Mobilna aplikacija s primjenom proširene stvarnosti za vođeno rješavanje slagalica

Radošević, Luka

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1689

MOBILNA APLIKACIJA S PRIMJENOM PROŠIRENE STVARNOSTI ZA VOĐENO RJEŠAVANJE SLAGALICA

Luka Radošević

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1689

MOBILNA APLIKACIJA S PRIMJENOM PROŠIRENE STVARNOSTI ZA VOĐENO RJEŠAVANJE SLAGALICA

Luka Radošević

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1689

Pristupnik:	**Luka Radošević (0036544309)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	prof. dr. sc. Maja Matijašević

Zadatak: **Mobilna aplikacija s primjenom proširene stvarnosti za vođeno rješavanje slagalica**

Opis zadatka:

Proširena stvarnost (engl. Augmented Reality, AR) je skup tehnologija koje, na zaslonu odgovarajućeg uređaja, omogućuju prikazivanje računalno generiranog sadržaja (primjerice, teksta ili 3D objekata) unutar izravnog prikaza stvarnog svijeta. Zadatak je osmisliti, oblikovati i programski razviti mobilnu aplikaciju s primjenom proširene stvarnosti koja korisniku pomaže pri rješavanju (2D ili 3D) slagalice putem kamere pametnog telefona. Vrsta slagalice može biti slobodno odabrana. U prikazu stvarnog svijeta putem kamere pametnog telefona treba omogućiti lociranje i prepoznavanje dijelova slagalice te u taj prikaz ugraditi računalno generirane objekte koji korisniku daju naznake poteza za rješavanje slagalice te ga tako vode do konačnog rješenja. Programsko rješenje izvedite primjenom pogonskog sustava za igre Unity.

Rok za predaju rada: 14. lipnja 2024.

Sadržaj

Uvod

Cilj ovog rada je razvoj mobilne aplikacije s primjenom proširene stvarnosti koja korisniku pomaže pri rješavanju slagalice. Slagalica odabrana za prikaz ovog rada je Rubikova kocka. Izumio ju je ranih 1970-ih godina Erno Rubik, u to vrijeme profesor arhitekture na Fakultetu primijenjenih umjetnosti u Budimpešti u Mađarskoj. Slaganje Rubikove kocke bez uputa je veoma zahtjevno, toliko da je čak i njezinom izumitelju navodno bilo potrebno mjesec dana da ju složi nakon što ju je prvi puta izmiješao [1]. Danas svatko može naučiti kako riješiti slaganje izmiješane Rubikove kocke različitim algoritmima rješavanja dostupnima putem Interneta.

Proširena stvarnost (engl. *augmented reality*, skr. AR) tehnologija je koja omogućuje prikaz računalno generiranih (virtualnih) objekata unutar korisnikovog pogleda na stvarni (fizički) svijet, posredstvom odgovarajućeg korisničkog uređaja [2]. Navedena tehnologija bit će osnova aplikacije na pametnom telefonu, nazvane ARPrime, razvijene u okviru ovog završnog rada. Korisnik aplikacije kamerom pametnog telefona skenira stranice Rubikove kocke, a aplikacija uz pomoć dobivenih informacija stvara rješenje zadanog stanja prema najpoznatijem početničkom algoritmu za slaganje kocke te korisniku pruža upute za slaganje kocke korak po korak.

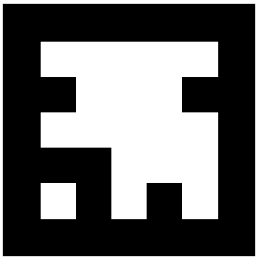
Završni rad podijeljen je u tri poglavlja. Prvo poglavlje opisuje pojam proširene stvarnosti, uz kratki pregled povijesti navedene i srodnih tehnologija. Opis i matematička osnova Rubikove kocke, uz potrebne informacije za razumijevanje algoritama, sadržaj su drugog poglavlja, dok se treće poglavlje odnosi na razvoj aplikacije, analizu zahtjeva aplikacije i opis same implementacije programskog rješenja, uz analizu problema i rješenja tijekom razvoja.

1. Proširena stvarnost

Proširena stvarnost temelji se na kombinaciji stvarnosti s virtualnim svijetom, točnije, postavljanjem interaktivnog virtualnog sloja u prikaz stvarne, fizičke okoline korisnika u stvarnom vremenu. Stvoreni virtualni sloj ima cilj unaprijediti iskustvo stvarnosti prikazom simuliranih objekata. Uporaba navedene tehnologije korisniku proširuje prikaz svijeta uz pomoć uređaja koji funkcionira kao posrednik. Pametni telefoni su neki od uređaja koji koriste AR, a postoje i specijalizirani uređaji poput pametnih naočala, primjerice Microsoft HoloLens 2 (slika 1.1) [3], te Meta Quest Pro [4], koji se zbog dodatnih mogućnosti interakcije svrstavaju i u tehnologiju miješane stvarnosti.

Slika 1.1 Uređaj HoloLens 2, preuzeto s [3]

Proširena stvarnost nalazi primjenu u brojnim sferama, na primjer za razvoj igara, edukativnih sadržaja, interaktivnih uputa, te na mnoge druge načine. Proširena stvarnost često se oslanja na *markere* (primjer na slici 1.2), kao posebne vizualne simbole koji se koriste za prepoznavanje i praćenje objekata u stvarnom vremenu.



