BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2019
http://excel.fit.vutbr.cz

Rozšírená realita na iOS – zobrazenie prvkov z mapy v teréne

Bc. Adam Bezák

Abstrakt
Táto práca si dáva za cieľ zlepšiť užívateľovi orientáciu v teréne pomocou jeho smartphonu a rozšírenej reality. Spojením týchto dvoch prvkov a bezplatnej databáze geografických dát OpenStreetMap je možné zobraziť okolité prvky z máp alebo prípadne zobraziť doplňujúce informácie o okolitých objektoch. Výsledná technológia je použitá v naimplementovanej iOS aplikácii s názvom Water Radar. Táto aplikácia zobrazuje okolité vodné toky (alebo iné vodné objekty) vzhľadom na GPS polohu užívateľa. Aplikácia využíva taktiež Google Elevation API aby zobrazené vodné toky kopírovali reliéf krajiny.
V článku je popísaný spôsob získania a spracovania dát vhodných na zobrazenie. Taktiež je vysvetlená technika zisťovania, čo najpresnejšej lokácie užívateľa vďaka ktorej sa odfiltrujú nepresné GPS polohy. Dôraz implementácie je kladený najmä na jednoduché a zrozumiteľné užívateľské rozhranie, kde sa musia napríklad vzdialené prvky zväčšiť alebo odfiltrovať.

Kľúčové slová: rozšírená realita — iOS — mobilná — aplikácia — ARKit — SceneKit — OpenStreetMaps

Priložené materiály: *N/A*

*xbezak01@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

Cieľom tejto práce je využiť rozšírenú realitu k zobrazeniu prvkov z mapy. Pojem rozšírená realita sa do podvedomia ľudí dostáva len krátku dobu a smeruje každú oblasť v ktorej sa používa rapídnym krokom vpred [1]. Jedným z možných využití je aj pri navigácii alebo skúmaní neznámeho terénu. Bez použitia rozšírenej reality musí užívateľ často hľadať informácie v zložito spracovaných mapách, ktoré chcú zobraziť čo najviac informácií na jednom mieste. Pre niektorých je to prínosné, iní by radšej použili aplikáciu v ktorej hneď po spustení vidia to, čo chcú vidieť. Z týchto dôvodov som sa rozhodol vytvoriť technológiu pre AR, ktorá bude sťahovať verejne dostupné dáta okolitej krajiny z databáze **OpenStreetMap**[1] a transformovať ich na vhodné 3D objekty. Tieto 3D objekty budú následne vložené do priestoru na čo najpresnejšiu pozíciu. Aby vložené 3D objekty reflektovali realitu čo najlepšie, tak je potrebné zistiť čo najpresnejšiu polohu užívateľa. Taktiež objekty musia vhodne reprezentovať skutočný objekt. To znamená, že v prípade zobrazenia nejakej cesty sa musí vytvoriť vhodná 3D úsečka. V prípade zobrazenia jedného statického objektu sa zobrazí len samotný bod s vhodnou ikonou ktorá ho vystihuje.

[1] https://www.openstreetmap.org

2. Rešerš existujúcich AR aplikácií

Medzi hlavné aspekty patrí aj analýza už vyvinutých aplikácií. Zameral som sa len na platformu iOS a na bezplatné aplikácie, ktoré sú pravidelne udržiavané a aktualizované. Aplikácií, ktoré využívajú rozšírenú realitu je pomerne dosť avšak väčšina z nich zobrazuje len menej významné objekty v bezprostrednej blízkosti užívateľa alebo sa jedná o jednoduchú hru. Jednou z aplikácií, ktorá pracuje s mapou je **ViewRanger**. Slúži na zobrazenie turistických trás a bodov záujmov v okolí užívateľa. V platenom AR móde (obr. 1 vľavo) zobrazuje okolité pohoria a ich najbližšiu trasu k nim. Okrem týchto možností ponúka veľa iných funkcionalít, ktoré nesúvisia s rozšírenou realitou. Na-

príklad je možné vyhľadávať okolité turistické trasy, vytvárať nové a tie následne zdielať s ostatnými užívateľmi. Druhou vybranou aplikáciou je **ARCity** od vývojového štúdia Blippar. Táto aplikácia sa zameriava na rozdiel od aplikácie ViewRanger len na tri druhy funkcionalít:

AR navigácia – vizualizácia cesty do cieľovej destinácie v rozšírenej realite pomocou 3D šípok.

