analýzy cílové skupiny. U každého z participantů je možnost, že by produkt někdy využil. I přes omezené možnosti se podařilo domluvit celkem 5 participantů s minimálně jedním reprezentantem v každé kategorii. Bližší rozdělení je uvedeno v tabulce 5.1. Vzhledem k tomu, že pouze jediný uživatel měl vlastní zařízení splňující požadavky aplikace a možnosti vývoje (viz kapitola 2.3.5) a navíc je tento telefon technickými specifikacemi téměř totožný se zařízením, na kterém byla aplikace vyvíjena v průběhu proběhly všechny testy právě na zařízení Xiaomi Mi 9 s operačním systémem Android ve verzi 10.

Kategorie	Počet
Žák základní školy	1
Žák střední školy	1
Rodič žáka	1
Nadšenec	2

Tabulka 5.1: Rozdělení participantů testování dle uživatelských skupin

Samotné testování probíhalo v místnosti, kde se nacházel pouze participant a vývojář v pozici pozorovatele. Všichni participanti dostali za úkol projít sadu úkolů, které pokrývají veškeré možnosti použití aplikace. Se samotnou aplikací byli všichni uživatelé v rychlosti seznámeni a zároveň měli k dispozici připravený globus a vytištěný planární cíl. Vývojář v roli pozorovatele nijak s úkoly nenapomáhal, v některých případech pouze upřesnil překlad z angličtiny a vyřešil krizové situace.

5.2.2 Testovací scénáře

V následující kapitole představím všechny použité testovací scénáře a jejich očekávaný průchod.

- Scénář 1

  Zadání: Chcete se dozvědět více informací o testované aplikaci. Pokuste se tedy nalézt informace o jejím používání a prozkoumejte dostupná menu.

  Očekávané řešení: Uživatel klikne na ikonu otazníku a přečte si informace v otevřeném modálním okně. Následně modální okno zavře, na úvodní obrazovce vysune panel s nastavením a případně zkusí otevřít některé select prvky, popřípadě informace k pozemnímu segmentu.

- Scénář 2

  Zadání: Naskenujte planární cíl. Poté zobrazte libovolný satelit z kategorie GPS.

  Očekávané řešení: Uživatel namíří telefon na papír s vytištěným obrázkem, po zobrazení virtuálního globusu vysune menu a v select boxu pro výběr z kategorií vybere v kategorii GPS náhodný satelit, který si následně prohlédne ve vizualizaci.

- Scénář 3

  Zadání: Odeberte zobrazený GPS satelit a zobrazte všechny satelity z kategorie Education a satelit s názvem FORMOSAT-3 FM1.

  Očekávané řešení: Uživatel ve výsuvném panelu klikne na křížek u zvoleného GPS satelitu. Dále v prvním selectu vybere celou kategorii Education klepnutím na ní. Poté ve vyhledávacím řádku zadá text FORMOSAT a vybere správný satelit z navržených výsledků. Vybrané satelity následně zkontroluje ve vizualizaci.

- Scénář 4

  Zadání: Restartujte aplikaci a naskenujte globus jako cíl. Zobrazte si prvek pozemního segmentu v Jihoafrické republice, USA a na ostrově Diego Garcia. Pokuste se všechny prvky spatřit v rozšířené realitě.

  Očekávané řešení: Po restartování aplikace a následném zamíření telefonu na globus s menší odezvou proběhne jeho detekce. V selectu pro pozemní segment uživatel následně zvolí požadované prvky. Po namíření na vizualizaci otočí globusem, nebo přesunutím po místnosti zkontroluje všechny strany vizualizace.

- Scénář 5

  Zadání: Odeberte veškeré zobrazené pozemní segmenty a pokuste se zjistit aktuální rychlost oběhu stanice ISS (ID 25544).

  Očekávané řešení: Uživatel nejprve odebere všechny prvky pozemního segmentu pomocí tlačítka Remove All. Následně zvolí ISS z prvního selectu nebo stanici přidá ručně pomocí jejího ID. Poté klepnutím na zobrazený model ISS otevře modální okno s informacemi ohledně ISS a vyčte příslušnou rychlost.

