SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1132

# RAZVOJ GEOLOKACIJSKE MOBILNE IGRE ZA POTICANJE FIZIČKOG VJEŽBANJA UPORABOM TEHNOLOGIJE PROŠIRENE STVARNOSTI

Eugen Preglej

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1132

# RAZVOJ GEOLOKACIJSKE MOBILNE IGRE ZA POTICANJE FIZIČKOG VJEŽBANJA UPORABOM TEHNOLOGIJE PROŠIRENE STVARNOSTI

Eugen Preglej

Zagreb, lipanj 2023.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Zagreb, 10. ožujka 2023.

# ZAVRŠNI ZADATAK br. 1132

| | |
|---|---|
| Pristupnik: | **Eugen Preglej (0036535836)** |
| Studij: | Elektrotehnika i informacijska tehnologija i Računarstvo |
| Modul: | Računarstvo |
| Mentorica: | prof. dr. sc. Lea Skorin-Kapov |

Zadatak: **Razvoj geolokacijske mobilne igre za poticanje fizičkog vježbanja uporabom tehnologije proširene stvarnosti**

Opis zadatka:

Proširena stvarnost (engl. Augmented Reality, AR) je tehnologija pomoću koje se elementi virtualnog okruženja dodaju u stvarni svijet na način da izgledaju kao dio stvarnog svijeta. Aplikacije u proširenoj stvarnosti temeljene na lokaciji omogućuju prikaz interaktivnih virtualnih objekata u stvarnom prostoru na temelju geolokacijskih oznaka. Područja primjene uključuju navigacijske aplikacije, turističke aplikacije te mobilne igre (npr. Pokémon GO). Vaš je zadatak oblikovati i implementirati mobilnu igru za operacijski sustav Android koja potiče fizičko vježbanje. Aplikaciju je potrebno implementirati uporabom tehnologije proširene stvarnosti na način da se prati kretanje korisnika te se na temelju lokacije korisniku prikazuju odabrani virtualni objekti. Korisniku je omogućena interakcija s virtualnim objektima koji su postavljeni u ovisnosti o unaprijed definiranim geolokacijskim oznakama.

Rok za predaju rada: 9. lipnja 2023.

## Sadržaj

# Uvod

U današnje doba, obilježeno sjedilačkim načinom života i brzim tehnološkim napretkom, rekreacija i kretanje postali su zapostavljeni aspekti našega života. Redovita tjelesna aktivnost pruža brojne fizičke, mentalne i emocionalne koristi te je ključna za održavanje zdravlja i blagostanja.

U okviru ovog rada spominju se dvije, danas veoma zastupljene, tehnologije. Prva je proširena stvarnost (engl. *Augmented Reality*, skr. AR), kojom se u stvarni svijet dodaju elementi virtualnog okruženja tako da izgledaju kao da su dio stvarnog svijeta [1]. Druga je globalni sustav za pozicioniranje (engl. *Global Positioning System*, skr. GPS), kojim uz pomoć 24 satelita koji orbitiraju Zemlju, možemo odrediti naš približan položaj, tj. koordinate na Zemlji [2].

Primjenom navedenih tehnologija, mogu se razviti različite vrste aplikacija, a u ovome radu bit će opisan proces stvaranja aplikacije ***Squiare***. *Squiare* je mobilna geolokacijska aplikacija za Android sustave čiji je cilj potaknuti korisnika na tjelesnu aktivnost. Aplikacija radi tako da korisniku iscrtava rutu koju on zatim mora prehodati kako bi mogao uspješno odraditi vježbu. Svaka ruta koju aplikacija iscrta vodi korisnika do njemu najbliže, još nezauzete virtualne tvrđave. Pomoću tehnologije prošurene stvarnosti, virtualna tvrđava prikazuje se korisniku kao objekt smješten u stvarnom svijetu. Kako bi korisnik zaista uspješno odradio vježbu, na kraju prehodanog puta, mora odigrati, tj. pobijediti u AR mini igri (engl. *minigame*). Korisniku su na karti iscrtani razni virtualni elementi šume poput drveća, panjeva i kamenja te su isto tako iscrtani katapulti koje korisnik sakuplja pritiskom na njih. Virtualni katapulti se sakupljaju kako bi se pobijedilo u AR mini igri, u kojoj korisnik mora postaviti sakupljene katapulte tako da oni gađaju i u konačnici poraze virtualnu tvrđavu. Aplikacija je zamišljena tako da cilj svakog korisnika bude osvojiti što više tvrđava i usput se fizički aktivirati.

Ovaj je rad podijeljen u tri glavna poglavlja. U prvom je poglavlju objašnjen pojam proširene stvarnosti te je ukratko opisana povijest tehnologije, njezin način rada te današnja primjena. U drugom je poglavlju opisan razvoj aplikacije *Squiare*, tj. opis same aplikacije, sve korištene tehnologije te konkretan opis procesa razvoja i izgradnje igrive

inačice aplikacije. Posljednje poglavlje sadrži zaključak, reference na korištene izvore i literaturu, moguće smjerove proširivanja aplikacije te sažetak rada.

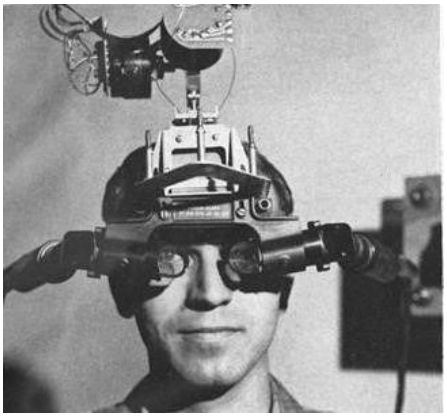
# 1. Proširena stvarnost

## 1.1. Definicija proširene stvarnosti

Kao što je spomenuto u uvodu, AR je tehnologija kojom se u stvarni svijet dodaju elementi virtualnog okruženja tako da izgledaju kao dio stvarnog svijeta. Ovdje je važno naznačiti razliku proširene i virtualne stvarnosti (engl. *Virtual Reality*, skr. VR). Razlika između te dvije tehnologije leži upravo u zastupljenosti slike stvarnog svijeta. Dakle, osnovna razlika između tehnologija AR i VR je u tome što AR koristi sliku stvarnog svijeta kao pozadinu te time pruža korisniku osjećaj interakcije između stvarnog i virtualnog svijeta, dok VR stvara potpuno virtualno okruženje te time pruža korisniku osjećaj uronjenosti, stoga je važno ne miješati te pojmove [3]. Glavne karakteristike AR su kombinacija stvarnog svijeta i virtualnih podataka, interakcije u stvarnom vremenu i 3D poravnavanje virtualnog sa stvarnim [1]. Tehnološka rješenja koja podržavaju glavne karakteristike AR razvijala su se dugi niz godina pa je zahvaljujući tome, danas moguće koristiti AR na različitim uređajima, što je dovelo do povećane popularnosti i razvoja AR aplikacija.

## 1.2. Povijest proširene stvarnosti

Proširena stvarnost ima dugu povijest koja seže u prošlo stoljeće, prvi veliki iskorak učinio je Ivan Sutherland kada je 1968. proizveo prvi zaslon montiran na glavu (engl. *Head-Mounted Display*, skr. HMD) (slika 1.1) [4].



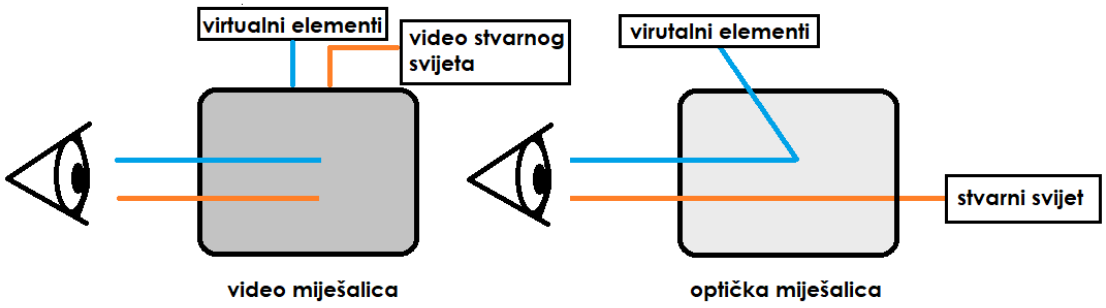
Slika 1.1 Prikaz prvog HMD-a „The Sword of Damocles“, preuzeto iz [4]

To je pokrenulo brojna akademska istraživanja i eksperimente 1990-ih kojima je cilj bio razvitak algoritama i tehnika praćenja položaja, prepoznavanja oblika te integracije virtualnih objekata u stvarni svijet. Sam pojam „proširena stvarnost“ nastaje u tim godinama od strane Boeingova inženjera T. Caudella [5]. Kontinuiranim napretkom u tehnologijama senzora, zaslona i računalne grafike AR polako, ali sigurno napreduje kroz godine. Što nas vodi do današnjice, kada se pomoću razvoja mobilnih tehnologija, AR masovno proširio u svakidašnjost industrije zabave, edukacije i marketinga.

## 1.3. Načini rada proširene stvarnosti

### 1.3.1. Miješanje slike

Miješanje slike je pojam kojim se opisuje kako spojiti virtualnu sliku sa stvarnim svijetom [1]. Može se ostvariti na više načina, jedan od njih je optička miješalica (engl. *optical see-through*) gdje se uz pomoć poluprozirnog ogledala omogućuje da korisnik u isto vrijeme vidi dvije slike (stvarnu izravno kroz ogledalo i virtualnu na ogledalu). Drugi je video miješanje (engl. *video see-through*) gdje se korisniku, uz virtualnu sliku, na zaslonu prikazuje i stvarna slika snimljena kamerom koja se nalazi ispred zaslona (slika 1.2). Postoji i takozvano projekcijsko miješanje gdje su virtualni podaci projicirani direktno na stvaran svijet (miješanje se odvija direktno u prostoru, a ne na zaslonu). U ovom radu korišteno je video miješanje. Stvarna slika dobiva se iz kamere mobilnog uređaja, a virtualna se generira u uređaju na temelju stvarne slike. Virtualna i stvarna slika se kombiniraju u video miješalici (engl. *video compositor*) te se tako pomiješana slika korisniku prikazuje na zaslonu njegovog uređaja [1].