Slika 1.2 Primjer AR markera, preuzeto s web.dev [5]

Osnovna podjela proširene stvarnosti je na proširenu stvarnost s markerima (engl. *marker-based AR*) i proširenu stvarnost bez markera (engl. *markerless AR*) [6]. Proširena stvarnost s markerima temelji se na identifikaciji aplikaciji već poznatih slika koje su korištene za postavljanje virtualnih objekata na željenu poziciju sa željenom veličinom ili za detekciju pozicije promatranog objekta. Svrha markera je da djeluje kao referentna točka koja sustavima proširene stvarnosti „dojavljuje“ poziciju, orijentaciju i tip objekta. Implementacija proširene stvarnosti bez markera treba raspoznati što detektira preko boje, oblika ili uzorka promatranog objekta, te je njena izvedba znatno složenija od proširene stvarnosti s markerima.

1.1. Povijesni pregled razvoja proširene stvarnosti

Američki računalni znanstvenik Ivan Sutherland 1968. godine stvara jedan od prvih zaslona nosivih na glavi (engl. *head mounted display*, HMD) s ciljem proširenja stvarnosti dodavanjem virtualnog sloja u stvarni svijet [7]. Taj uređaj prikazan je na slici 1.3. Virtualni sloj prikazan uređajem bio je jednostavan žičani prikaz prostorije.

Slika 1.3 Prvi HMD uređaj (preuzeto iz [7])

Kroz godine AR je počeo biti korišten u svrhe edukacije vojnih pilota Zračnih snaga Sjedinjenih Američkih Država. Proširena stvarnost pokazala se kao veoma koristan alat u edukaciji i treniranju vojnika. Simuliranje korištenja vojne opreme jeftinije je i sigurnije od korištenja te iste opreme uživo [8]. Godine 1998. Sportsvision emitira prvu NFL utakmicu uživo u kojoj se koristi tehnologija proširene stvarnosti. Preko video prijenosa utakmice

dodana je žuta linija koja pokazuje gledateljima korisnu informaciju za bolje praćenje utakmice. Godinu kasnije NASA razvija novi sustav prikazivanja informacija na pilotovom ekranu (slika 1.4), koji je kasnije postao uobičajen u mnogim avionima i helikopterima.

Slika 1.4 Sustav prikazivanja informacija aviona, preuzeto iz [8]

Japanski inženjer Hirokazu Kato 1999. godine stvara biblioteku ARToolKit (https://www.hitl.washington.edu/artoolkit/documentation/), koja je proširenu stvarnost učinila pristupačnijom i jednostavnijom za programsku implementaciju. ARToolKit je biblioteka otvorenog koda za razvijanje aplikacija koje koriste proširenu stvarnost. Omogućava praćenje objekata i postavljanje istih u stvarni svijet uz pomoć markera. Georg Klein i David Murray 2007. godine predstavljaju rješenje koje ne ovisi o markerima [9]. Njihovo rješenje temeljilo se na metodi paralelnog praćenja i mapiranja, kao alternativi simultanoj lokalizaciji i mapiranju (engl. *Simultaneous Localization and Mapping*, skr. SLAM), koja je dobivala podatke iz rukom držane kamere te obradom podataka procjenjivala položaj objekta.

AR tehnologija se 2010-tih godina nastavlja se razvijati iznimnom brzinom, potaknuta napretkom u hardveru, softveru i algoritmima potrebnima za prepoznavanje i praćenje objekata. Prvi eksperimentalni sustav pokazao je koncept miješanja virtualnog sloja s prikazom stvarnog svijeta korisnim i zanimljivim. Napreci prvog desetljeća 21. stoljeća su oblikovali daljnji razvoj tehnologije proširene stvarnosti. Postavili su osnovnu konstrukciju AR aplikacije i uz ARToolKit, te druge nove biblioteke proširene stvarnosti inspirirane ARToolKitom, omogućili su širokom krugu programera i korisnika da upoznaju mogućnosti AR tehnologije.

1.2. Pojmovi proširene, miješane i virtualne stvarnosti

Uz pojam proširene stvarnosti blisko su povezani i pojmovi miješane stvarnosti (MR), te virtualne stvarnosti (VR), no između njih postoje bitne razlike prvenstveno temeljene na načinu interakcije korisnika s virtualnim slojem. Poput proširene stvarnosti, miješana stvarnost temelji se na kombinaciji virtualnog i stvarnog sloja, no za razliku od proširene stvarnosti, miješana stvarnost omogućava fizičku interakciju korisnika s virtualnim slojem prikazanim u stvarnom, fizičkom svijetu. Interakcija korisnika s virtualnim slojem proširuje mogućnosti tehnologije i bitno povećava područje uporabe. Interakcija je moguća pomoću naprednih senzora i pomoću upravljača. Pomoću senzora detektira se položaj ruke korisnika i virtualnog objekta koji prikazuje uređaj, pa korisnik definiranim gestama obavlja željenu radnju, na primjer, pomicanje ili rotaciju objekta. Korištenjem upravljača također se prati položaj, no umjesto geste korišteni su gumbi za interakciju korisnika i virtualnog sloja.