Rozšírený kontext mapy – Prináša ďalšie informácie o okolitých objektoch (obr. 1 vpravo). Tým sa myslí mená ulíc, názvy budov alebo lokálne body zájmu.

Urban visual positioning - Rozpoznávanie, pozicovanie a priama informácia vďaka počítačovému videniu. Momentálne funguje len vo vybraných mestách (San Francisco, Londýn a Mountain View). Jedná sa o systém, ktorý priamo využíva počítačové videnie a vďaka nemu precízne spresňuje lokáciu užívateľa.

Po spustení aplikácie sa zobrazí mapa s dotazom zadať presnú polohu užívateľa na mape. To je kvôli čo najpresnejšej kalibrácií lokácie keďže presnosť GPS v meste sa pohybuje až na hranici 16m, čo môže byť pre takúto aplikáciu veľmi nevhodné. Po zadaní lokácie sa na mape zobrazia okolité budovy a body zájmu. Táto aplikácia taktiež využíva databázu OpenStreetMaps. Veľkou výhodou je jednoduchosť aplikácie. Okrem mapy a prezeraní v rozšírenej realite ponúka len drobné nastavenia. Nevýhodou je, že neponúka aspoň základné filtrovanie zobrazených objektov. Taktiež aplikácia zobrazuje primárne len cesty prvej triedy.

3. Návrh systému

3.1 Požiadavky na systém

Cieľom práce je vytvoriť užívateľsky prívetivý systém, ktorý bude vhodne vytvárať mapové dáta a tie zobrazovať na čo najpresnejšiu pozíciu vzhľadom k realite. Tento systém by mal efektívne dáta sťahovať a následne ich transformovať na 3D objekty, ktoré zodpovedajú typu dát. Vytvorené objekty je potrebné vložiť do scény na čo najpresnejšiu pozíciu. Je nutné sa zamyslieť nad relatívnou vzdialenosťou od kamery (polohy užívateľa) k zobrazenému objektu. Pokiaľ sa objekt vloží príliš ďaleko, tak bude veľmi malý a takmer neviditeľný. Taktiež by bolo vhodné keby zobrazené body reflektovali reliéf krajiny. Pokiaľ bude užívateľ stáť na vyvýšenom mieste a smerovať aplikáciu na krajinu s nižšou nadmorskou výškou tak by sa tento rozdiel mal viditeľne prejaviť a zobrazené objekty by sa nemali vznášať vo vzduchu.

Obrázok 1. Na ľavom obrázku je znázornená aplikácia ViewRanger, ktorá za príplatok prémiového účtu ponúka zobrazenie okolitých pohorí a smer trasy. Táto trasa sa zobrazuje ako 3D šípky smerujúce od užívateľa priamo k najbližšiemu pohoriu. Na pravo je zobrazená aplikácia ARCity. Tá zobrazuje presne všetky okolité body zájmu či už na dynamickej mape alebo v rozšírenej realite. Dokáže zobraziť názvy ulíc, trajektórie ciest a názvy okolitých objektov.

Presná poloha užívateľa je v tomto type aplikácie najdôležitejším bodom. Počas prvotného testovania som experimentálne zistil, že chyba v určení polohy v meste (okolie budov) môže dosiahnuť až hodnotu 64 metrov. Takáto nepresná hodnota môže rapídne ovplyvniť užívateľov zážitok z rozšírenej reality. Z tohto dôvodu by bolo vhodné, keby v aplikácií bola možnosť si zvoliť statickú lokáciu z mapy.

Z hľadiska výkonu aplikácie a na základe tzv. AR Human Interface Guidelines priamo od spoločnosti Apple sa odporúča nezobrazovať užívateľovi príliš veľa objektov v jednej scéne [2]. Preto by mala aplikácia podporovať aspoň jedno-úrovňový filter objektov. V tomto filtri by si mohol užívateľ jednoducho zvoliť aké druhy objektov ho zaujímajú.

Dôležitým aspektom je aj samotné testovanie aplikácie. To je v tomto druhu aplikácie, ktorá pracuje s rozšírenou realitou a okolitou krajinou celkom problematické. Na priebežné testovanie pri vývoji je za určitých okolností prijateľné testovanie z domu avšak pre najlepšie výsledky je potrebné ísť von do prírody a zistiť, ako systém funguje za takých podmienok, za akých je predpoklad jeho použitia.