- Scénář 6

  Zadání: Zobrazte si hlavní kontrolní stanici pozemního GPS segmentu a přečtěte si v informacích něco o jejím fungování. Dále zobrazte všechny satelity kategorie Education. Skryjte orbitu pro ISS a pro ostatní orbity snižte jejich průhlednost na minimum. Dále zrychlete čas simulace na maximum. Po prohlédnutí vizualizace veškeré vizualizované prvky odeberte a aplikaci ukončete.

  Očekávané řešení: Uživatel nejprve zvolí první prvek v selectu pro pozemní segment, kliknutím na ikonku písmene i v kruhu otevře modální okno s informacemi o pozemním segmentu a přečte si požadované informace. Dále v prvním selectu zobrazí všechny satelity kategorie Education kliknutím na její název. Skrytí orbity ISS je možné z modálního okna po kliknutí na její model. Snížení průhlednosti všech ostatních orbit se nachází v okně informací o aplikaci, které bylo použito v prvním scénáři. Následné zrychlení času provede pomocí slideru ve výsuvném panelu. Odebrání všech vizualizovaných prvků je možné skrze tlačítka Remove All v každé sekci.

  Jedná se o nejkomplexnější úkol, jaký se dá v této aplikaci provést, pokud jej uživatel zvládne můžeme ho považovat za zkušeného uživatele.

5.2.3 Průběh testování

Na začátku uživatelé dostali krátké shrnutí aplikace a jejích možností. Dále jim byl předán papír s jednotlivými vytištěnými úkoly pokud by si je raději sami četli.

- Scénář 1

  Průběh: Všichni participanti tento úkol zvládli bez obtíží. Okno s informacemi o aplikaci nalezli ihned a většina z nich také vyzkoušela prozkoumat výsuvný panel a v něm obsažené select boxy.

  Poznatky: Umístění nápovědy je na správném místě a znázornění možnosti vysunutí spodního panelu je také dostatečné.

- Scénář 2

  Průběh: Dva uživatelé se nejprve pokusili kliknout na tlačítko ADD u přidání satelitu dle jeho ID a nebylo ošetřené přidání prázdného ID což vedlo k pádu aplikace. Následně se už všichni správně dostali k do výběru kategorií nicméně zhruba polovina uživatelů se pokusila kategorii rozbalit kliknutím na její název což vedlo k vybrání celé kategorie. Skenování planárního cíle probíhalo bez obtíží.

  Poznatky: Změnit pořadí prvků ve výsuvném panelu, aby první bylo přidání podle ID, pokud to bude první možnost uživatelé si tím spíše přečtou o co se jedná a nebudou ji využívat nesprávně. Další úpravou by mohlo být zvýraznění možnosti rozbalit kategorie pomocí výraznější šipky na straně, popřípadě ji rozbalit klepnutím na název. Tato možnost by však odebrala možnost zvolení celé kategorie, pro větší customizaci by bylo potřeba nevyužívat prvek na základě externí knihovny ale vyvinout si vlastní.

- Scénář 3

  Průběh: Odebrání satelitu bylo bezproblémové, stejně jako zobrazení celé kategorie díky předchozím zkušenostem. Vyhledávání bylo pro všechny uživatele intuitivní, nicméně byli zaražení výchozím textem „Search category . . . “ a očekávali že se dá vyhledat pouze název kategorie.

  Poznatky: Pro vyjasnění je nutné změnit výchozí text ve vyhledávacím poli například na „Search by satellite name . . . “.

- Scénář 4

  Průběh: Naskenování globusu bylo u některých uživatelů problémem, nicméně se jednalo o problém spojený s nedostatečným osvětlením, se zvýšením intenzity osvětlení se detekce zlepšila. Zobrazení pozemních segmentů probíhalo bez obtíží a všichni uživatelé následně zkusili intuitivně globusem otočit pro zobrazení prvků pozemního segmentu na opačné straně Země. Detekce rotace fungovala ve všech případech, někdy však se zpožděním což participanty vedlo k přílišnému otočení globusu, protože sledovali pouze globus virtuální.

  Poznatky: Pro použití aplikace je potřeba dostatečného osvětlení a rotace globusu musí být prováděna o něco pomalejším tempem než by člověk běžně zvolil.

- Scénář 5

  Průběh: Většina participantů se pokusila informace zobrazit kliknutím na objekt satelitu, několik z nich se však tyto informace vydalo hledat do výběru samotného satelitu. Problémem však u většiny byla detekce kliknutí na zmíněný objekt satelitu, kdy nebyla příliš přesná a bylo potřeba kliknutí opakovat popřípadě na doporučení vývojáře provádět kliknutí pomaleji.