Za razliku od proširene i miješane stvarnosti, virtualna stvarnost smještena je isključivo u računalno generiranom svijetu. Virtualna stvarnost omogućuje korisniku da pomoću pokreta u stvarnome svijetu, praćenih senzorima ili upravljačima, utječe na virtualni svijet u kojem se nalazi. Izgrađeni virtualni svijet ne ovisi o markerima, nego je u potpunosti simuliran oko osobe. Takva tehnologija često se koristi u razvoju računalnih videoigara (slika 1.5), virtualnih iskustava i edukativnih sadržaja.

Slika 1.5 Primjer scene iz VR video igre *Beat Saber*
(Izvor: https://www.vrfitnessinsider.com/beat-saber-ragesaq-explains-darth-maul/)

2. Rubikova kocka

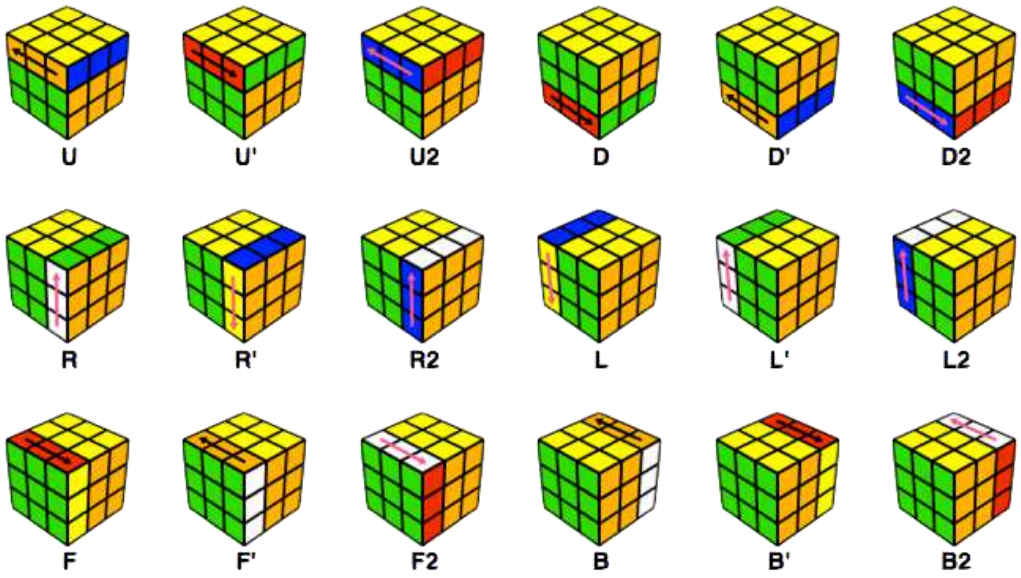
Rubikova kocka (slika 2.1) je trodimenzionalna slagalica koja se sastoji od 8 rubnih dijelova s 3 boje, 12 dvobojnih bridova, te 6 središnjih dijelova, svaki obojan u jednu od 6 boja: plava, bijela, crvena, zelena, narančasta i žuta. Miješanjem kocke moguće je postići više od 43 trilijuna (10^{18}) različitih kombinacija uz samo 18 mogućih različitih poteza.

Slika 2.1 Izvorna Rubikova kocka, preuzeto s [10]

Mogući potezi su okretanje gornje ili donje strane kocke (označene slovima U za *up*, odnosno D za *down*), okretanje lijeve ili desne strane (označene slovima L za *left*, odnosno R za *right*), te okretanje prednje ili stražnje strane kocke (označene slovima F za *front*, odnosno B za *back*) u smjeru kazaljke na satu.

Ostalih 12 poteza su varijacije na ponavljanje istih poteza. Dva ista poteza zaredom neovisno o smjeru kojem se izvode, sve dok su međusobno jednaki, je potez za koji se u standardnoj notaciji dodaje broj 2 uz slovo poteza, npr. R2.

Tri bilo koja ista poteza zaredom u smjeru kazaljke na satu istovjetni su okretanju iste stranice u smjeru suprotnom od kazaljke na satu, zvanom (engl.) „*Prime*“ varijantom, prema kojoj je aplikacija dobila ime. *Prime* se označava jednim navodnikom, npr. R', i čita kao R-*prime* (tj. R-crtano, na hrvatskome). Svi navedeni potezi prikazani su na slici 2.2.



Slika 2.2 Mogući potezi pri slaganju Rubikove kocke, preuzeto s [11]

2.1. Matematička osnova

Matematička osnova Rubikove kocke je grupa. U matematici, grupa je skup elemenata nad kojim vrijede 4 aksioma: zatvorenost, asocijativnost, postojanje neutralnog elementa i postojanje inverznog elementa [12]. Rubikova kocka poštuje ova 4 pravila što pokazuje da je i ona sama matematička grupa. Pravilo zatvorenosti glasi da za svaki x i y koji su elementi grupe G, rezultat njihovih međusobnih operacija je također element grupe G. U slučaju Rubikove kocke to je točno, jer ne postoji standardni potez koji bi mogao na neki način učiniti kocku, primjerice, nerješivom.