3.2 Technický popis použitých systémov

Celý systém som cielil na operačný systém iOS, ktorý od verzie iOS 11 ponúka oficiálnu knižnicu rozšírenej

reality s názvom **ARKit**. ARKit umožňuje sledovanie relatívnej pozície zariadenia v reálnom 3D priestore. Na to využíva aj tzv. Visual Inertial Odometry ktorá spája údaje obrazu kamery s údajmi z gyroskopu a akcelerometru. Nevýhodou je, že ARKit neponúka možnosť pracovať s týmito údajmi avšak ich zabaľuje do SDK a programátorovi sa poskytne už len určitá reprezentácia scény. Po detekcii priestoru sa umiestni počiatočný súradnicový systém na základe polohy telefónu. Tento systém obsahuje osy X, Y a Z, ktorých hodnoty sa menia pri každej zmene polohy telefónu. Základnou jednotkou každej osi je 1 meter. Mimo ARKitu, ktorý sa stará len o spracovanie nasnímanej reálnej scény, je potrebné použiť vykreslovací framework vďaka ktorému je možné vkladať do scény objekty. Pre prácu s 3D objektami sa používa **SceneKit**. SceneKit obsahuje graf scény na základe koreňového objektu. Táto scéna obsahuje hierarchiu uzlov a atribútov, ktoré spolu reprezentujú vizuálne elementy. Každý z vložených objektov je popísaný svojou pozíciou v 3D priestore, teda sám o sebe nemá žiadny viditeľný obsah, avšak je možné pridať geometriu, osvetlenie alebo iný vizualizačný obsah.

Zdrojom pre mapové dáta je databáza **OpenStreetMap**. Táto databáza obsahuje upravenú mapu celého sveta, ktorú zhotovili dobrovoľníci. Je vhodná na použitie hlavne vďaka tomu, že je publikovaná s open-content licenciou. To znamená, že je možné pristúpiť ku všetkým dátam úplne zdarma [3]. Nevýhodou je, že OpenStreetMap neobsahuje dáta o nadmorskej výške. Ako zdrojom týchto informácií sa používa **Google Elevation API**. Momentálne všetky verejné API od firmy Google využívajú systém pay-as-you-go. To znamená, že sa platí určitá suma za každý požiadavok. Avšak každý mesiac Google poskytne kredit vo výške 200 dolárov a pokiaľ sa tento kredit neprečerpá tak užívateľ neplatí nič. Na druhú stranu je povolené maximálne len 512 lokácií v jednom požiadavku a maximálne 100 požiadavkov za sekundu. Tieto obmedzenia bez problémov postačia k bezplatnému používaniu vo výslednom systéme.

4. Realizácia systému

V jednotlivých podkapitolách je vysvetlený princíp funkčnosti všetkých definovaných požiadavkov na systém z kapitoly 3.1.

4.1 Spresňovanie lokácie GPS

Ako bolo písané vyššie, tak čo najpresnejšia poloha užívateľa je pre tento systém najdôležitejšia. Táto kalkulácia je založená na základe idei popísanej v článku od Andrewa Harta [4]. Aplikácia zachytáva zistenú GPS polohu užívateľa a priebežne si ju ukladá do internej štruktúry aj s aktuálnou pozíciou kamery v scéne. Pozícia kamery v scéne sa ukladá ako 3-rozmerný vektor. Takto vzniká kolekcia štruktúr, ktorá je vstupom v metóde na nájdenie najpresnejšieho odhadu poslednej lokácie a pozície kamery v scéne. V tejto metóde sa zoradia všetky zaznamenané lokácie podľa presnosti a časového razítka. V prípade posunu kamery v scéne sa zistí rozdiel v metroch medzi posledným najpresnejším odhadom pozície vektoru kamery podľa vzorcov:

$$\Delta x = Bx_1 - Ax_2 \quad (1)$$

$$\Delta z = Az_2 - Bz_1 \quad (2)$$

$$\Delta y = By_1 - Ay_2 \quad (3)$$

kde:

A = aktuálny vektor kamery v scéne
B = posledný najpresnejší odhad vektoru
x = zemepisná dĺžka
z = zemepisná šírka
y = nadmorská výška