Slika 1.2 Ilustrativni prikaz različitih vrsta miješanja slike, prilagođeno iz [6]

### 1.3.2. Prikaz slike

Miješanje i prikaz slike su usko povezani pojmovi jer prikaz slike proširene stvarnosti ovisi o tehnologiji miješanja računalne grafike i stvarnog svijeta, također i o načinu na koji je zaslon montiran i tehnologiji zaslona koji se koristi [1]. Kada se govori o prikazu slike, fokus se stavlja upravo na površine na kojima se prikaz zbiva pa tako možemo razlikovati prikaze na zaslonu na glavi (slika 1.3), zaslonu u oku, zaslonu u ruci itd.

Slika 1.3 Prikaz suvremenog HMD-a Oculus Quest 2, preuzeto iz [7]

Zasloni na glavi jedan su od najčešćih rješenja za prikaz proširene stvarnosti te ih je moguće koristiti i za optičko i za video miješanje. Zasloni se razlikuju po veličini, težini, udobnosti, vidnom kutu i rezoluciji zaslona. Potencijalno bolji prikaz može se dobiti korištenjem zaslona u oku. Primjer takvog zaslona bio bi virtualni retinalni zaslon koji koristi lasere niske snage i mikromehanički upravljana ogledala za crtanje slike izravno na retini oka. Postoje i izvedbe AR koje ne zahtijevaju da korisnik nosi ikakav zaslon, već je zaslon stacionaran negdje u prostoru. Primjer takvih zaslona bili bi zasloni koji funkcioniraju poput proširenih ogledala (engl. *augmented mirror*) ispred kojih bi korisnik stajao i mogao vidjeti virtualnu sliku sebe u različitoj odjeći.

Zasloni u ruci su veoma kompaktna i praktična izvedba video miješanja, stoga se prikaz pomoću takve vrste zaslona koristi i u ovom radu. Uređaji kojima se ostvaruje zaslon u ruci su tableti, osobni digitalni asistenti (engl. *personal digital assistant*, skr. PDA) i pametni telefoni (engl. *smartphone*). Takav prikaz postaje sve popularniji zbog rasprostranjenosti mobilnih uređaja te činjenice da današnji uređaji imaju većinu sklopovlja potrebnog za izvedbu AR. Posljednja vrsta prikaza su projekcijski prikazi koji se koriste isključivo projekcijskim miješanjem slike. Oni se ostvaruju projekcijom slike s projektora na okolinu,

a sami projektori mogu biti statični ili dinamični poput projektora montiranog na glavu i projektora držanog rukom. Problem ovakvog sustava je deformacija slike koja nastaje projiciranjem na zidove iz promjenjivog položaja projektora.

### 1.3.3. Poravnavanje virtualne slike i stvarnog okruženja

Poravnavanje (engl. *registration*) je središnji problem proširene stvarnosti, kojemu je cilj precizno poravnati stvarni svijet i virtualne predmete, i to ne na zaslonu, nego u 3D prostoru. Cilj poravnanja je konstruirati virtualnu scenu u kojoj koordinatni sustav odgovara koordinatnom sustavu stvarnog svijeta, drugim riječima, scenu u kojoj je položaj promatrača i svih predmeta poznat.

#### 1.3.3.1 Slijeđenje objekata

Slijeđenje objekata je omogućeno pomoću računalnog vida (engl. *computer vision*) i algoritama strojnog učenja koji omogućuju prepoznavanje karakteristika objekata i virtualno preklapanje informacija ili objekata u stvarnom svijetu. Razlikujemo dva načina slijeđenja objekata, slijeđenje s oznakama i slijeđenje bez oznaka.

U slijeđenju s oznakama (engl. *marker-based tracking*), posebne oznake, tj. markeri (često QR kodovi) se postavljaju na objekte ili površine u stvarnom svijetu te se skeniraju kamerom uređaja. Na taj način uređaj može lagano prepoznati površine, tj. oznake te stvarati virtualne objekte nad njima [1].

Složenija inačica slijeđenja, slijeđenje bez oznaka (engl. *marker-less tracking*) ne zahtijeva posebne oznake za prepoznavanje objekata. Umjesto toga, algoritmi za prepoznavanje oblika i uzoraka koriste prirodne značajke objekata kako bi ih prepoznali i pratili (npr. rubove stvari, kutove, promjene teksture, itd.). Tako se istodobno lokalizira kamera u prostoru te stvara 3D mapa scene koja se puni virtualnim objektima [1].

#### 1.3.3.2 Pogreške poravnavanja

Prilikom poravnavanja može doći do raznih pogrešaka koje rezultiraju krivim položajem virtualnih predmeta u odnosu na stvarni svijet. Pogreške poravnavanja mogu nastati zbog nepreciznosti opreme i optičkog sustava. Precizno poravnavanje virtualnih objekata zahtijeva ispravno podešavanje parametara virtualne kamere prema stvarnoj kameri ili ljudskom oku. Dinamičke pogreške proizlaze iz kašnjenja prikaza virtualne slike, do kojeg dolazi zbog raznih elemenata nekog sustava, stoga je najbolji način smanjenja te pogreške

smanjenje kašnjenja, ali i ono ima svoje granice. Tehnike predviđanja položaja mogu donekle kompenzirati pogreške, ali nagli pokreti i orijentacija korisnika mogu i dalje prouzročiti pogreške. Stoga se, za što brže i jednostavnije iscrtavanje, nekada koristi jednostavna translacija postojeće slike kako bi se kompenziralo za pogreške orijentacije.

### 1.3.4. Prikupljanje podataka

Prikupljanje podataka (engl. *sensing*) je zajedničko ime za široki skup tehnologija dobivanja dodatnih podataka za prikaz u proširenoj stvarnosti [1]. To uključuje medicinske slike, slike dubine i prikaze iz baza podataka. Preciznost i poravnavanje sa stvarnim predmetima su ključni. Medicinske slike se mogu poravnati s pacijentovim tijelom, a iz takvih slika mogu se generirati 3D modeli za prikaz u proširenoj stvarnosti. Slike dubine pružaju informacije o udaljenosti svake točke u slici pa se koriste za precizno preklapanje stvarnog svijeta i virtualnih objekata te praćenje ljudi. Metode dobivanja slike dubine uključuju stereo slike, mjerenje vremena povratka svjetlosti, lasersko skeniranje i projekciju strukturiranog svjetla. Prikazivanjem informacija iz baza podataka omogućen nam je unos korisnih informacija u proširenu stvarnost poput shema instalacija, unutarnje strukture zgrada ili dijelova strojeva.

## 1.4. Primjene tehnologije proširene stvarnosti

Proširena stvarnost je područje čiji intenzivni razvoj kreće tek od 1990-ih, stoga je veliki dio primjena još uvijek eksperimentalan. Jedan od primamljivih aspekta AR-a je izravan pristup informacijama unutar vidokruga korisnika (integracija informacija u stvarni svijet). Takva tehnologija bi omogućila brz i jednostavan pristup informacijama, ali trenutni nedostaci opreme, zaslona i slijeđenja ograničavaju potencijalnu korist u svakodnevnoj uporabi. U budućnosti se očekuje da će proširena stvarnost pronaći široku primjenu na velikim, otvorenim prostorima, posebno s dolaskom jednostavnih, laganih i prenosivih sustava, a već sada postoje brojna područja u kojima se AR primjenjuje poput medicine, proizvodnje i održavanja, arhitekture, navigacije, robotike i drugih komercijalnih svrha [1].

U medicini se AR može koristiti za treniranje kirurga, neovisno o njihovoj razini iskustva. Medicinske slike (MRI, CT, ultrazvuk) mogu se predstaviti direktno na pacijentu, čime se dobiva vrsta virtualnog rendgena u stvarnom vremenu (slika 1.4) [8]. Takva je pomoć korisna i početnicima koji se tek upoznavaju s ljudskim tijelom i iskusnim kirurzima za uvježbavanje kompliciranih kirurških procedura.

Slika 1.4 Prikaz primjera uporabe AR-a u medicini, preuzeto iz [8]

U arhitekturi se pak AR može koristiti za dizajn interijera ili prikazivanje strukturi poput električnih ili vodovodnih instalacija unutar zidova. Takva primjena može biti korisna i drugim profesionalcima poput vodoinstalatera ili električara prilikom popravaka ili instalacije novih vodova. Čak i najobičniji korisnik, npr. vlasnik stana, može profitirati od takvih aplikacija jer pomoću njih može isprobati hoće li mu se neki komad namještaja koji želi kupiti dobro uklopiti u stan (slika 1.5) [9].

Slika 1.5 Prikaz primjera uporabe AR-a za dizajn interijera, preuzeto iz [9]

Primjena najsličnija aplikaciji *Squiare* bila bi primjena u industriji videoigara. Primjer koji možemo izdvojiti je mobilna igra Pokemon GO. Igra je objavljena 2016. godine i odmah je doživjela veliki uspjeh, a danas je najpopularnija videoigra u povijesti Sjedinjenih Američkih Država (skr. SAD) [10]. To je umrežena igra u kojoj se korisnici natječu tko će skupiti jača čudovišta i koji tim će imati prevlast u kojem dijelu grada. Igra se tako da korisnik, hodajući uokolo, gleda kartu u aplikaciji i na njoj traži razna virtualna čudovišta. Kada korisnik pronađe jedno takvo čudovište, on ulazi u AR mini igru u kojoj mu je cilj zarobiti to čudovište (slika 1.6). Igra Pokemon GO imala je utjecaj na mnoge korisnike te je također pridonijela razvoju i imala snažan utjecaj na ovaj rad.


Slika 1.6 Prikaz AR mini igre iz igre Pokemon GO, preuzeto iz [11]

# 2. Razvoj aplikacije Squiare

U ovom poglavlju opisan je razvoj aplikacije *Squiare*. Prvo je dan kratak opis aplikacije, zatim se govori o specifičnim tehnologijama koje su korištene prilikom razvoja te je na kraju opisan proces razvoja od početnih koraka do puštanja aplikacije u pogon.

## 2.1. Opis aplikacije

Aplikacija *Squiare* mobilna je geolokacijska aplikacija razvijena za Android sustave, zamišljena kao motivator, tj. pomagalo korisnicima da se kreću i zabave tijekom rekreacije. Aplikacija je razvijena kao igra za jednog igrača (engl. *singleplayer*), ali se zbog praćenja rezultata i implementiranog bodovnog sustava (engl. *scoring system*) očekuje da će korisnici međusobno uspoređivati svoje rezultate izvan aplikacije.