Drugi aksiom, asocijativnost, zahtjeva da bilo koje operacije iste težine (npr. zbrajanje i oduzimanje), neovisno o redoslijedu odnosno postavljanju zagrada, uvijek daju isti rezultat. Asocijativnost je na kocki vidljiva ako tretiramo parove poteza koji se međusobno ne sijeku, npr. U i D, odnosno okretanje gornje i donje stranice kocke kao jednaku „težinu“ operacija, zato što okretanjem gornje strane jednom, donje dva puta i gornje još dva puta dovede do jednakog stanja kocke kao i bilo koji drugi redoslijed tih poteza. Okretanje stanica koje se međusobno sijeku, npr. U i F, isto se ponaša kao i miješanje operacija različitih „težina“ na što utječe redoslijed izvođenja operacija, odnosno postavljanje zagrada.

Svaka grupa mora imati neutralni element, odnosno element koji pripada grupi takav da operacije tog elementa i nekog elementa *x* kao rezultat daju element *x*, npr. pri zbrajanju i oduzimanju to je broj nula. Dodatkom praznog poteza u skup od prije definiranih 18 mogućih poteza kocke, kocka zadovoljava i taj aksiom grupe, iako se prazan potez ne dodaje u standardnu notaciju iz očitih razloga.

Posljednji aksiom je postojanje inverznog elementa u grupi. Za svaki element iz grupe postoji inverzni element koji operacijom daje neutralni element. Četiri ista poteza za redom uvijek daju neutralni element u slučaju kocke, ali i za svaki potez postoji i njegov inverzan potez. U slučaju poteza okretanja stranice za 90°, inverzni potez je *prime* (oznaka ') odnosno normalna varijanta tog poteza, dok je dvostruki potez sam sebi inverzni potez.

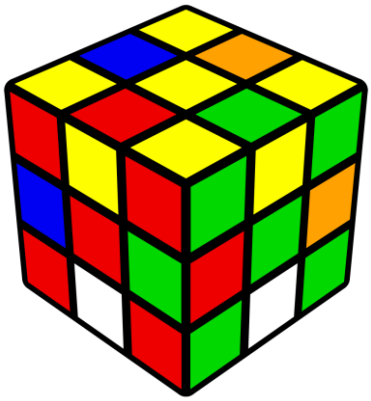
Sva moguća stanja kocke od početnog složenog stanja mogu biti povezana u usmjereni graf, gdje je svaka točka stanje kocke, a linije koje te točke povezuju su potezi, odnosno, inverzni potezi ako u slučaju da idemo u suprotnom smjeru. Za pronalazak optimalnog rješenja svakoga stanja kocke korišten je takav graf i računalni algoritam pretraživanja u širinu (engl. *breadth-first search*, skr. BFS).

2.2. Programsko rješenje

Ubrzo nakon što je Rubikova kocka postigla svjetsku slavu, programere je zainteresirao tzv. problem Božjeg broja (engl. *God's Number*). Problem se odnosi na pitanje koliko bi najviše poteza bilo potrebno Bogu, odnosno nekom sveznajućem biću, da riješi bilo koje zadano stanje kocke. Potraga je započela definiranjem gornje i donje granice broja poteza, te je u početku bila relativno brza, smanjivanjem gornje granice za nekoliko poteza svakih nekoliko godina. Donja granica je odmah bila poznata kao broj 18, zato što je 18 najmanji broj poteza s kojim se može postići prije navedenih $43 \cdot 10^{18}$ permutacija [13].

Uz ručne pokušaje i razne programe, granice su mijenjane sve do 2010. godine kada je, na temelju računalne analize svih mogućih stanja, dokazano da su gornja i donja granica jednake broju 20. Vrijeme za pronalazak rješenja za svih približno 10^{20} stanja BFS algoritmom (uz tadašnje mogućnosti računanja) procijenjeno je na nekoliko milijuna godina. Podjelom problema na više milijardi manjih problema te optimizacija BFS-a na njihov specifični slučaj učinila je tu potragu mogućom. Prije pronalaska dokazane

vrijednosti gornje granice, gornja granica bila je postavljena na 22, dok je donja granica bila podignuta na 20, zbog pronalaska jedne pozicije za koju je trebalo najmanje 20 poteza, te se ta pozicija naziva *superflip* [13] (slika 2.3).



Slika 2.3 *Superflip* pozicija, prikaz s pomoću *VisualCube* [14]

Bitna programska optimizacija došla je u obliku poboljšanja BFS-a. BFS algoritam obilazi graf „nivo po nivo“. Svakim potezom broj stanja koje BFS algoritam posjećuje rastao je eksponencijalno, što je znatno utjecalo na brzinu izvođenja pretrage. Taj problem je riješen tako da se provedu dvije BFS pretrage. Obje pretrage imale su zadanu maksimalnu dubinu deset (10). Maksimalna dubina 10 odabrana je s ciljem da se provjeri je li tadašnja donja granica od 20 poteza jednaka gornjoj granici.

Prva pretraga započinje na složenoj varijanti kocke, te prolazi kroz svako stanje do dubine 10. Druga pretraga započinje na izmiješanoj permutaciji koju provjeravamo i također prolazi kroz svako stanje do zadane maksimalne dubine. Nakon izvršavanja obje pretrage, rješenje bi bio niz dug 20 poteza, koji bi se sastojao od 10 poteza prve i 10 poteza druge pretrage. Korištenjem dvije pretrage pregledava se „samo“ oko 10^{10} stanja, umjesto oko 10^{20} stanja.