Poznatky: Bylo by vhodné zvětšit oblast pro zobrazení dat satelitu pokud místem pro zobrazení těchto informací bude právě sám satelit. Popřípadě by se dalo modální okno s informacemi zobrazovat na základě klepnutí na zvolené prvky pod select boxem - to by však vedlo k potřebě větší customizace Multi Select komponentu kde by bylo potřeba vytvořit vlastní zobrazení vybraných satelitů.

- Scénář 6

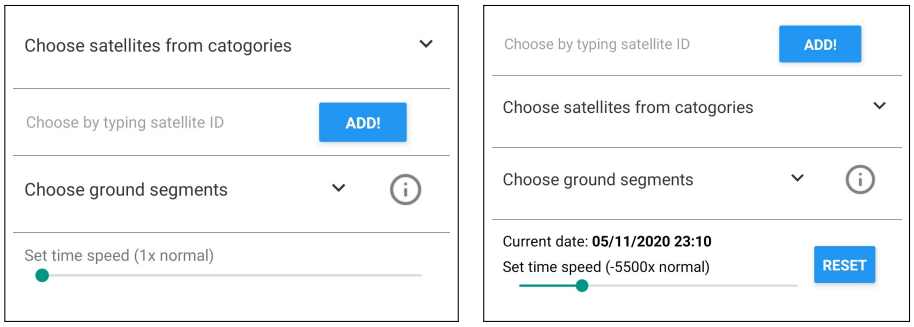
Průběh: Při závěrečném úkolu nenastaly žádné problémy, převážně díky tomu že tento úkol byl složen z kroků prováděných v předchozích úkolech. Jediným přidaným prvkem bylo zrychlení času, které je však dle reakcí uživatelů dosti intuitivní.

Poznatky: Jedním participantem bylo zmíněno, že by bylo vhodné mít u posuvníku času uvedeno v jakém času se aktuálně aplikace nachází a také zde mít možnost posouvat čas do minulosti.

5.3 Úpravy na základě testování

Kontrola času V aplikaci byla možnost zrychlení času pro vizualizaci i v jiném než aktuálním čase. Jak bylo ale jedním uživatelem zmíněno nikde nebyla možnost zobrazení času ve kterém se aplikace právě nachází. Zároveň s tím chyběla i možnost pro návrat v čase zpět aniž by bylo nutné restartovat celou aplikaci a vizualizaci znovu nastavit.

Obě tyto funkce byly přidány do výsuvného panelu a tak jsou veškeré funkcionality spojené s časem přehledně na jednom místě. Díky tomu, že rozsah slider komponentu pro kontrolu rychlosti je $(-10000, 10000)$ bylo by složité přesně trefit hodnotu 1 pro návrat ke skutečné rychlosti času. Tuto možnost tedy zastřešuje tlačítko *Reset*. Vzhled celé této nové části je vidět na obrázku 5.2.



Obrázek 5.2: Porovnání původní a upravené verze výsuvného panelu

Výchozí texty Na základě zpětné vazby uživatelů došlo k několika změnám textů v aplikaci. Všechny tyto změny se týkají Sectioned Multi Select

komponent pro výběr satelitů a pozemních prvků. Tvůrci tohoto komponentu předvídali různorodé použití a tak nebylo třeba na základě tohoto komponentu vytvářet vlastní, ale pouze pomocí předání textů v props jednotlivých komponent požadované texty změnit.

Konkrétně se jedná o změnu výchozího textu celého výběru satelitů, kde byl text *Choose from categories* nahrazen výstižnějším textem *Choose satellites from categories*. Díky této změně si bude uživatel více jistý obsahem každého selectu a nebude docházet ke zbytečnému rozevírání nabídky, jen kvůli zjištění že se zde vybírají satelity a ne prvky pozemního segmentu GPS.

Druhá změna je společná pro oba zmíněné selecty a spočívá ve změně výchozího textu vyhledávacího pole. Původní text *Search categories ...* mohl evokovat možnost vyhledávat pouze název kategorie. Samotný select však umožňuje vyhledání konkrétního prvku podle jména. Tato možnost je tedy znázorněna textem *Search by name ...* v záhlaví.