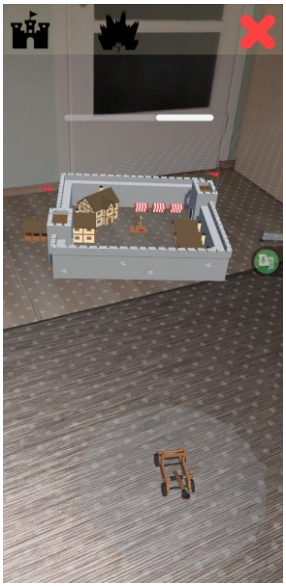
Naziv aplikacije dolazi od spoja engleske riječi *squire* i kratice AR. Riječ *squire* može značiti štitonoša ili plemić ovisno o kontekstu u kojem se koristi, no točna interpretacija nam ovdje nije bitna jer je cilj te riječi predstaviti srednjovjekovnu tematiku aplikacije. Kratica AR u imenu pak predstavlja AR aspekt ove aplikacije, točnije mini igru osvajanja tvrđavi koja je implementirana na kraju svake vježbe. Kombinacijom tih dvaju pojmova dobivamo naziv *Squiare*.

Aplikacija se sastoji od tri scene. Prva scena, koja se korisniku prikazuje nakon što upali aplikaciju, je glavni izbornik (engl. *main menu*). U glavnom izborniku korisniku se nudi da započne igru pritiskom na "Play", provjeri kako igrati igru pritiskom na "How to play" i izađe iz aplikacije pritiskom na "Quit". Druga scena je geolokacijski dio igre, koji se pokreće, kada korisnik odabere "Play". U njoj korisnik može vidjeti svoju trenutnu lokaciju u stvarnom svijetu prikazanu na virtualnoj karti. Korisnikova lokacija se osvježava skoro u stvarnom vremenu i prikaz korisnika na karti prati korisnikovo kretanje po stvarnom svijetu. Iz te scene korisnik započinje svoje vježbe i pritiskom sakuplja virtualne katapulte, koji se nasumično stvaraju posvuda na virtualnoj karti. Kada korisnik započne vježbu, na karti mu se iscrtava ruta (dobivena iz Mapbox Directions API-ja) za koju se očekuje da će korisnik pratiti kako bi odradio vježbu. Iscrtana ruta narančaste je boje i podsjeća na rute s kojima se susrećemo u navigacijskim aplikacijama poput Google Maps. Korisniku je u

cilju prilikom putovanja sakupiti što više katapulta jer će mu oni biti potrebni na kraju vježbe, tj. u trećoj sceni (slika 2.1).


Slika 2.1 Prikaz geolokacijske scene

Treća scena je AR mini igra i ona kreće kada korisnik prehoda rutu koja mu je bila iscrtana u prethodnoj sceni te tako dođe do posljednje točke te rute, koja je uvijek virtualna tvrđava prikazana na karti. U trećoj sceni se, uporabom kamere na korisnikovu uređaju, skenira okolina i stvara virtualna AR scena. U toj sceni korisnik ima zadatak postaviti sakupljene virtualne katapulte tako da oni gađaju virtualnu tvrđavu. Kada katapulti dovoljno puta pogode tvrđavu, tvrđava će biti osvojena te mini igra završava, a korisnik biva nagrađen za svoj trud (slika 2.2).



Slika 2.2 Prikaz AR scene

Kako bi ispravno funkcionirala, aplikacija od korisnika traži pristup njegovim lokacijskim podatcima (za ispravno funkcioniranje geolokacijske scene) i pristup kameri (za ispravno funkcioniranje AR scene). Budući da te dvije scene čine srž ove aplikacije, potrebno ih je detaljnije analizirati. Prva je analiza geolokacijske scene. U njoj se korisniku prikaže tematski prikaz karte i 3D objekt, koji predstavlja korisnika. Korištenjem tehnologije GPS, uređaj pokušava dobiti signal od barem 4 od ukupno 24 GPS satelita koji kruže oko Zemlje. Na temelju dobivenih signala i kompleksnih matematičkih operacija, GPS prijemnik u uređaju izračunava koordinate na kojima se sam uređaj nalazi. Aplikacija od mobilnog uređaja traži te koordinate te tako saznaje gdje se u svijetu nalazi korisnik.

Svakim novodobivenim koordinatama, na virtualnoj se karti osvježava položaj 3D objekta koji predstavlja korisnika te se tako simulira njegovo kretanje. Na karti se isto tako prikazuju i razni virtualni objekti poput drveća, kamenja, tvrđava itd. Modeli tih objekata izrađeni su stilom s malim brojem poligona (engl. *low poly style*), zbog kojeg aplikacija postiže bolje performanse na slabijim uređajima, što je veoma važno kada se na kartu iscrtava više desetaka istih objekata. Među tim virtualnim objektima nalaze se i katapulti koje korisnik sakuplja jednostavnim pritiskom na njihov model iscrtan na karti. Katapulti su potrebni korisniku kako bi na kraju vježbe uspješno osvojio virtualnu tvrđavu koja se nalazi na unaprijed određenom mjestu na karti. Kako bi mogao započeti proces osvajanja tvrđave, korisnik prvo mora započeti svoju vježbu pritiskom na sličicu u donjem desnom kutu aplikacije koja pokazuje čovječuljka koji hoda. Kada je vježba pokrenuta, korisniku se na karti iscrtava ruta kojom može doći do najbliže neosvojene tvrđave. Na primjer, korisnik u gradu Zagrebu pokrene vježbu, vježba crta rutu koja vodi korisnika do neke točke, gdje se na karti unutar aplikacije nalazi virtualna tvrđava. Kada dođe na kraj rute, korisnik se nalazi na Trgu bana J. Jelačića te unutar aplikacije, na mjestu koje bi odgovaralo adresi Trg bana J. Jelačića 15, pronalazi virtualnu tvrđavu. Nakon što korisnik dohoda do tvrđave, pritiskom na nju, ulazi u AR mini igru.

Slijedi analize AR scene. U AR sceni korisnik prvo mora malo okretati svoj mobilni uređaj uokolo na način da mu kamera vidi što veći dio njegova okruženja. Kada algoritmi za prepoznavanje površina unutar aplikacije uspiju pronaći horizontalne površine, korisniku se pronađene površine označuju virtualnim pokazivačem na zaslonu njegovog uređaja. Korisnik tada može postaviti reprezentaciju virtualne tvrđave, koju napada, bilo gdje na iscrtane površine unutar svoje scene. Nakon toga korisnik postavlja sakupljene virtualne katapulte isto tako na iscrtane površine oko virtualne tvrđave i pokušava je osvojiti.

Tvrđava će biti osvojena kada joj razina otpora padne na nulu. Svaki katapult ispucava projektil jednom u 5 sekundi. Svaki projektil koji pogodi tvrđavu, spušta razinu otpora tvrđave za vrijednost ovisnu o brzini kojom je projektil letio. Svi korišteni modeli u ovoj sceni jednaki su onima iz geolokacijske scene, samo su skalirani na puno manje veličine kako bi se dobila preglednost u AR-u. Tvrđava će biti osvojena kada joj se razina otpora, prikazana crtom iznad nje, spusti na nulu. Korisnik, ako uspješno zauzme tvrđavu, biva nagrađen određenim brojem dukata (količina se skalira duljinom rute koju aplikacija iscrta). Dukati predstavljaju postignuti rezultat te služe korisnicima kako bi mogli jedni s drugima usporediti svoju uspješnost u osvajanju tvrđava. Cilj svakog korisnika trebao bi biti sakupiti što više dukata, tj. osvojiti što više tvrđava te tako zapravo prehodati što više kilometara.

## 2.2. Korištene tehnologije

### 2.2.1. Unity

Unity je popularna razvojna platforma za izradu višeplatformske programske potpore koji omogućuje stvaranje interaktivnih 2D i 3D aplikacija poput videoigara, AR te VR aplikacija [12]. Unity Editor je program u kojemu oblikujemo naše projekte, tj. aplikacije, a radi na principu povuci-i-ispusti (engl. *drag-and-drop*) zahvaljujući kojem korisnici koji razvijaju svoju aplikaciju jednostavno mogu dodavati ili uklanjati elemente poput skripti unutar neke scene i upravljati odnosima među samim scenama. Za zahtjevnije funkcije Unity podržava programski jezik C#, u kojem korisnik može pisati svoje skripte za specifične funkcionalnosti koje su mu potrebne. Važno je naglasiti da postoji i Unity Asset Store gdje korisnici s manjim znanjem u nekim područjima, npr. modeliranju mogu nabaviti već gotove modele te si tako olakšaju razvoj svoje aplikacije. Prilikom razvoja ovog rada korišten je Unity verzije 2021.3.21f1 uz prateći Android SDK koji dolazi uz tu verziju. Pošto smo razvijali AR aplikaciju za Android sustave, unutar Unity Editora morali smo naznačiti da želimo koristiti Googleovu AR podršku pod nazivom ARCore koja nam omogućuje korištenje stvari poput stvaranja dubinskih slika koje su potrebne za ispravno funkcioniranje AR [13].

### 2.2.2. Microsoft Visual Studio

Microsoft Visual Studio je integrirano razvojno okruženje (engl. *integrated development environment*, IDE) namijenjeno za razvoj računalnih programa [14]. Program podržava široki spektar jezika, a ovdje se koristi za jednostavnije uređivanje i pisanje našeg C# koda. Prilikom razvoja ovoga rada korišten je Microsoft Visual Studio 2019.