Z najpresnejšieho odhadu lokácie, počiatočného azimutu a vypočítanej vzdialenosti sa zistí výsledná lokácia [5] podľa vzorcov:

$$\varphi_2 = \arcsin\left(sin\varphi_1 \cdot \cos\delta + \cos\varphi_1 \cdot \sin\delta \cdot \cos\theta\right) \quad (4)$$

$$\omega = \arctan(\sin\theta \cdot \sin\delta \cdot \cos\varphi_1, \cos\delta - \sin\varphi_1 \cdot \sin\varphi_2) \quad (5)$$

$$\lambda_2 = \lambda_1 + \omega, \quad (6)$$

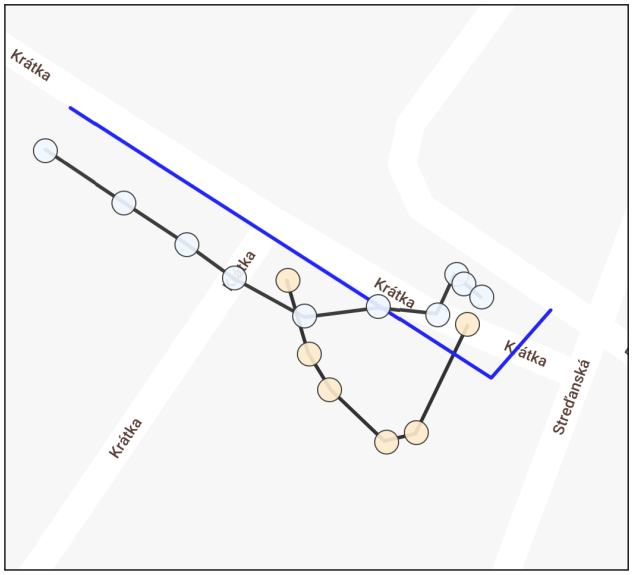
kde:

φ = zemepisná šírka
λ = zemepisná dĺžka
θ = azimut
d = vzdialenosť
R = polomer zeme
δ = uhlová vzdialenosť vypočítaná ako d/R

Azimut pre výpočet zemepisnej šírky je 0 stupňov a pre výpočet zemepisnej dĺžky je 90 stupňov. Teda pokiaľ užívateľ zmenil polohu v scéne podľa kladnej osy Z (na juh) o 5 metrov tak z poslednej pozície vypočítame posun v kladnej časti osy Z o 5 metrov.

Takýmto spôsobom je možné odfiltrovať a eliminovať nepresné GPS polohy, ktoré sú často získavané z lokalizačných služieb telefónu najmä v meste v okolí vyšších budov. Druhým spôsobom získania lokácie užívateľa ja možnosť použiť statickú mapu, kde si užívateľ sám zvolí svoju polohu na mape.

Na obrázku 2 je zobrazené testovanie implementácie spresňovania lokácie GPS užívateľa v mestskom prostredí.



Obrázok 2. Experimentálne testovanie metódy spresňovania lokácie GPS užívateľa.

Modrá čiara predstavuje reálnu cestu užívateľa pri spustenej aplikácii. Žlté body znázorňujú zistenú GPS polohu z lokalizačných služieb telefónu a sivé body reprezentujú vypočítanú najpresnejšiu polohu užívateľa pomocou vyššie spomenutého algoritmu.

4.2 Získanie a spracovanie dát

Ako zdroj dát sa používa databáza **OpenStreetMap**. Ponúka verejné read-only API rozhranie, ktoré pracuje na štýle dotaz-odpoveď. Klient zašle na API dotaz a vráti sa mu odpoveď s odpovedajúcimi dátami. Ako vyhľadávací jazyk je možné použiť Overpass XML alebo Overpass Query Language. Toto API ponúka nespočetne veľa možností vyhľadávania a filtrovania dát, ktoré sú popísané v dokumentácii[2]. Základným princípom vyhľadávania je využitie dvojice kľúč a hodnota.

```
<osm-script output="json">
 <union>
  <query type="node">
   <has-kv k="amenity" regv="\b(fountain)\b"/>
   <around lat="LAT" lon="LON" radius="1000"/>
  </query>
  <query type="way">
   <has-kv k="waterway" regv="\b(river)\b"/>
   <around lat="LAT" lon="LON" radius="1000"/>
  </query>
  <recurse type="way-node" />
 </union>
 <print/>
</osm-script>
```

Ukážka 1. Ukážka často používaného dotazu v aplikácii Water Radar

Vo vzorovej ukážke 1 je možné vidieť najčastejšie používaný dotaz z aplikácie Water Radar. Ako typ dotazu sa používa príznak `node`, ktorý vráti jednotlivé statické body z mapy a príznak `way` vráti všetky cesty. Aby bolo možné cieliť len na určitú oblasť v okolí užívateľa tak sa používa element `around`, ktorému sa zadajú hodnoty zemepisnej šírky, zemepisnej dĺžky a rádius v metroch. Teda tento dotaz vráti všetky fontány a rieky v okolí 1000 metrov od zadanej polohy.