Pořadí ve výsuvném panelu Ve výsuvném panelu byla přesunuta možnost přidání satelitu podle jeho ID na první místo. Bylo tak učiněno na základě několikerého kliknutí účastníků testování na tlačítko *ADD*, aniž by si konkrétní uživatel přečetl text selectu nad ním. Je zvykem dávat nejdůležitější věc na první místo a z toho důvodu tam byla přesunuta i možnost volby satelitu dle ID, které je právě hlavním identifikátorem jednotlivých satelitů.

5.4 Shrnutí testování

Závěrečné testování aplikace se ukázalo jako velmi přínosné, protože odhalilo několik chyb, které v průběhu vývoje odhaleny nebyly i přes jejich triviálnost a očekávatelnost viz přidání satelitu s prázdným ID. Zároveň od uživatelů přišla zpětná vazba ohledně používání aplikace a nutnosti upravit samotné uživatelské rozhraní. Také byly také vzneseny požadavky na nové funkce, které byly buď naimplementovány jako výsledek testování nebo v případě větších funkcí je možnost implementace ponechána na vývoj po odevzdání této práce.

Na základě testování také byly odstraněny chyby, kde tou hlavní bylo zapomenutí kontroly vstupu pro přidání satelitu dle jeho ID. Bylo možné přidat satelit s prázdným ID což mělo za následek pád aplikace při dotazu na SpaceTrack API. Dalšími opravenými chybami bylo několik překlepů v textu.

Závěrečné testování odhalilo pouze několik chyb, což přisuzuji vývoji dle principů UCD a tedy neustálému testování v průběhu vývoje, díky čemuž byla většina chyb odhalena a opravena ještě před závěrečným testováním.

Kapitola 6

Závěr

Cílem této práce bylo prostudování frameworků pro vývoj multiplatformních mobilních aplikací s využitím rozšířené reality a následný vývoj aplikace pro vizualizaci aktuálních poloh umělých satelitů Země a pozemního segmentu GPS na základě dat z příslušných webových služeb.

Čtenář byl seznámen s možnostmi multiplatformního vývoje mobilních AR aplikací a s problematikou výpočtů pozic satelitů společně s potřebnými daty a jejich poskytovateli. Na základě těchto poznatků byla vytvořena mobilní aplikace, která vizualizaci v rozšířené realitě umožňuje. Tato aplikace byla testována jak v samotném průběhu vývoje na základě User-Centered Designu tak i po ukončení vývoje Usability testováním s cílovými uživateli. Na základě zpětné vazby z testování byla aplikace upravena tak aby lépe reflektovala uživatelské potřeby.

Samotná aplikace je ve stavu ve kterém se dá používat, nicméně je zde spousta možností kam její vývoj směřovat. V první řadě je to kompilace a testování na platformě iOS. Z funkcí, které by se daly přidat, je to například překlad aplikace do několika jazyků, komunikace se SATCAT katalogem pro zobrazení většího množství informací o satelitech nebo například rozšíření specializovaných modelů satelitů. Dále by bylo možné přidat možnost nasnímání libovolného globusu, čímž by se odbouralo omezení na vlastnictví jednoho konkrétního typu. Jediným problémem při budoucím vývoji by mohlo být ukončení vývoje Viro React, ke kterému se možná schyluje, jelikož dle GitHubu vývojáři neodpovídají na dotazy zhruba od ledna 2020 a je tedy možné, že tato technologie již nebude dále podporována.

Příloha A

Bibliografie

[1] *ARCore supported devices*. 2020. URL: https://developers.google.com/ar/discover/supported-devices (cit. 27. 04. 2020).

[2] *Augmented Reality Market : Global Industry Analysis, Size, Share, Growth, Trends, and Forecast, 2018-2025*. Zář. 2019. URL: https://www.americanewshour.com/2019/09/27/augmented-reality-market-global-industry-analysis-size-share-growth-trends-and-forecast-2018-2025/65574/ (cit. 13. 04. 2020).

[3] *AugmentedImage.TrackingMethod Documentation*. Květ. 2019. URL: https://developers.google.com/ar/reference/java/arcore/reference/com/google/ar/core/AugmentedImage.TrackingMethod (cit. 03. 05. 2020).