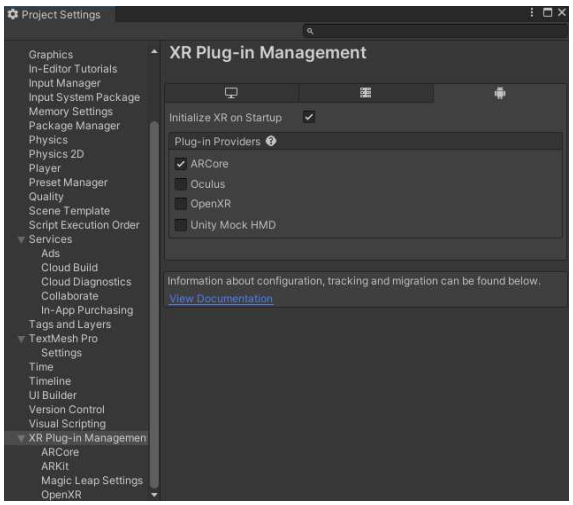
### 2.2.3. Mapbox

Mapbox je platforma za praćenje lokacije koja pruža usluge kartiranja, geokodiranja, usmjeravanja i slične [15]. U ovom projektu se koristi Mapbox SDK for Unity, alat koji se koristi za integraciju geoprostornih funkcionalnosti u aplikacije i platforme. Mapbox pruža globalne kartografske podatke i informacije o prometu, što omogućuje preciznu navigaciju na kartama. Također nudi alate koji olakšavaju pristup i manipulaciju s lokacijskim podacima stoga je korišten kao podloga za razvoj funkcionalnosti ovog projekta. Sam Mapbox SDK for Unity je nažalost napušten od 2019. godine stoga je bilo potrebno ručno izvesti velik broj funkcija. Inače se Mapbox koristi u raznim industrijama poput automobilske industrije, logistike, poslovne inteligencije, maloprodaje, putovanja i mnogim drugima te predstavlja sjajnu alternativu Google Maps API-ju.

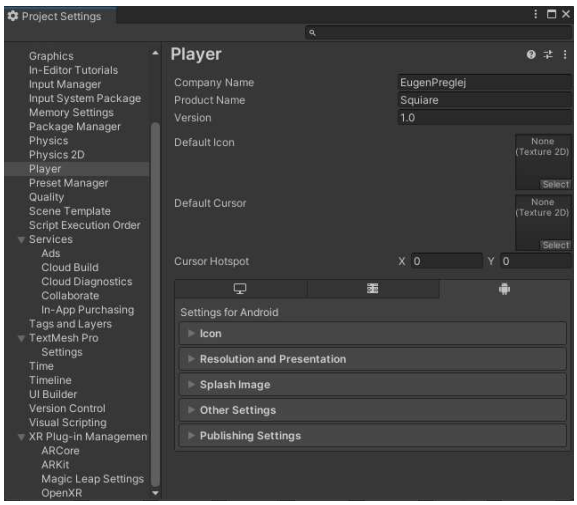
## 2.3. Postavljanje razvojne okoline

Nakon preuzimanja i instalacije prethodno navedenih alata potrebno je stvoriti novi projekt u Unity Hubu. Prije stvaranja projekta potrebno je preuzeti verziju Unityja koja će se koristiti te označiti da se instalira prateći modul Android Build Support. Android Build Support sadrži alate za razvoj Android programske podrške (engl. *Software Development Kit*), razvoj Android programske podrške s nativnim kodom (engl. *Native Development Kit*) te implementaciju platforme Java (OpenJDK), stoga je neophodan za razvoj Android aplikacija. Prilikom stvaranja novog projekta odabran je AR predložak kako bi si olakšali postavljanje AR aspekta aplikacije. Nakon što je projekt stvoren, uvezeni (engl. *import*) su svi alati i modeli koji su potrebni za realizaciju projekta poput Mapbox SDK for Unity te Low Poly Ultimate Pack. Posljednje promjene koje je potrebno napraviti prije početka rada su promjene u postavkama projekta unutar Unityja. U Project Settingsima u kategoriji XR Plug-in Management pod sličicom koja predstavlja Android operativni sustav moramo označiti da želimo koristiti ARCore (slika 2.3) te u kategoriji Player treba postaviti

informacije o aplikaciji, tj. naziv proizvođača aplikacije i naziv same aplikacije kako bi se datoteke pravilno inicijalizirale na mobilnim uređajima (slika 2.4).




Slika 2.3 Prikaz uključenog korištenja ARCore




Slika 2.4 Prikaz informacija o aplikaciji

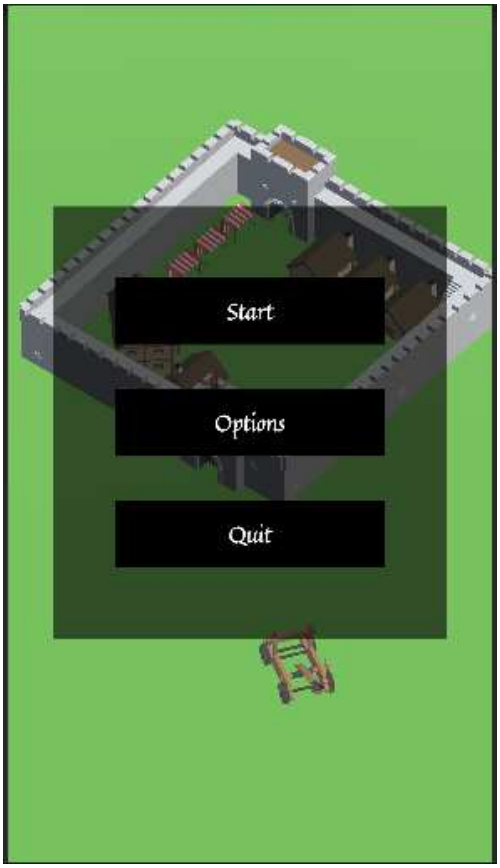
## 2.4. Razvoj sceni

Nakon što je postavljena razvojna okolina, potrebno je razviti sve scene od kojih je aplikacija građena. U izborniku *Project* unutar direktorija *Assets* stvaramo novi direktorij *Scenes* u njemu će se nalaziti sve razvijene scene.

### 2.4.1. Razvoj scene glavnog izbornika

Scena glavnog izbornika stvorena je tako da je u izborniku Project unutar direktorija Scenes stvorena nova scena, desnim klikom te odabirom opcija *Create* → *Scene*. Nakon što je scena imenovana u „*Main Menu",* započinje njen razvoj. U scenu se u izborniku *Hierarchy* dodaje prazno platno (engl. *canvas*) koje predstavlja zaslon našeg mobilnog uređaja. Desnim klikom na platno u izborniku Hierarchy i odabirom *UI* → *Image* dodaje se slika koja će biti okvir glavnog izbornika. Slika je obojena u crno te joj je dodana prozirnost, jer će iza nje biti dodana pozadina sačinjena od 3D objekata. Na objekt slike, u izborniku Hierarchy, desnim klikom i odabirom *Create Empty* u scenu je dodan novi prazni objekt. Imenujemo ga „*Menu Items"* i u njemu stvaramo 3 nova objekta na koje dodajemo komponentu *Button*. Ti objekti predstavljat će gumbe (engl. buttons) koji će biti prikazani korisniku u glavnom izborniku. U svaki od njih upisan je odgovarajući tekst poput „Start", „How To Play" i „Quit" (slika 2.5). Na objekt Menu Items dodana je skripta *MainMenu.cs* koja sadrži funkcije za svaki gumb u glavnom izborniku i skripta

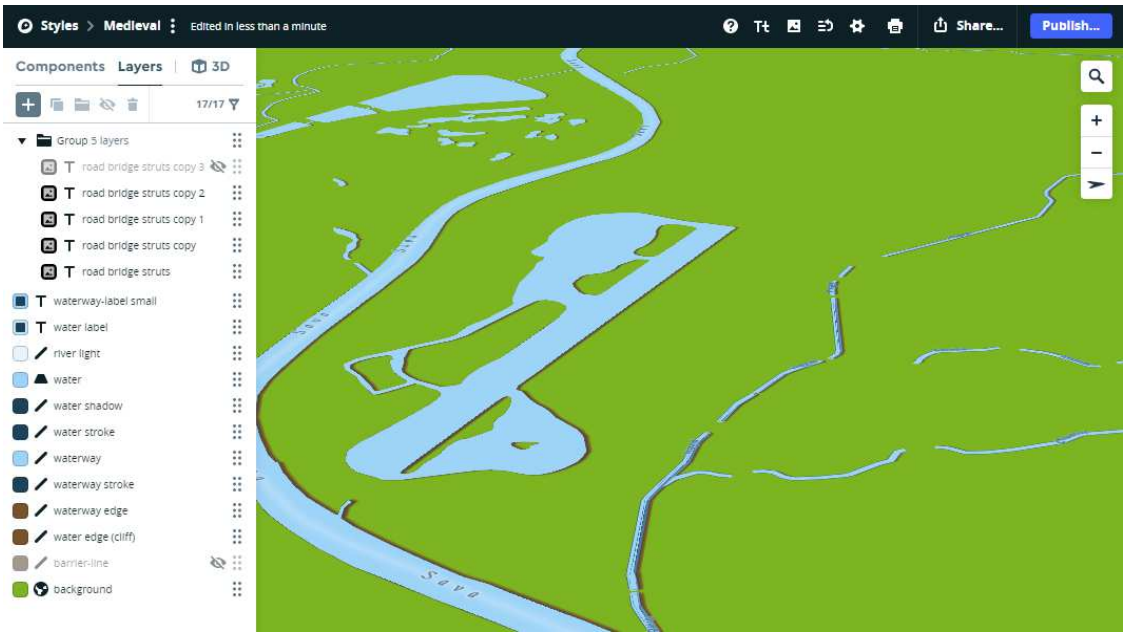
TransitionManager.cs. *TransitionManager.cs* je skripta iz preuzeta iz EasyTransitions *asseta* [16]. Ona upravlja prijelazima, tj. vizualnim efektima prilikom prijelaza između dviju scena. Svakom gumbu iz objekta *Menu Items* na *OnClick* element komponente *Button* stavljena je odgovarajuća funkcija iz skripte *MainMenu.cs* (npr. na *Start Button*-u za *OnClick* stavljana je funkcija *StartGame()*). Nakon ovog koraka, započinje razvoj geolokacijske scene, ali nakon što je njezin razvoj gotov, nakratko se vraćamo upravo na ovu scenu. Razlog povratku je taj što je sada moguće stvoriti 3D pozadinu iza izbornika. U scenu dodajemo objekte Fort i Catapult koji su stvoreni u geolokacijskoj sceni i namještamo kameru da gleda na njih. Time je ova scena završena.