U slučaju da se za ikoje početno stanje kocke nije pronašlo rješenje na taj način, donja granica bi bila pomaknuta na 21 ili više, no to se nije dogodilo. Tom završnom pretragom, koja je trajala 35 procesorskih godina preko više računala dokazano je da je 20 najveći broj poteza za riješiti kocku. Detaljan pregled kretanja gornje i donje granice dostupan je u tablici 2.1., preuzetoj sa stranice [13].

Tablica 2.1 Rezultati pretrage, preuzeto s [13]

Godina	Donja granica	Gornja granica	Razlika	Postignuće
1981	18	52	34	Morwen Thistlethwaite
1990	18	42	24	Hans Kloosterman
1992	18	39	21	Michael Reid
1992	18	37	19	Dik Winter
1995	18	29	11	Michael Reid
1995	20	29	9	Michael Reid dokazuje "superflip"
2005	20	28	8	Silviu Radu
2006	20	27	7	Silviu Radu
2007	20	26	6	Dan Kunkle i Gene Cooperman
2008	20	25	5	Tomas Rokicki
2008	20	23	3	Tomas Rokicki i John Welborn
2008	20	22	2	Tomas Rokicki i John Welborn
2010	20	20	0	Tomas Rokicki, Herbert Kociemba, Morley Davidson i John Dethridge

2.3. Postupak rješavanja

U usporedbi s Božjim algoritmom, postoje mnogi, manje efikasni, algoritmi rješavanja Rubikove kocke. Najpoznatiji su CFOP (engl. *Cross, First 2 Layers, Orientation, Permutation*) Jessice Fridrich [15][16], Roux (prema prezimenu autora Gillesa Rouxa) [17] te početnička metoda [18][19], na kojoj se temelji algoritam korišten pri izradi ovog završnog rada. CFOP i početnička metoda zasnivaju se na jednakom principu rješavanja, dok CFOP zahtjeva više pamćenja mogućih stanja, što nadoknađuje manjim brojem potrebnih poteza za dobivanje rješenja [16].

Algoritmi rješavanja se zasnivaju na konceptu komutatora, koji proizlazi iz činjenice da je Rubikova kocka matematička grupa. Osnovna ideja komutatora je zamjena određenih dijelova kocke bez remećenja već složenog dijela. To je moguće postići time da

se učini željena promjena za jedan dio, okrene se jedan od slojeva koji se nije okretao za vrijeme prvih promjena, zatim se ponove inverzni potezi u suprotnom redoslijedu.

Početnička metoda je veoma jednostavna, te zahtjeva pamćenje samo nekoliko jednostavnih algoritama. Zasniva se na ideji rješavanja kocke „sloj po sloj.“ Pri procesu rješavanja važno je shvatiti da se kocka ne rješava „stranicu po stranicu,“ odnosno da sve boje koje su na pojedinom elementu, odnosno kockici, koju netko pokušava staviti na ispravno mjesto moraju odgovarati boji središnje kockice (polja) na stranici na kojoj se nalaze. Početni koraci uglavnom se mogu riješiti intuitivno, dok je na završnim koracima trećeg sloja potrebno koristiti algoritme, specifično komutatore, da se složi ostatak kocke, a da se pritom ne naruše prije složeni slojevi.

3. Razvoj aplikacije *ARPrime*

U ovome poglavlju opisani su ciljevi razvoja aplikacije, postupak samog razvoja i uočeni problemi, te kompromisi koji su bili rješenja za te probleme. Također su opisani i analizirani zahtjevi aplikacije uz dizajn i implementaciju potrebnih sustava koji zadovoljavaju te zahtjeve.

3.1. Cilj i motivacija aplikacije

Cilj aplikacije „ARPrime“ je pojednostaviti proces učenja rješavanja Rubikove kocke koristeći tehnologiju proširene stvarnosti. Motivacija za razvoj aplikacije dolazi iz želje da se korisnicima omogući interaktivno i intuitivno iskustvo učenja rješavanja kocke, uz mogućnost povratka na prijašnje korake, te ponovno skeniranje u slučaju da pogriješe u rješavanju i ne znaju se vratiti na točan put. Nakon svakog poteza, korisnik na temelju stanja u aplikaciji može vidjeti je li pogriješio te ima šansu pokušati se samostalno ispraviti, ili ponovno skenirati kocku ako je potrebno.

Uporaba odabranih tehnologija dolazi s mnogo prednosti. Korištenje tehnologije proširene stvarnosti dopušta vizualizaciju prije spomenutih mogućih poteza označenih standardnom notacijom. Uz tu vizualizaciju pojednostavljuje se korisnikovo povezivanje stranice i poteza, odnosno shvaćanje utjecaja koji svaki potez korisnika ima nad stanjem kocke. Također uz poboljšanje vizualizacije, konstantna mogućnost samostalnog provjeravanja točnosti poteza korisniku ubrzava učenje i shvaćanje učinka svakog poteza.