Server vracia odpovede v štandardnom XML alebo JSON formáte. Výsledný systém využíva už naprogramovanú knižnica v programovacom jazyku Swift s názvom **SwiftOverpass**[3]. Keď že tento wrapper ponúka len jednoduché dotazovanie na kľúč a hodnotu, bolo nutné ho prevziať (forknúť) a upraviť. V budúcnosti sa plánuje na túto úpravu vytvoriť pull request do originálneho git repozitára a poskytnúť tieto zmeny aj ostatným užívateľom, ktorý s touto knižnicou pracujú.

4.3 Vytvorenie 3D objektov

Stiahnuté dáta je potrebné pretransformovať do vnútorných štruktúr z ktorých sa budú jednoducho vytvárať vhodné 3D objekty. Súčasťou tejto transformácie je pre každú súradnicu zistenie jej nadmorskej výšky z Google Elevation API.

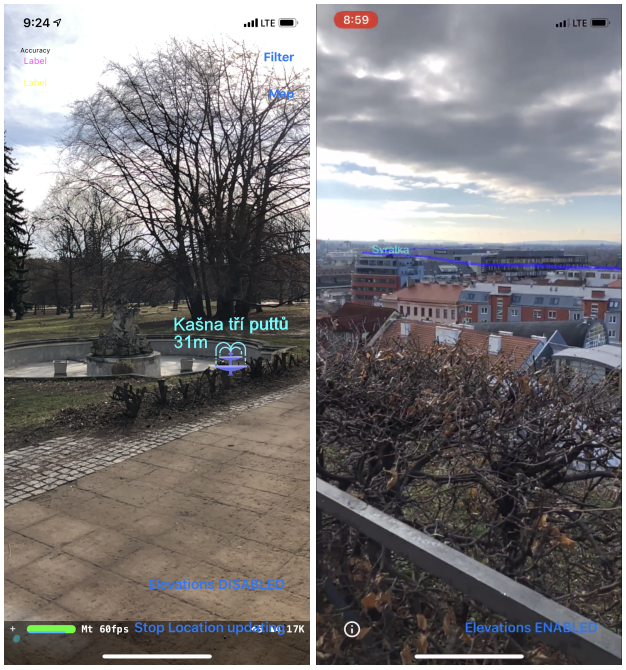
Pre každý statický bod na mape sa vytvorí objekt `LocationNode`, ktorému sa ako geometria nastaví 2D plocha a na túto geometriu sa aplikuje textúra vopred uloženého obrázku v aplikácií. Užívateľov by primárne zaujímala aktuálna vzdialenosť k bodom a z toho dôvodu sa nad každý vytvorený `LocationNode` objekt umiestni 3D text, ktorý zobrazuje aktuálnu vzdialenosť (viď obr. 3 vľavo). Pri pohybe užívateľa scénou sa táto vzdialenosť dynamicky mení.

Keďže cesty z databáze OpenStreetMap môžu byť rozdelené do viacerých častí tak je nutné spraviť pred vytvorením vhodných vnútorných štruktúr určité predspracovanie dát. Tým sa myslí, že všetky časti ciest sú spojené na základe ich mena a až potom je z nich vytvorená vhodná vnútorná štruktúra (viď obr. 3). Každá cesta obsahuje postupnosť lokácií, medzi ktorými sa vytvorí 3D objekt so 6 stranovou geometriou. Každá lokácia už obsahuje získanú nadmorskú výšku a na základe nej sa vypočíta uhol ktorý zvierajú dva susedné body. O tento zistený uhol sa 3D objekt otočí okolo svojej osy X. Na základe používateľovej vzdialenosti k jednotlivým cestám sa nastaví veľkosť objektu a tým sa eliminuje problém malej veľkosti vzdialených objektov.