Slika 2.5 Prikaz prototipa glavnog izbornika

### 2.4.2. Razvoj geolokacijske scene

Razvoj geolokacijske scene započinje modificiranjem Location Based Game scene koja dolazi u paketu s Mapbox SDK-om. U toj sceni nalazi se jednostavan map loader objekt i model igrača koji su unaprijed povezani, što nam dosta olakšava posao. Ime scene je promijenjeno u Map. Prvi korak je podešavanje, tj. stvaranje novog stila izgleda karte. Na stranici https://studio.mapbox.com/ kreiran je jednostavan stil izgleda karte isključivo od pozadinske zelene boje i na njoj je uključen prikaz rijeka i ostalih tijela vode poput potoka, jezera i sličnih (slika 2.6). Nakon što je karta kreirana, njezin stil se može koristiti u Unityju tako da na stranici pritisnemo ikonu pored Share i kopiramo link u naš Unity projekt, preciznije u polje Tileset ID/Style URL (prikazuje nakon što za Data Source odaberemo Custom) unutar odjeljka IMAGE skripte Abstract Map koja se nalazi na objektu Map kojeg pronalazimo koristeći se izbornikom Hierarchy.

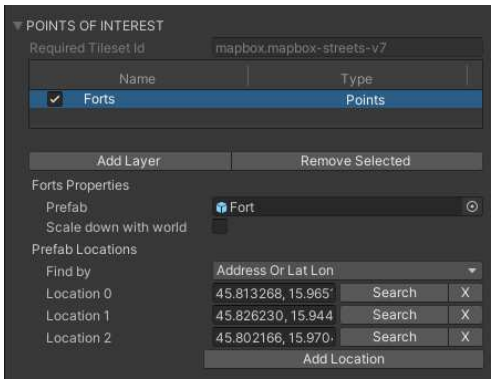



Slika 2.6 Prikaz karte unutar Mapbox Studija

Zatim je potrebno objekt Main Camera staviti unutar objekta Player Target kako bi kamera pratila našeg igrača i u skripti *RotateWithLocationProvider.cs* objekta PlayerTarget označiti polja Use Device Orientation i Use Negative Angle kako bi dobili ispravan prikaz orijentacije igrača unutar aplikacije na Android uređaju.

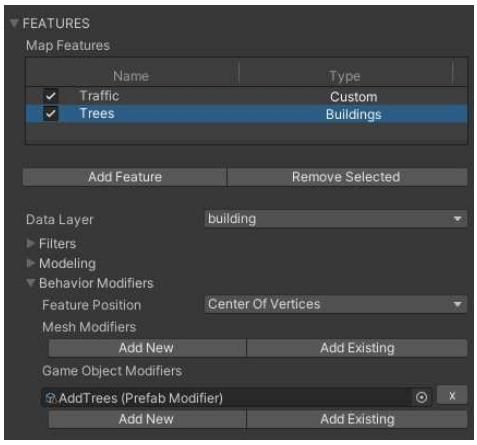
Kombinacijom jednostavnijih modela iz Low Poly Ultimate Packa [17] stvoren je model koji će predstavljati tvrđave te je na taj objekt postavljena skripta *Fort.cs*. Skripta, u uskom radijusu oko sebe, briše ostale objekte, kako ne bi došlo do prolaženja drugih objekata kroz

tvrđavu. Potrebno postaviti sve lokacije na kojima će se objekti tvrđave stvoriti. To se radi tako da se na objektu Map u skripti *AbstractMap.cs* unutar odjeljka *POINTS OF INTEREST* doda sloj na kojem će biti tvrđave pritiskom na Add Layer te postavljanjem polja Prefab na objekt koji predstavlja tvrđave i zatim dodavanjem koordinata (na kojima će biti tvrđave) pritiskom na Add Location (slika 2.7).




Slika 2.7 Prikaz odjeljka *POINTS OF INTEREST* unutar skripte *AbstractMap.cs*

Trenutno u aplikaciji postoje samo tlo i tvrđave. Kako bi se stvorio osjećaj ispunjenog svijeta, koristi se Mapboxova funkcija stvaranja 3D modela zgrada na virtualnoj karti. Mapboxov API omogućuje stvaranje 3D modela zgrada na točkama koje odgovaraju (stvarnim) koordinatama zgradi u stvarnom svijetu. Umjesto modela zgrade, koristeći odjeljak *FEATURES* unutar *AbstractMap.cs*, na virtualnu karte se stavljaju modeli drveća te se time dobiva privid šume (slika 2.8). Kako bi ta drveća tvorila što uvjerljiviju šumu, napisane su skripte *BushSpawner.cs*, *RockSpawner.cs*, *StumpSpawner.cs* te skripta Spawner.cs čija je zadaća upravljati ostalim skriptama koje završavaju na „Spawner“. Skripta *Spawner.cs* upravlja i skriptom *CatapultSpawner.cs* koja je po strukturi jednaka ostalim spawner skriptama, a služi za stvaranje katapulta na pseudonasumičnim lokacijama u stvorenoj šumi (slika 2.9). Nakon što je napravljeno sve što je potrebno za stvaranje objekata na karti, prelazi se na programiranje interakcija s korisnikom i interakcija među objektima.




Slika 2.8 Prikaz odjeljka *FEATURES* unutar skripte *AbstractMap.cs*

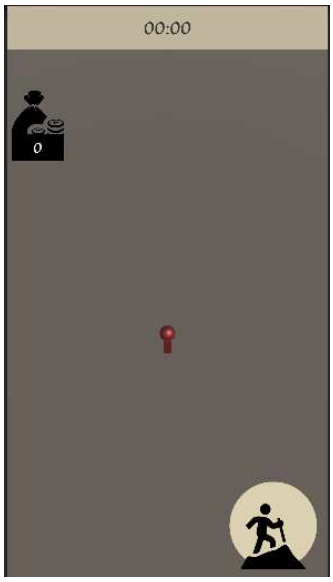
```
private void FixedUpdate()
{
    if (numChildren != transform.childCount)
    {
        for (int i = 0; i < transform.childCount; i++)
        {
            Transform child = transform.GetChild(i);
            if (child.name != "TileProvider" && !processedTiles.Contains(child))
            {
                CatapultSpawner catapultSpawner = child.gameObject.AddComponent<CatapultSpawner>();
                catapultSpawner.catapultPrefab = catapultPrefab;
                catapultSpawner.Spawn();

                BushSpawner bushSpawner = child.gameObject.AddComponent<BushSpawner>();
                bushSpawner.bushPrefab = bushPrefab;
                bushSpawner.Spawn();

                RockSpawner rockSpawner = child.gameObject.AddComponent<RockSpawner>();
                rockSpawner.rockPrefab = rockPrefab;
                rockSpawner.Spawn();

                StumpSpawner stumpSpawner = child.gameObject.AddComponent<StumpSpawner>();
                stumpSpawner.stumpPrefab = stumpPrefab;
                stumpSpawner.Spawn();

                processedTiles.Add(child);
            }
        }
        numChildren = transform.childCount;
    }
}
```

Slika 2.9 Prikaz metode *FixedUpdate()* iz skripte *CatapultSpawner.cs*

Kako bi se korisniku omogućila jednostavnija interakcija s aplikacijom, potrebno je stvoriti jednostavno korisničko sučelje[1] (engl. *user interface*, skr. UI) (slika 2.10). U scenu se dodaje prazno platno koje predstavlja zaslon našeg mobilnog uređaja. Na platno su dodani razni objekti čija će vidljivost i sadržaj koji prikazuju biti kontrolirani skriptom *UIManager.cs*. Welcome Message Panel se prikazuje samo kada korisnik prvi put upali aplikaciju i služi za informiranje korisnika kako se služiti aplikacijom. Start Button prikazuje sličicu čovječuljka koji hoda i pritiskom na nju korisnik pokreće vježbu. U Timer Panelu se nakon započinjanja vježbe prikazuje preostalo vrijeme u kojem je potrebno dovršiti aktivnu vježbu. Mission Result Panel se prikazuje nakon što je korisnik gotov s

[1] sve ikone korištene pri razvoju korisničkog sučelja preuzete su s https://uxwing.com/

vježbom, tj. pokušajem osvajanja tvrđave i informira ga o tome ako je i koliko je bio uspješan u svom pothvatu. Posljednji element korisničkog sučelja je Score Panel koji korisniku prikazuje količinu dukata koje posjeduje, tj. ukupan postignuti rezultat iz svih uspješno odrađenih vježbi.