Za razvoj aplikacije „ARPrime“ nisu korišteni markeri zbog želje da aplikacija bude uporabljiva sa svakom izvornom Rubikovom kockom, bez potrebe za modifikacijom. S druge strane, izbjegavanjem markera ograničava se kvaliteta praćenja kocke u prostoru pri vizualizaciji poteza, što je nadoknađeno korištenjem biblioteke OpenCV.

3.2. Korištene tehnologije

Aplikacija je razvijena primjenom pogonskog sustava za igre Unity [20]. Unity je sveobuhvatno razvojno okruženje koje omogućava izradu igara, interaktivnih 3D sadržaja, aplikacija proširene i virtualne stvarnosti i raznih drugih iskustava. Prva verzija Unityija izdana je 2005. godine, potom je do 2015. godine izdano pet verzija, a u razdoblju od 2017. do 2023. godine slijede godišnja izdanja. Unity podržava distribuciju na mnoge platforme, od računala do igraćih konzola i mobilnih telefona raznih operacijskih sustava. Za razvoj aplikacije korišten je Microsoftov objektno orijentirani programski jezik C#, dok je samo razvojno okruženje pisano u programskom jeziku C++. Unity je odabran kao razvojno okruženje ovoga rada zbog dostupnosti kvalitetne dokumentacije, te veoma aktivne baze korisnika. Za potrebe korištenja AR tehnologija, instalirani su paketi „AR Foundation“ i „XR Core“, te njihovi povezani paketi.

U razvojnom okružju Unity korištena je pozadina za skriptiranje (engl. *scripting backend*) IL2CPP (engl. *Intermediate Language To C++*) koja pretvara C# kôd u C++ kôd korišten za stvaranje izvršne datoteke za odabranu platformu [21]. Takvo kompiliranje koda naziva se AOT (engl. *ahead-of-time*), odnosno prijevremeno kompiliranje. Za primjenu jezika C# za razvoj programa koriste se različita razvojna okruženja, kao što su Visual Studio, Visual Studio Code, JetBrains Rider i MonoDevelop.

Umjesto markera korištena je biblioteka otvorenog koda OpenCV za računalni vid i strojno učenje [22]. OpenCV nudi više od 2500 optimiziranih algoritama koji se mogu koristiti u svrhe detekcije lica, identifikacije objekata, praćenje objekata, prepoznavanje okoliša, te mnoge druge. Navedena biblioteka je korištena zbog mogućnosti detekcije, te kategorizacije boja i njihove pozicije, što omogućava jednostavno spremanje navedenih vrijednosti u korisnom obliku (slika 3.2).

Slika 3.2 Prikaz prepoznavanja boja

3.3. Analiza zahtjeva

Prije razvoja aplikacije potrebno je proučiti potrebne, odnosno željene specifikacije programskog rješenja problema. U idućim poglavljima opisani su funkcionalni i nefunkcionalni zahtjevi aplikacije, koje je potrebno ispuniti za razvoj kvalitetnog programskog rješenja.

3.3.1. Funkcionalni zahtjevi

Funkcionalni zahtjevi su izjave o uslugama koje programski proizvod mora pružati, kako će sustav reagirati na određeni ulazni poticaj te kako bi se trebao ponašati u određenim situacijama [23]. Za aplikaciju „ARPrime“ funkcionalni zahtjevi odnose se na: detekciju stranica kocke, algoritam za rješavanje skeniranog stanja, te prikaz rješenja u stvarnom vremenu korak po korak.

Detekcija stranica kocke je temelj aplikacije. Skeniranje stranica predstavlja ulaz u aplikaciju, te čitajući ulazne podatke programsko rješenje ih zapisuje u šest matrica (3x3) tekstualnog tipa, gdje svako polje predstavlja jednu od 9 boja skenirane stranice kocke. Stranica kocke je prepoznata po boji polja u njezinoj sredini, zato što su te boje uvijek u

istom međusobnom odnosu, što je temelj za redoslijed upisivanja iščitanih podataka u matrice stanja.

Algoritam za rješavanje predstavlja početničku metodu slaganja Rubikove kocke. Prije korištenja algoritma provedena je provjera je li skenirano stanje kocke moguće, a ako nije, korisnik bi trebao biti naveden da ponovno skenira kocku.

Prikaz rješenja u stalnom vremenu koristi grafike koje pokazuju potez koji je potrebno učiniti u tom trenutku. Prikaz u stalnom vremenu poput skeniranja prepoznaje stranicu preko središnje boje.

Navedeni funkcionalni zahtjevi predstavljaju temelj mogućnosti korištenja aplikacijom. Implementacija navedenih zahtjeva će osigurati potrebnu funkcionalnost koja pruža mogućnosti korisniku da skenira izmiješanu kocku, dobije njeno rješenje i složi ju uz pomoć vizualizacije niza poteza koji vode do rješenja.

3.3.2. Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi definiraju atribute performansi programskog rješenja [23]. Takvi zahtjevi osiguravaju pouzdanost, dostupnost i performanse, te kvalitetno korisničko iskustvo primjene softvera.

Potrebno je osigurati kvalitetno praćenje i detekciju stranica kocke u različitim osvjetljenjima i osigurati da korisnik može jasno vidjeti što je aplikacija skenirala, te da ima mogućnost korigirati netočan ulaz.