[4] Filippo Bergamasco, Andrea Albarelli a Andrea Torsello. „Pi-Tag: a fast image-space marker design based on projective invariants“. In: *Machine Vision and Applications* 24 (2012), s. 1295–1310.

[5] Mark Billinghurst. „Augmented reality in education“. In: *New horizons for learning* 12.5 (2002), s. 1–5.

[6] Amanda Edwards-Stewart, Tim Hoyt a Greg Reger. „Classifying different types of augmented reality technology“. In: *Annual Review of CyberTherapy and Telemedicine* 14 (2016), s. 199–202.

[7] Nat Friedman. *New year, new GitHub: Announcing unlimited free private repos and unified Enterprise offering*. Led. 2019. URL: https://github.blog/2019-01-07-new-year-new-github/ (cit. 01. 05. 2020).

[8] *GitHub Issues: Viro React - OnClick detected by wrong object*. Říj. 2019. URL: https://github.com/viromedia/viro/issues/744 (cit. 01. 05. 2020).

[9] *GPS Control Segment*. Lis. 2018. URL: https://www.gps.gov/systems/gps/control/ (cit. 04. 05. 2020).

[10] Mohinder S Grewal, Lawrence R Weill a Angus P Andrews. *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons, 2007.

[11] Dieter Holger. *The Australian Air Force Is Now Testing the Microsoft HoloLens*. Led. 2017. URL: https://vrscout.com/news/the-australian-air-force-is-now-testing-the-microsoft-hololens/ (cit. 24. 04. 2020).

[12] Lumír Honzík. *Minislovníček: Jarní bod*. Pros. 2010. URL: http://www.hvezdarnaplzen.cz/2010/12/27/minislovnicek-jarni-bod/ (cit. 23. 04. 2020).

[13] Christopher Chedeau. *A Deep Dive into React Native*. Červ. 2015. URL: http://www.reactnative.com/a-deep-dive-into-react-native/ (cit. 01. 05. 2020).

[14] *IKEA launches IKEA Place*. Srp. 2017. URL: https://newsroom.inter.ikea.com/news/ikea-launches-ikea-place--a-new-app-that-allows-people-to-virtually-place-furniture-in-their-home/s/f5f003d7-fcba-4155-ba17-5a89b4a2bd11 (cit. 23. 04. 2020).

[15] *Integrating with React Native Projects*. 2020. URL: https://docs.viromedia.com/docs/integrating-with-react-native-projects (cit. 15. 05. 2020).

[16] *iOS ARKit Augmented Reality Support*. 2020. URL: https://www.apple.com/lae/ios/augmented-reality/ (cit. 27. 04. 2020).

[17] Shubheksha Jalan. *An introduction to how JavaScript package managers work*. Říj. 2016. URL: https://www.freecodecamp.org/news/javascript-package-managers-101-9afd926add0a/ (cit. 02. 05. 2020).

[18] Dr. T.S. Kelso. „More Frequently Asked Questions“. In: (břez. 1998). URL: https://celestrak.com/columns/v04n05/#FAQ07.

[19] Camila Kohles. *Augmented Reality Experiences for Museums: Three Success Stories*. Zář. 2019. URL: https://www.wikitude.com/blog-ar-apps-for-museums-three-success-stories/ (cit. 08. 04. 2020).

[20] Greg Kumparak. *Hands-on with an Alpha build of Google Maps Augmented Reality mode*. Ún. 2019. URL: https://techcrunch.com/2019/02/11/hands-on-with-an-alpha-build-of-google-maps-augmented-reality-mode/ (cit. 30. 04. 2020).

[21] Ben Lang. *Analysis: Monthly-connected VR Headsets on Steam Pass 1 Million Milestone*. Červ. 2019. URL: https://www.roadtovr.com/monthly-connected-vr-headsets-steam-1-million-milestone/ (cit. 02. 05. 2020).

[22] Mark Livingston et al. „An augmented reality system for military operations in urban terrain“. In: (2008).

[23] Tricia McKinnon. *10 of the Best Augmented Reality Shopping Apps to Try Today*. Květ. 2020. URL: https://www.indigo9digital.com/blog/how-six-leading-retailers-use-augmented-reality-apps-to-disrupt-the-shopping-experience (cit. 10. 05. 2020).

[24] Nicholas Z Miura. „Comparison and design of Simplified General Perturbation Models (SGP4) and code for NASA Johnson Space Center, Orbital debris program office“. In: (2009).