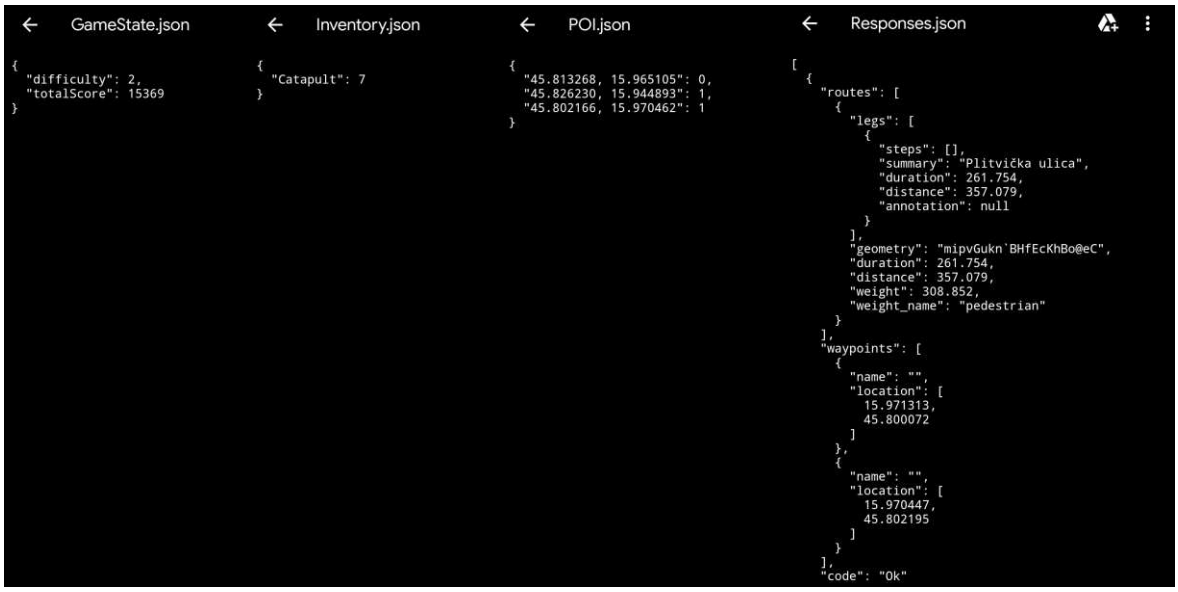



Slika 2.10 Prikaz prototipnog 3D modela igrača i korisničkog sučelja

Sve skripte čije ime završava s „Manager“ bit će u izborniku Hierarchy postavljene na novostvoreni objekt Manager. Uz spomenuti *UIManager.cs* korišteno je još desetak „Manager“ skripti. *DirectionsManager.cs* je najsloženija „Manager“ skripta jer popravlja i nadopunjava nedostatke Mapbox for Unity SDK-a, čiji je razvoj napušten 2019. godine. Directions Manager služi za iscrtavanje već pređenih ruti iz svih uspješno odrađenih vježbi te za iscrtavanje novih, još nezapočetih, ruti. Naime, Mapbox ima ograničen broj API upita za besplatne korisnike pa će svaka vježba koju korisnik uspješno odradi spremiti podatke potrebne za iscrtavanje njene rute u *Responses.json*. Aktivacijom geolokacijske scene, skripta iscrtava sve već pređene rute, a pritiskom na spomenuti Start Button, skripta iscrtava rutu za novu vježbu i započinje odbrojavanje u Timer Panelu. *SingleLocationProvider.cs* je nadopuna za *DirectionsManager.cs* čija je funkcija dohvaćanje trenutnih koordinata igrača u stvarnom svijetu pomoću GPS-a kako bi *DirectionsManager.cs* imao početnu točku rute koju mora iscrtati. *POIManager.cs* je još jedna pomoćna skripta, a služi za čitanje i pisanje podataka u *POI.json*. Datoteka *POI.json* predstavlja rječnik u kojem parove ključ-vrijednost čine koordinate na kojima se nalazi neka tvrđava i broj 0 ili 1 koji predstavlja ako je dotična tvrđava osvojena. Informacija o

tome koja tvrđava je osvojena, a koja nije, potrebna je aplikaciji kako bi odredila do koje tvrđave će voditi ruta u sljedećoj vježbi (ne može voditi do već osvojenih tvrđava).

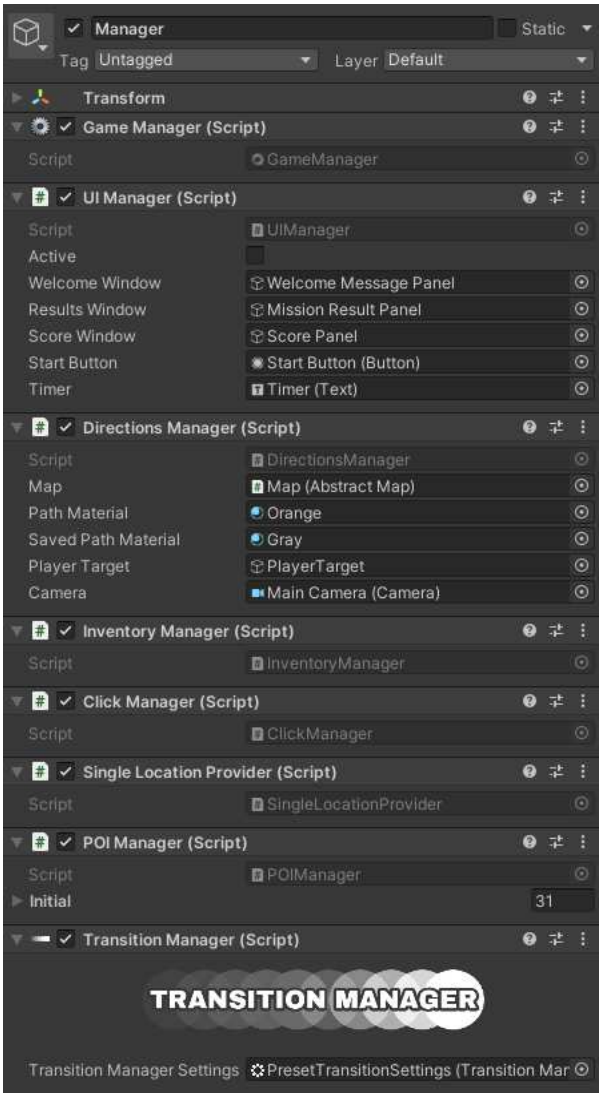
Skripta *InventoryManger.cs* čita i upravlja sadržajem datoteke *Inventory.json*. U toj datoteci čuva se broj sakupljenih katapulta, stoga ona predstavlja predmete koje igrač posjeduje (engl. *inventory*). *GameManager.cs* zadužen je za upravljanje s raznim aspektima aplikacije koji se uglavnom vrte oko stanja igre (engl. *game state*). Najvažnije zaduženje te skripte je upravljati težinom igre (engl. difficulty) i pratiti količinu dukata koje je korisnik zaradio. Ti podatci se spremaju u datoteku *GameState.json*. Koristi se i već spomenuta skripta TransitionManager.cs iz scene glavnog izbornika. Posljednja „Manager" skripta je *ClickManager.cs* koja rukuje s ulazima dobivenim pritiskom elemenata na ekranu. Mnoge spomenute skripte imaju funkciju koja se svodi na upravljanje sadržajem neke datoteke, razlog tome je potreba za perzistentnošću podataka. Naime, svi podaci koji se nalaze u aplikaciji dok je ona aktivna, nestaju gašenjem aplikacije, stoga uporabom spomenute vrste skripti nadilazimo taj problem (slika 2.11). Sve JSON datoteke koje nastaju radom tih skripti spremaju se u zadani direktorij za perzistenciju (engl. default persistence folder) dobiven uporabom `Application.persistentDataPath` varijable koja ovisno o operacijskom sustavu za koji razvijamo aplikaciju vraća put koji će aplikacija koristiti za pohranu nekih svojih podataka.




Slika 2.11 Prikaz prvih 37 redaka iz svih JSON datoteka koje aplikacija koristi

Znajući tu informaciju korisnik bi u teoriji mogao lažirati neke podatke te tako varati u igri, no osiguravanje integriteta tih podataka je izvan dosega ovog rada pa se u radu računa na poštenost svakog potencijalnog korisnika. Nakon što su sve skripte postavljene na objekt

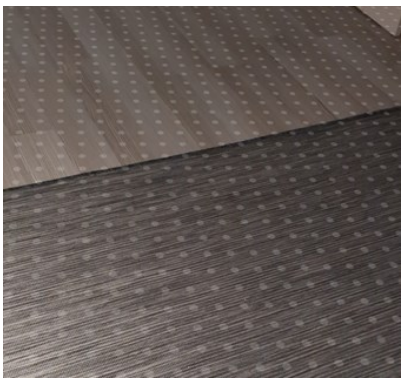
Manager, potrebno je sve javne varijable tih skripti povezati s odgovarajućim komponentama poput materijala, elemenata korisničkog sučelja, 3D objekata i sličnih (slika 2.12). Time je gotov razvoj geolokacijske scene te se prelazi na razvoj AR scene.




Slika 2.12 Prikaz svih skripti na objektu *Manager*

### 2.4.3. Razvoj AR scene

Kako smo prilikom stvaranja projekta u *Unity Hubu* koristili predložak *AR*, stvaranjem projekta stvorila se i zadana (engl. *default*) scena. Ta scena je veoma korisna jer zbog odabranog predloška, sadrži osnovne objekte potrebne za razvoj AR aplikacije. Razvoj scene AR mini igre se tako zapravo zasniva na modificiranju zadane scene. Naziv scene je promijenjen u „AR“. Prvi korak je prikazati korisniku koje površine njegov uređaj uspijeva detektirati. U sceni, na već postojećem objektu *AR Session Origin*, nalazi se skripta *ARPlaneManager.cs* koja stvara objekte na mjestu u sceni gdje je aplikacija detektirala neku površinu. Kako bi korisnik mogao vidjeti te površine, tj. objekte, potrebno je skripti pridružiti objekt koji želimo da se stvara. Drugim riječima, skripti je potreban objekt kojim će teksturirati pronađene površine. U ovom radu korišten je objekt *ARPlane* koji dolazi u sklopu AR predloška, a odabran je upravo zato što se savršeno uklapa u ovu scenu. Naime, objekt *ARPlane* predstavlja jednostavnu prozirnu površinu s točkastim uzorkom koji je lagano vidljiv kada se iscrta, a opet dovoljno je niskoprofilan da ne narušava izgled i jednostavan stil aplikacije (slika 2.13).



Slika 2.13 Prikaz izgleda pronađenih površina teksturiranih objektom ARPlane unutar aplikacije

Sljedeći korak je omogućiti korisniku da stvara virtualne objekte na prikazanim površinama. Kako bi to bilo moguće, stvoren je jednostavan UI koji sadrži 3 ikone. Prva predstavlja tvrđavu, druga katapult, a treća služi za odustajanje od pokušaja osvajanja tvrđave, tj. za povratak u geolokacijsku scenu (slika 2.14). Ovisno o tome koja je ikona odabrana, korisnik će pritiskom na prikazane površine u virtualnoj sceni stvoriti odgovarajuće objekte. Zato je potrebna skripta koja prati koja ikona je odabrana u UI-ju.