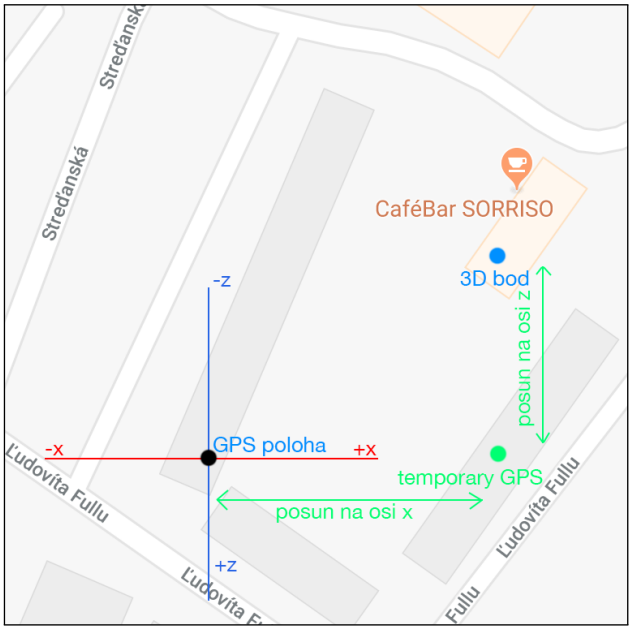
Týmto vytvoreným 3D objektom je potrebné nastaviť vhodnú pozíciu v scéne. Keďže extrémne vzdialené objekty sa nezobrazujú tak zaokrúhlenie zeme

[2] https://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide

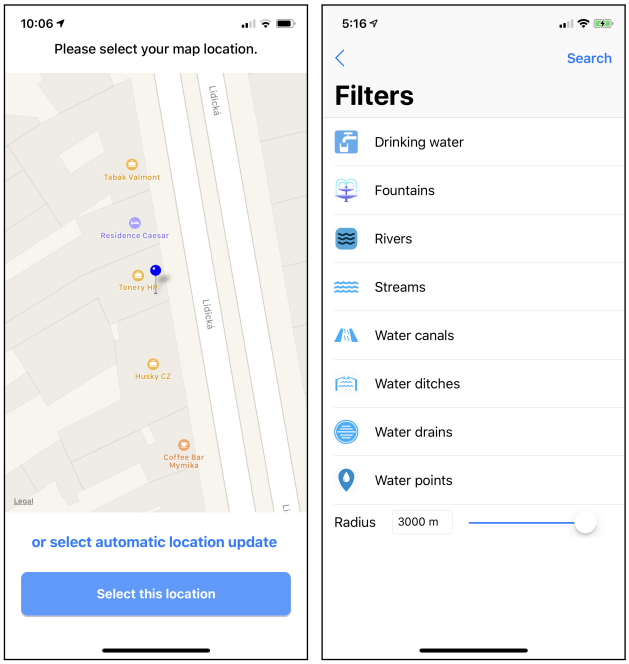
[3] https://github.com/holyturt/SwiftOverpassWrapper



Obrázok 3. Na ľavom obrázku je zobrazená statická lokácia z mapy. Na pravom obrázku je zobrazenie postupnosti GPS lokácií, ktoré reprezentujú jednu cestu.



Obrázok 4. Princíp výpočtu posunu pre vložené objekty v scéne.



Obrázok 5. Na ľavom obrázku je vstupná obrazovka do aplikácie. Užívateľ si môže vybrať svoju polohu ručne alebo použije automatickú detekciu polohy. Napravo je vidieť jedno-úrovňový filter.

sa môže zanedbať. Na obrázku 4 je znázornený princíp výpočtu posunu pre jednotlivé objekty, ktoré sú vložené v scéne. Zobrazené osy reprezentujú súradnicový 2D systém frameworku ARKit. Os X smeruje od severu na juh a znázorňuje zemepisnú šírku a os Z smeruje od východu na západ a znázorňuje zemepisnú dĺžku. Pre každý jednotlivý objekt sa pri výpočte vytvorí dočasná GPS poloha pomocou zemepisnej dĺžky GPS užívateľa a zemepisnej šírky objektu. Kedže ARKit pracuje v metroch tak sa táto dočasná poloha využije pre výpočet vzdialeností v scéne. Rozdiel vo vzdialenosti od GPS polohy užívateľa a dočasnej GPS polohy udáva posun na osi X. Obdobne rozdiel vo vzdialenosti od GPS polohy objektu a dočasnej GPS polohy udáva posun na osi Z. Pri výpočte týchto rozdielov je potrebné zistiť v ktorom kvadrante sa bod nachádza a na základe toho nastaviť záporný alebo kladný posun.