[25] *Mobile Operating System Market Share Worldwide*. 2020. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (cit. 10. 04. 2020).

[26] *Multiplatform support of Unity Engine*. 2020. URL: https://unity3d.com/unity/features/multiplatform (cit. 23. 04. 2020).

[27] Paweł Nowacki a Marek Woda. „Capabilities of ARCore and ARKit Platforms for AR/VR Applications“. In: 2020, s. 358–370.

[28] *Optimizing Target Detection and Tracking Stability*. 2020. URL: https://library.vuforia.com/articles/Solution/Optimizing-Target-Detection-and-Tracking-Stability.html#local (cit. 03. 05. 2020).

[29] *Orbitrack - Apps on Google Play*. 2020. URL: https://play.google.com/store/apps/details?id=com.southernstars.orbitrack&hl=en_US (cit. 30. 04. 2020).

[30] NC Patro. *Choose the best - Native App vs Hybrid App*. Květ. 2018. URL: https://codeburst.io/native-app-or-hybrid-app-ca08e460df9 (cit. 07. 04. 2020).

[31] Daniel Pražák. *Recenze Neobear AR Globe*. Dub. 2018. URL: https://www.letemsvetemapplem.eu/2018/04/15/recenze-neobear-ar-globe-kdyz-vam-rozsirena-realita-predstavi-cely-svet/ (cit. 30. 04. 2020).

[32] Louis Rosenberg. „The Use of Virtual Fixtures as Perceptual Overlays to Enhance Operator Performance in Remote Environments“. In: (1992), s. 52.

[33] *SATCAT BoxScore*. 2020. URL: https://celestrak.com/satcat/boxscore.php (cit. 29. 04. 2020).

[34] *Satellite AR - A New Satellite Viewer*. 2019. URL: https://agi.com/satellite-ar (cit. 30. 04. 2020).

[35] *Scanning and Detecting 3D Objects*. 2020. URL: https://developer.apple.com/documentation/arkit/scanning_and_detecting_3d_objects?language=objc (cit. 03. 05. 2020).

[36] *Setting up the development environment · React Native*. 2020. URL: https://reactnative.dev/docs/environment-setup (cit. 15. 05. 2020).

[37] *Space-Track API Use Guidelines*. 2020. URL: https://www.space-track.org/documentation#/api (cit. 07. 05. 2020).

[38] *Stack Overflow Developer Survey: Development Environments and Tools*. 2019. URL: https://insights.stackoverflow.com/survey/2019#development-environments-and-tools (cit. 01. 05. 2020).

[39] *The State of the Octoverse*. 2020. URL: https://octoverse.github.com/#top-and-trending-projects (cit. 01. 05. 2020).

[40] Arthur Toga Tyler Ard. *ScholAR: Augmented Reality for Scholarly Research & Communication*. 2019. URL: https://www.ini.usc.edu/ScholAR/download.html (cit. 30. 04. 2020).

[41] *Usability Testing*. Lis. 2013. URL: https://www.usability.gov/how-to-and-tools/methods/usability-testing.html (cit. 08. 05. 2020).

[42] *User-Centered Design Basics*. Dub. 2017. URL: https://www.usability.gov/what-and-why/user-centered-design.html (cit. 08. 05. 2020).

[43] David Vallado et al. „Revisiting Spacetrack Report No.3: Rev". In: 2006.

[44] *Viro React Overview*. 2020. URL: https://docs.viromedia.com/docs/viro-platform-overview (cit. 23. 04. 2020).

[45] *What are the best Cross-Platform Mobile Development Tools?* 2020. URL: https://stackshare.io/cross-platform-mobile-development (cit. 03. 05. 2020).

[46] Tadeu Zagallo. *Bridging in React Native*. Říj. 2015. URL: https://tadeuzagallo.com/blog/react-native-bridge/ (cit. 01. 05. 2020).