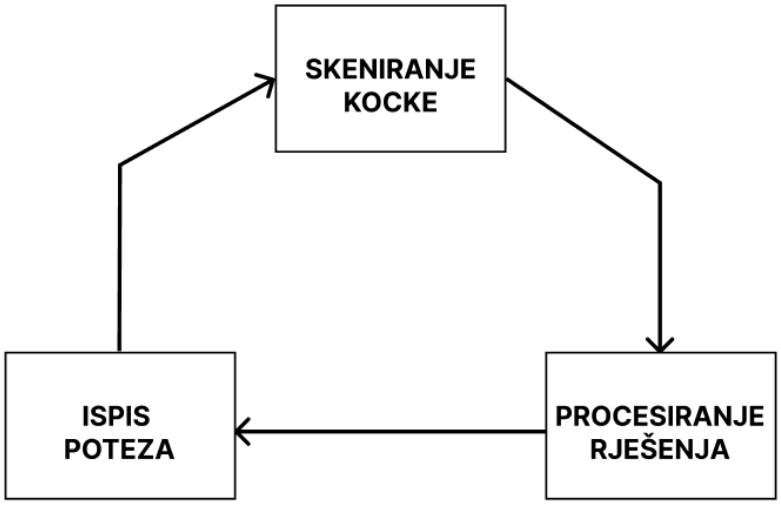
Algoritam za rješavanje mora riješiti stanje kocke u manje od nekoliko sekundi te rješenje mora biti skraćeno, odnosno trebaju se izbaciti nepotrebni potezi. Četverostruka ponavljanja istog poteza moraju se maknuti, zato što su ekvivalent praznom potezu, točnije tome da nije potrebno učiniti ništa. Trostruka ponavljanja moraju biti zamijenjena sa svojom inverznom varijantom. Rješenje kocke dobiveno tim algoritmom također mora biti jednostavno za prikazati da ga korisnik može razumjeti, te mora uvijek biti jasno na kojem potezu se postupak trenutno nalazi.

Grafike prikaza rješenja u stvarnom vremenu korisniku moraju biti jasne i jednostavne za interpretaciju. Uz kvalitetu grafika, važno je da je korisniku jasno na kojem koraku rješavanja se nalazi u svakom trenutku.

Aplikacija također treba biti stabilna, jednostavna za unaprjeđivanje, građena modularno i pristupačna, te intuitivna. Ispunjavanjem nefunkcionalnih zahtjeva gradi se kvalitetno iskustvo za korisnika i osigurava se upotrebljivost i pouzdanost programskog rješenja.

3.4. Dizajn aplikacije

Čitava aplikacija smještena je u jednu Unity scenu, u kojoj se nalaze svi elementi. Pokretanjem aplikacije korisnik započinje glavnu petlju korištenja. Glavna petlja aplikacije sastoji se od tri stanja: skeniranje kocke, procesiranje rješenja i korisnikovog rješavanja uz pomoć generiranih poteza. Navedena stanja povezana su ciklički, odnosno svaki put kada se skenira kocka, petlja započne od početka, što je vidljivo na slici 3.3.



Slika 3.3 Tok glavne petlje

Postoje dva načina prikaza aplikacije, aktivni i pasivni način prikaza. Aktivni način prikaza aplikacije dopušta skeniranje kocke, rješavanje i prikaz generiranih poteza putem grafike na kocki. Pasivni način prikaza aktivira se samostalno kada je mobilni uređaj postavljen na ravnu površinu, detektirajući žiroskopsku rotaciju uređaja.

U pasivnom načinu prikaza vidljivi su potezi u standardnoj notaciji koji se mogu listati pritiskom na strelice, te je vidljiva trodimenzionalna virtualna Rubikova kocka koja predstavlja trenutno stanje stvarne kocke nakon odabranog poteza. Pasivni način dodan je

kao međukorak između samostalnog rješavanja kocke i asistencije koju pruža aktivni način rada.

3.5. Implementacija

Sama kocka modelirana je kao šest polja (stranice kocke) prikazanih u kôdu 3.1. Svaki element u polju je jedno slovo koje predstavlja boju stranice kocke u toj poziciji. Referentna pozicija kocke je s žutom stranom prema gore, što je prikazano pri skeniranju.

```
string[,] yellowFace = { { "y", "y", "y" }, { "y", "y", "y" }, {
"y", "y", "y" } };
string[,] blueFace = { { "b", "b", "b" }, { "b", "b", "b" }, { "b",
"b", "b" } };
string[,] orangeFace = { { "o", "o", "o" }, { "o", "o", "o" }, {
"o", "o", "o" } };
string[,] whiteFace = { { "w", "w", "w" }, { "w", "w", "w" }, {
"w", "w", "w" } };
string[,] redFace = { { "r", "r", "r" }, { "r", "r", "r" }, { "r",
"r", "r" } };
string[,] greenFace = { { "g", "g", "g" }, { "g", "g", "g" }, {
"g", "g", "g" } };
```