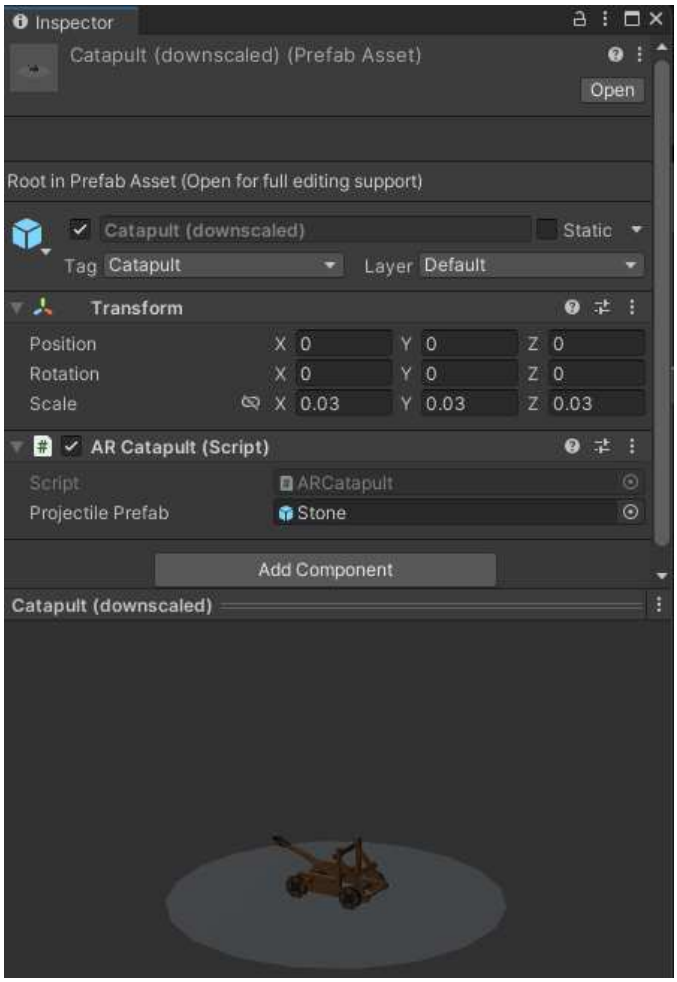
Slika 2.14 Prikaz prototipnog korisničkog sučelja za AR scenu

Kako će još neke funkcionalnosti koje će biti razvijene u ovoj sceni zahtijevati funkcije poput onih iz geolokacijske scene, u izborniku *Hierarchy* stvoren je novi prazan objekt *Manager*. U njega stavljamo *GameManger.cs*, *UIManger.cs*, *POIManager.cs*, *DirectionsManager.cs*, *InventoryManager.cs* i *TransitionsManager.cs* jer će nam sve te skripte biti potrebne i u ovoj sceni. Sada je moguće razviti skriptu *ARPlacer.cs* koja će upravljati stvaranjem objekata na pronađenim površinama u sceni. Kada korisnik pritisne prstom na zaslon, *ARPlacer.cs* uz pomoć skripte *UIManager.cs* dobiva informaciju koji je objekt odabran te ako su svi uvjeti zadovoljeni, stvara odabrani objekt unutar scene. Neki od uvjeta koji se provjeravaju su vrijeme proteklo od posljednjeg stvaranja objekta, udaljenost od drugih stvorenih objekata i slični uvjeti koji služe za osiguravanje što intuitivnijeg rada aplikacije (slika 2.15). Uz pomoć skripte *InventoryManager.cs* se također provjerava ako korisnik u svom *inventoryju* ima dovoljnu količinu objekata koje želi stvoriti jer ne bi imalo smisla dopustiti korisniku da stvara više katapulta nego što ih je sakupio u geolokacijskoj sceni.

```
void PlaceAttackers()
{
    if (Input.touchCount > 0 && Time.time - lastPlacementTime > placementCooldown)
    {
        if (arRaycastManager.Raycast(Input.GetTouch(0).position, hits))
        {
            var hitPose = hits[0].pose;

            if (uiManager.selectedItem == "Catapult")
            {
                inventory = inventoryManager.GetInventory();

                if (inventory.ContainsKey("Catapult"))
                {
                    Collider[] colliders = Physics.OverlapSphere(hitPose.position, 0.15f);

                    bool canSpawn = true;

                    foreach (Collider collider in colliders)
                    {
                        if (collider.transform.parent != null && collider.transform.parent.CompareTag("Catapult"))
                        {
                            canSpawn = false;
                            break;
                        }
                    }

                    colliders = Physics.OverlapSphere(hitPose.position, 0.5f);

                    foreach (Collider collider in colliders)
                    {
                        if (canSpawn && collider.transform.parent.CompareTag("Fort"))
                        {
                            canSpawn = false;
                            break;
                        }
                    }


                    if (canSpawn)
                    {
                        inventoryManager.RemoveItem("Catapult");
                        inventory = inventoryManager.GetInventory();

                        GameObject spawnedPrefab = Instantiate(catapultPrefab, hitPose.position, hitPose.rotation);
                        lastPlacementTime = Time.time;
                        Debug.Log(inventory["Catapult"]);
                    }

                    else
                    {
                        Debug.Log("Can't place a new catapult, there is already one in the range");
                    }
                }
            }
        }
    }
}
```

Slika 2.15 Prikaz funkcije *PlaceAttackers()* iz skripte *ARPlacer.cs*

Isprobavanjem mini igre u ovom stadiju, zaključeno je da je AR svijet drugačije skaliran nego potpuno virtualna scena poput geolokacijske. Ukratko, ne može se sve skalirati kako nam paše, već se treba paziti na odnos veličina objekata sa stvarnim svijetom. Zbog toga je u izborniku *Project* potrebno stvoriti umanjene kopije objekata katapulta i tvrđave. Te kopije će se koristiti isključivo u AR mini igri pa će umjesto uobičajenih *Fort.cs* i *Catapult.cs* skripti na njih biti postavljene nove *ARFort.cs* i *ARCatapult.cs* skripte koje definiraju njihovo ponašanje u mini igri (slika 2.16).




Slika 2.16 Prikaz umanjene kopije objekta Catapult unutar izbornika Inspector

Skripta *ARFort.cs* bavi se baratanjem količinom otpora unutar tvrđave i prikazivanjem količine tog otpora crtom iznad tvrđave. Sustav kojim se služi je veoma jednostavan i potpuno baziran na najobičnijem sustavu životnih bodova (engl. *health points*, skr. HP), stoga bi se moglo reći da količina otpora tvrđave zapravo predstavlja HP tvrđave. S druge strane, skripta *ARCatapult.cs* ima zadatak svakih 5 sekundi, ravno ispred sebe, pseudonasumičnom snagom, ispucati projektil. Kako količina otpora koju će uspješan pogodak oduzeti tvrđavi ovisi o brzini kojom projektil pogodi tvrđavu, razvijena je

*ARProjectile.cs* skripta. Skripta uz pomoć *OnCollisionEnter()* funkcije provjerava ako se sudarila s tvrđavom i ako je, onda zove *TakeDamage()* funkciju iz skripte *ARFort.cs* koja se nalazi na tvrđavi i tako simulira gubitak otpora unutar tvrđave. Kada količina otpora unutar tvrđave padne ispod 0, korisnik ju je uspješno osvojio te aplikacija u *POI.json* označava da je tvrđava pokorena i izlazi iz mini igre. Prilikom povratka u geolokacijsku scenu, zbog tamo implementiranih skripti, korisnika će dočekati poruka koja ga izvještava o uspješnosti njegovog podviga, tj. cjelokupne vježbe.

## 2.5. Izgradnja igrive inačice aplikacije

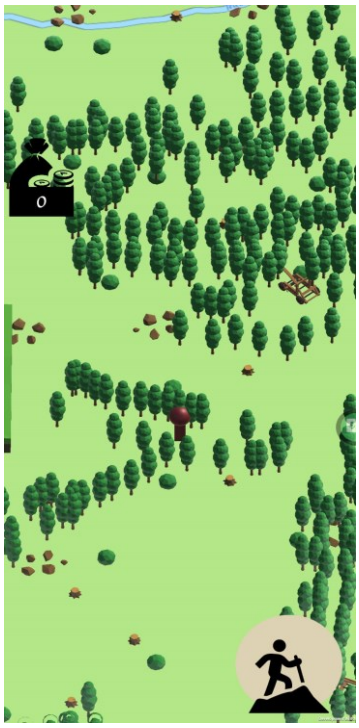
Nakon što su sve scene razvijene, aplikacija je puštena u pogon. Kako bi stvorili igrivu (engl. *playable*) inačicu aplikacije, potrebno je izgraditi (engl. *build*) naš projekt. U *Unityju* unutar izbornika *File*, pritiskom na *Build Settings*, otvaraju se postavke za izgradnju aplikacije. U prozoru *Platform* potrebno je odabrati *Android* i pritisnuti gumb *Switch Platform* kako bi se uvezle biblioteke potrebne za izgradnju *Android* aplikacije. Kako aplikacija za AR koristi dubinsku kameru, uređaji na kojima će se aplikacija pokretati moraju podržavati DepthAPI. Dodatno, da sve radi kako treba, zbog korištenja ARCore-a, potrebno je postaviti minimalnu verziju Android API-ja u izborniku Player Settings. Ulaskom u taj izbornik pronalazi se *Minimum API Level* postavka i pomoću padajućeg izbornika se stavlja na opciju *Android 7.0 'Nougat' (API Level 24)* (slika 2.17).


Slika 2.17 Prikaz postavke *Minimum API Level* iz izbornika *Player Settings*

Sljedeći korak je pritiskom na gumb *Add Open Scenes* dodati sve 3 razvijene scene u prozor *Scenes In Build* kako bi one bile dio izgrađene aplikacije. Nakon što su scene dodane, pritiskom na gumb *Build* otvara se prozor *Windows File Explorera* i bira se direktorij u kojeg će .apk datoteka biti spremljena. Kada je izrada .apk datoteke gotova, datoteku je potrebno prebaciti na mobilni uređaj i instalirati kao i svaku drugu .apk datoteku.

# 3. Prikaz rada aplikacije

Pokretanjem instalirane aplikacije, otvara se scena glavnog izbornika u kojoj korisnik može odabrati započeti igru, proučiti upute kako igrati ili izaći iz aplikacije. Započinjanjem igre, pritiskom na gumb *Start*, otvara se geolokacijska scena. Otvaranjem geolokacijske scene, aplikacija od korisnika traži dopuštenje za korištenje lokacijskih usluga, a kasnije otvaranjem AR mini igre, aplikacija traži dopuštenje za korištenje kamere. Potrebno je aplikaciji pružati oba dopuštenja kako bi ona ispravno funkcionirala. Ukoliko korisnik nema upaljen GPS na svom uređaju ili ako nije dopustio korištenje lokacijske usluge, na virtualnoj karti neće biti ničega, stoga je u uputama kako igrati posebno naglašeno da korisnik mora uključiti GPS na svom uređaju. Dakle, kada su zadovoljeni svi uvjeti za normalan rad aplikacije, korisniku se iscrtava virtualna karta (slika 3.1).