Příloha B

Seznam zkratek

Zkratka	Význam
2D	dvoudimenzionální
3D	trojdimenzionální
API	Application Programming Interface
APK	Android Application Package
AR	Augmented Reality
CLI	Command Line Interface
CSS	Cascading Style Sheets
DOF	Degrees of Freedom
DOM	Document Object Model
ECEF	Earth-Centered, Earth-Fixed
ECI	Earth-Centered Inertial
ENU	East, North, Up
ES6	ECMAScript 6
FBX	FilmBoX
glTF	Graphics Library Transmission Format
GPS	Global Positioning System
HTML	Hypertext Markup Language
HUD	Head-Up Display
ID	Identifikátor
IDE	Integrated Development Environment
ISS	International Space Station
JS	JavaScript
JSON	JavaScript Object Notation
JSX	JavaScript XML
MIT	Massachusetts Institute of Technology
npm	Node.js Package Manager
QR	Quick Response
REST	Representational State Transfer
rnpm	React Native Package Manager
SATCAT	Satellite Catalog

Zkratka	Význam
SDK	Software Development Kit
SDP	Simplified Deep Space Perturbations
SGP	Simplified General Perturbations
SLAM	Simultaneous localization and mapping
SSN	Space Surveillance Network
TLE	Two-Line Element set
UCD	User-Centered Design
UI	User Interface
URL	Uniform Resource Locator
USD	United States Dollar
UWP	Universal Windows Platform
VR	Virtual Reality
XAML	Extensible Application Markup Language
XML	eXtensible Markup Language

Příloha C

Návod k zprovoznění

C.1 Instalace Android .apk balíčku

Pro samotnou instalaci je potřeba mít v telefonu povolenou instalaci z neznámých zdrojů. U některých výrobců je toto nastavení obecné v záložce Bezpečnost, jiní to řeší povolením této možnosti jednotlivým aplikacím např. průzkumníku souborů.

1. Zkopírování .apk souboru do úložiště telefonu
2. Lokalizace souboru v telefonu pomocí správce souborů
3. Spuštění instalace pomocí klepnutí na soubor

C.2 Zprovoznění vývojářské verze

Tento návod se týká striktně kombinace platforem Windows + Android, která byla použita při vývoji. Pro ostatní platformy bude postup podobný nicméně v některých krocích se může mírně lišit. V takovém případě je potřeba následovat dokumentaci jednotlivých technologií.

1. Instalace závislostí React Native

   Následujte návod na `https://reactnative.dev/docs/environment-setup` v záložce React Native CLI Quickstart a proveďte instalaci potřebných závislostí. Ve zkratce se jedná o Node.js, které zároveň přináší npm; Python a Java JDK. I když je možné vývoj provádět v libovolném editoru je potřebná instalace Android Studia kvůli Android SDK. Následně je ještě třeba nastavit proměnné prostředí JAVA_HOME a ANDROID_HOME. Všechny tyto kroky jsou detailně popsány ve zmíněném návodu.

2. Přesunutí zdrojových kódů

   Přesuňte si kódy z CD ze složky development do libovolného umístění ve vašem počítači.

3. Otevření cmd ve složce s kódy

4. Instalace závislostí

   Zadáním příkazu `npm install` se provede instalace všech závislostí projektu potřebných ke spuštění.

5. Migrace na AndroidX

   Zadáním příkazu `npx jetify` se provede převedení nezmigrovaných závislostí právě na AndroidX. Více detailů v kapitole 4.2.

6. Připojení zařízení

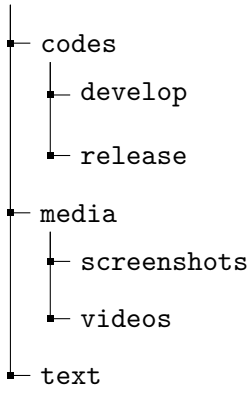
   Připojte vývojové zařízení k počítači pomocí USB kabelu. Na zařízení je nutné mít povolený Vývojářský režim. Jeho spuštění je možné po sedmi poklepání na Číslo sestavení v nastavení telefonu.

7. Spuštění aplikace

   Příkazem `npx react-native run-android --variant=arDebug` dojde ke spuštění Metro bundleru, instalaci vývojářské verze aplikace na telefon a k jejímu spuštění. Do vývojářského menu je následně možné se dostat zatřesením telefonu.

Příloha D

Obsah přiloženého CD



develop Zdrojové kódy React Native projektu.

release Zkompilovaná aplikace pro Android a obrázek planárního cíle k vytištění na papír formátu A4.

screenshots Snímky z používání aplikace.

videos Videa z průběhu používání aplikace.

text Zdrojové kódy textu této práce v jazyce LaTeX.