Počas testovania som zistil, že objekty ktoré boli vložené v scéne vo vzdialenosti väčšej ako 100m spôsobovali na testovacom zariadení (iPhone X) pokles FPS. Z tohto dôvodu je na všetky objekty, ktoré by mali byť vložené vo väčšej vzdialenosti aplikovaná transformácia zmeny mierky. Objekty sú zmenšené a posunuté bližšie avšak bez zmeny priestorového vnímania.

5. Výsledná aplikácia

Na ukážku funkčnosti systému bola naimplementovaná aplikácia s názvom **Water Radar**, ktorá sťahuje a zobrazuje všetky mapové prvky ohľadom vody. Dôvodom výberu tejto témy je absencia podobných aplikácií na Apple App store. Na základe užívateľského výskumu sa ako vstupná obrazovka zvolilo zobrazenie mapy s možnosťou ručného zadania lokácie alebo automatického zistenia najpresnejšej polohy (viď obr. 5 vľavo).

Po výbere polohy sa užívateľ dostane na hlavnú obrazovku kde je zobrazený priamy výstup z kamery. Na celej scéne je len jedno pomocné tlačidlo, ktoré

po stlačení zobrazí animovanú nápovedu ako používať aplikáciu. Potiahnutím prstom hore sa zobrazí znovu scéna, kde si užívateľ môže nastaviť presnejšiu polohu. Potiahnutím prstom vľavo sa zobrazí jednoduchý filter. V tomto filtri sú všetky možné hodnoty pomocou ktorých je možno stiahnuť dáta ohľadom vody z databáze OpenStreetMap. Taktiež sa tu dá nastaviť z akého veľkého rádiusu sa dajú dáta stiahnuť. Po vybratí nejakých filtrov a stlačení tlačidla Search sa zobrazí načítacia obrazovka. V prípade úspešného stiahnutia dát sa aplikujú všetky metódy popísané v kapitole 4.

Z hľadiska veľkosti objemu prenesených dát aplikácie spotrebuje na jeden komplexnejší dotaz okolo 15 kb dát. Jedná sa o predovšetkým o objemnú odpoveď z databáze OpenStreetMap, ktorá musí obsahovať všetky potrebné informácie k zobrazeniu bodov.

5.1 Generickosť aplikácie

Dôležitým poznatkom je generickosť aplikácie. Databáza OpenStreetMaps ponúka veľké množstvo prvkov a tagov, ktoré je možné nájsť v jej dokumentácií [4]. Celý systém vytvárania dotazov, spracovania odpovedí a zobrazovania 3D objektov je nezávislí od naimplementovanej aplikácie Water Radar, ktorá využíva len časť tagov zobrazujúcich objekty ohľadom vody. Pre zobrazenie inej množiny objektov stačí len prispôsobiť uživateľské rozhranie a upraviť tagy na základe ktorých sa vytvárajú dotazy do databáze.

6. Záver

Táto práca sa venuje vytvoreniu AR technológie, ktorá s pomocou verejne dostupných dát OpenStreetMap zobrazuje užívateľovi v rozšírenej realite informácie o okolitom teréne. Tieto informácie kopírujú reliéf krajiny za pomoci Google Elevation API. Postupne boli stanovené a implementované všetky kladené požiadavky. Výsledný systém je prezentovaný na aplikácií s názvom Water Radar. Táto aplikácia zobrazuje okolité prvky z mapy týkajúce sa vody. Ponúka jednoúrovňový zoznam hodnôt, pomocou ktorých je možné vyfiltrovať si konkrétne prvky.

Poďakovanie

Rád by som poďakoval pánovi profesorovi Ing. Adamovi Heroutovi, Ph.D. za jeho pomocné rady počas tvorby tejto práce.

[4] https://wiki.openstreetmap.org/wiki/Cs:Map_Features#

Literatúra

[1] Colby Gee. *End of year summary of Augmented Reality and Virtual Reality market size predictions*. [Online; navštívené 1.3.2019].

[2] *Augmented Reality*. [Online; navštívené 22.3.2019].

[3] *Why OpenStreetMap?* [Online; navštívené 8.3.2019].

[4] Andrew Hart. Current location accuracy. blogpost (english), August 2017. https://github.com/ProjectDent/ARKit-CoreLocation/wiki/Current-Location-Accuracy.

[5] *Calculate distance, bearing and more between Latitude/Longitude points*. [Online; navštívené 22.3.2019].