Slika 3.1 Prikaz virtualne karte unutar aplikacije

Kako se korisnik kreće po stvarnom svijetu tako se i na virtualnoj karti kreće model koji predstavlja korisnika. Oko korisnika se svuda po karti stvaraju katapulti koje korisnik pritiskom može sakupiti. Katapulti će mu biti potrebni za osvajanje tvrđava. Što ih više sakupi to mu je lakše osvojiti neku tvrđavu. Korisnik pritiskom na ikonu na donjem desnom kutu ekrana započinje vježbu. Na virtualnoj karti mu se iscrtava ruta koju mora

pratiti. Kada stigne do kraja rute, korisnik će na svom uređaju vidjeti da ga je ruta vodila do virtualne tvrđave, koju on tada može pokušati osvojiti. Pritiskom na tvrđavu, korisniku se pali AR mini igra (slika 3.2).

Slika 3.2 Prikaz AR mini igre unutar aplikacije

Korisnik kamerom skenira prostor oko sebe i kada aplikacija pronađe površine na kojima se mogu stvarati objekti, korisnik prvo u sceni može stvoriti tvrđavu koju napada te tek onda može postavljati sakupljene katapulte. Cilj mu je postaviti katapulte tako da gađaju tvrđavu i tako pobijediti u mini igri. Nakon što je mini igra gotova, ovisno o uspješnosti osvajanja tvrđave, korisnik je nagrađen dukatima koji predstavljaju rezultat (engl. *score*).

## 3.1. Moguća proširenja aplikacije

Aplikaciju je u budućnosti moguće nadograditi na razne načine. Prije svega, važno je napomenuti da aplikacija trenutno radi samo na području grada Zagreba, ali proširenje na druge gradove i naselja je vrlo jednostavno i izvedivo unutar par minuta. Kako bi se aplikacija mogla u potpunosti koristiti u drugim naseljima, jedino što je potrebno napraviti je dodati nove koordinate na kojima bi se nalazile virtualne tvrđave u skriptu *AbstractMap.cs* unutar odjeljka *POINTS OF INTEREST* i u skriptu *POIManager.cs*.

Jedan od jednostavnijih načina nadogradnje je dodatak novog oružja i oruđa za opsadu (engl. *siege tools*) poput balisti ili pokretnih tornjeva. Njihovim dodatkom, u mini igru bi dodali raznovrsnost i natjerali korisnika na veću količinu planiranja i razmišljanja prilikom osvajanja tvrđave. Isto tako, mogli bismo stvoriti više modela tvrđavi i postavljati ih na proceduralno generirane brežuljke. Tako opet tjeramo korisnika na proaktivnije razmišljanje prilikom igranja mini igre. Još jedan koristan dodatak bio bi kada bi se kupila pretplata na Mapbox za neograničeni broj API upita. Tada bi se mogao razviti putokaz poput onog na Google Mapsu, koji ovisno o našoj lokaciji svako malo ažurira iscrtanu rutu. Tako bi korisnici dobili osjećaj navigacije s kojim su upoznati, ali bi se zakomplicirao način spremanja pređenih ruti.

Još jedna nadogradnja bila bi umrežiti igru tako da postoji ljestvica (engl. *leaderboard*) na kojem igraći mogu vidjeti koliko tko ima sakupljenih dukata, pređenih kilometara ili osvojenih tvrđava. To bi omogućilo korisnicima da se međusobno natječu, dijele postignuća i motiviraju jedni druge. Sve u svemu, ova aplikacija pruža brojne mogućnosti za daljnji razvoj i može poslužiti kao odlična osnova za stvaranje sličnih aplikacija.

# Zaključak

Proširena stvarnost, spajanjem virtualnih elemenata i stvarnog svijeta, pruža korisnicima nove načine interakcije s digitalnim sadržajem. Od jednostavnih pomagala poput virtualnog metra, sve do složenih medicinskih i vojnih sustava, primjenama proširene stvarnosti naizgled nema kraja. Kombinacijom proširene stvarnosti i geolokacijskih usluga možemo dobiti aplikacije koje na zabavan način promiču tjelesnu aktivnost i istraživanje.

U ovome radu prikazan je razvoj upravo takve aplikacije. *Squiare* je geolokacijska mobilna igra za poticanje fizičkog vježbanja. Glavni cilj aplikacije je motivirati korisnike na kretanje i pritom im pružiti zabavno iskustvo. Aplikacija korisnicima prikazuje tematsku virtualnu kartu i zadaje im rute za koje očekuje da će korisnici prehodati u određenom vremenskom periodu. Na kraju svake rute na karti se nalazi virtualna tvrđava koju korisnik može osvojiti. Da bi ju osvojio, korisnik mora pobijediti u AR mini igri. Na karti se svugdje po svijetu nalaze virtualni katapulti koje korisnik sakuplja i kasnije koristi u mini igri. Cilj unutar mini igre je postaviti sakupljene katapulte u prostor, tako da gađaju i u konačnici poraze tvrđavu koju korisnik pokušava osvojiti. Uz implementirane funkcionalnosti, postoje i dodatna poboljšanja koja bi obogatila korisničko iskustvo i povećala motivaciju korisnika za korištenje ove aplikacije u svrhu rekreacije.

Proširena stvarnost na mobilnim uređajima još uvijek se razvija i napreduje. Ovisi o napretku u područjima kao što su računalni vid, dubinsko snimanje i drugima. Stoga je potrebno napomenuti da se, prilikom razvoja aplikacija koje koriste proširenu stvarnost, mogu pojaviti neočekivani problemi i izazovi. Potrebno je biti strpljiv i ne gubiti nadu jer se uporabom ovakvih tehnologija mogu postići zaista iznenađujuća rješenja.

# Literatura


[1] Pandžić, I. S., Pejša, T., Matković, K., Benko, H., Čereković, A., Matijašević M. (2011) *Virtualna okruženja: Interaktivna 3D grafika i njene primjene*, Element, Zagreb

[2] NCO, *The Global Positioning System*, poveznica: https://www.gps.gov/systems/gps, pristupljeno: 8. svibnja 2023.

[3] Cointelegraph, *Augmented reality vs. virtual reality: Key differences*, poveznica: https://cointelegraph.com/learn/augmented-reality-vs-virtual-reality-key-differences, pristupljeno: 8. svibnja 2023.

[4] ResearchGate, *The Sword of Damocles by Ivan Sutherland*, poveznica: https://www.researchgate.net/figure/The-Sword-of-Damocles-by-Ivan-Sutherland_fig2_291516650, pristupljeno: 9. svibnja 2023.

[5] García Requejo, What is augmented reality (AR)? Origin and evolution, poveznica: https://garciarequejo.com/en/what-is-augmented-reality-ar-origin-and-evolution/, pristupljeno: 8. svibnja 2023.

[6] Doughty, M., Ghugre, N. R., Wright, G. A.: Augmenting Performance: A Systematic Review of Optical See-Through Head-Mounted Displays in Surgery, poveznica: https://www.mdpi.com/2313-433X/8/7/203, pristupljeno: 17. svibnja 2023.

[7] New Atlas, *Facebook's Oculus Quest 2 takes wireless VR to the next level*, poveznica: https://newatlas.com/vr/facebook-oculus-quest-2-vr-headset/, pristupljeno: 9. svibnja 2023.

[8] ARMarket, *Augmented Reality benefits in medicine*, poveznica: https://www.arealitymarket.com/en/augmented-reality-benefits-in-medicine/, pristupljeno: 17. svibnja 2023.

[9] Bockler, A. *Four reasons Augmented Reality will change assistive technology*, Assistive Technology Blog, poveznica: https://assistivetechnologyblog.com/2017/03/four-reasons-augmented-reality-will-change-assistive-technology.html, pristupljeno: 17. svibnja 2023.

[10] Allan, R. *Pokémon GO usage statistics say it's the most popular mobile game in U.S. history*, poveznica: https://medium.com/@sm_app_intel/pok%C3%A9mon-go-usage-statistics-say-its-the-most-popular-mobile-game-in-u-s-history-ea09ea2bf6df , pristupljeno: 20. svibnja 2023.

[11] Plante, C. *Pokémon Go is an average game, but an astonishing fantasy*, poveznica: https://www.theverge.com/2016/7/11/12130312/pokemon-go-niantic-nintendo-game-virtual-world, pristupljeno: 20. svibnja 2023.

[12] Unity, *Unity User Manual 2021.3 (LTS),* poveznica: https://docs.unity3d.com/Manual/UnityManual.html, pristupljeno: 11. svibnja 2023.

[13] Google, *Depth adds realism*, poveznica: https://developers.google.com/ar/develop/depth, pristupljeno: 22. svibnja 2023.

[14] Microsoft, *Visual Studio*, poveznica: https://visualstudio.microsoft.com/#vs-section, pristupljeno: 11. svibnja 2023.

[15] Mapbox, *Maps SDK for Unity*, poveznica: https://docs.mapbox.com/unity/maps/guides/, pristupljeno: 10. svibnja 2023.

[16] Flof, *Easy Transitions*, Unity Asset Store, poveznica: https://assetstore.unity.com/packages/tools/gui/easy-transitions-225607, pristupljeno: 23. svibnja 2023.

[17] PolyPerfect, *Low Poly Ultimate Pack*, Unity Asset Store, poveznica: https://assetstore.unity.com/packages/3d/props/low-poly-ultimate-pack-54733, pristupljeno: 2. svibnja 2023.

# Sažetak

U ovome radu opisana je tehnologija proširene stvarnosti. Predstavljena je njezina povijest, osnovne karakteristike, načini izvođenja i današnje primjene. Također je izrađen prototip mobilne geolokacijske igre s elementima proširene stvarnosti i opisan je postupak razvoja iste.

Aplikacija *Squiare* razvijena je u Unity razvojnom okruženju uz korištenje Mapbox SDK alata. Cilj aplikacije je motivirati korisnike na kretanje i pritom im pružiti zabavno iskustvo. Opisan je rad aplikacije i predstavljene su ideje za buduće nadogradnje i poboljšanja.

**Ključne riječi:** proširena stvarnost, rekreacija, Android aplikacija, Unity, Mapbox

# Summary

This thesis describes the technology of Augmented Reality. Its history, key features, implementation methods and today's applications are all described. In the scope of the thesis, a prototype of a mobile geolocation game with AR elements has been designed and developed, with the development process described in detail.

The application *Squiare* is developed in the Unity development environment using the Mapbox SDK tools. The aim of the application is to motivate users to move while providing them with an entertaining experience. The usage of the application is described and ideas for future upgrades and improvements are presented.

**Keywords:** Augmented Reality, recreation, Android application, Unity, Mapbox