Kôd 3.1 Polja koja prikazuju stanje stranica kocke

Svaki mogući potez modeliran je kao metoda koja izmjenjuje dijelove jedne stranice kocke s dijelovima druge stranice kao što se to događa u stvarnosti. Prikaz modeliranja poteza U vidljiv je u kôdu 3.2. U programskoj implementaciji postoji i potez rotacije cijele kocke oko osi *y* u smjeru kazaljke na satu uz 18 osnovnih poteza. Potez rotacije kocke olakšava shvaćanje korisniku što čini potez bitnim, točnije što potez radi.

```
for (int i = 0; i < 3; i++)
{
    string temp = blueFace[0, i];
    blueFace[0, i] = redFace[0, i];
    redFace[0, i] = greenFace[0, i];
    greenFace[0, i] = orangeFace[0, i];
    orangeFace[0, i] = temp;
}
string temp1, temp2, temp3, temp4;
temp1 = yellowFace[0, 0];
temp2 = yellowFace[0, 1];
temp3 = yellowFace[0, 2];
```

```
temp4 = yellowFace[1, 2];
yellowFace[0, 0] = yellowFace[2, 0];
yellowFace[0, 1] = yellowFace[1, 0];
yellowFace[0, 2] = temp1;
yellowFace[1, 0] = yellowFace[2, 1];
yellowFace[1, 2] = temp2;
yellowFace[2, 0] = yellowFace[2, 2];
yellowFace[2, 1] = temp4;
yellowFace[2, 2] = temp3;
```

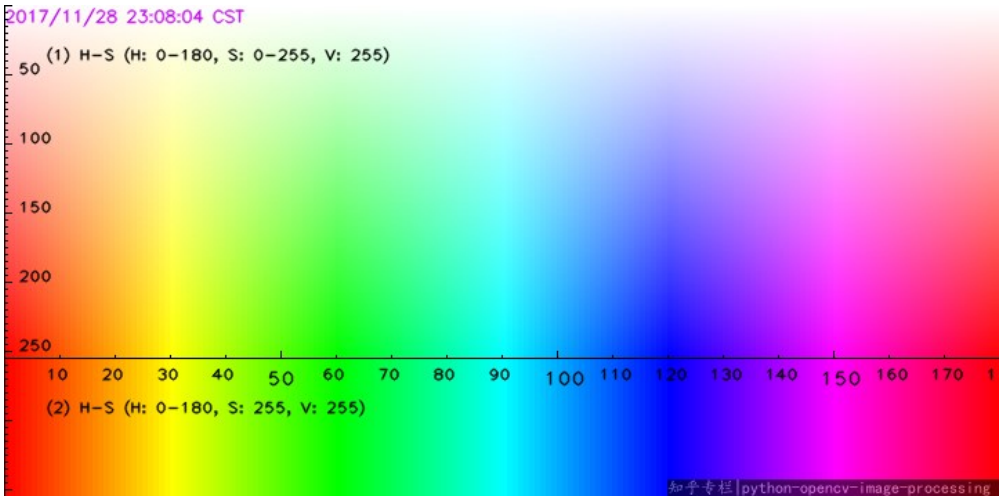
Kôd 3.2 Okretanje gornje stranice (U) kocke u smjeru kazaljke na satu

3.5.1. Interpretacija skenirane stranice kocke

Biblioteka OpenCV korištena je za interpretaciju skeniranog sadržaja stranice kocke. Prije programske analize stranice kocke potrebno je optimizirati uvjete s ciljem da povećamo kvalitetu interpretacije skeniranog. Prvo poboljšanje uvjeta je postignuto korištenjem izvorne Rubikove kocke ili bilo kakve kocke koja ima jasne rubove drugačije boje od šest osnovnih boja. Jasni rubovi znatno olakšavaju računanje površine koju boja zauzima, te sprječavaju kombiniranje polja istih boja u jednu veću površinu u slučaju da se elementi (polja) iste boje nađu jedan pored drugog. Idući faktor na koji treba pripaziti je kvaliteta osvjetljenja. Skeniranje kocke u dobrim uvjetima osvjetljenja, gdje su svi elementi jednako osvjetljeni, ali nisu pod direktnim svjetlom koje bi moglo dovesti do zrcaljenja, iznimno je bitno za točnost. Nakon fizičkih poboljšanja ulaza, moguće je i programski utjecati na kvalitetu skeniranih podataka. Kvalitetno definiranje vrijednosti boja, točnije raspon nijanse, zasićenja i svjetline boje (HSV vrijednost) komponente omogućava detekciju u širem spektru uvjeta osvjetljenja (slika 3.4).

```
if (name == "blue") {
    setHSVmin (new Scalar (92, 50, 50));
    setHSVmax (new Scalar (132, 255, 255));
    setColor (new Scalar (0, 0, 255));
}
```

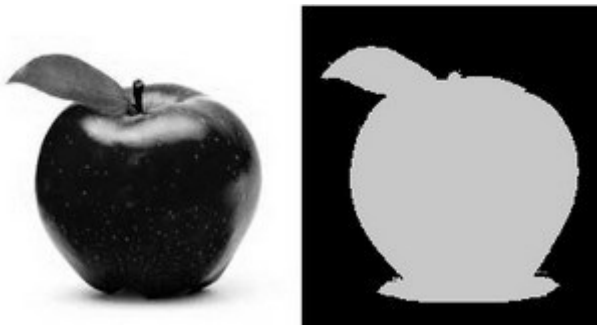
Kôd 3.3 Postavljanje plave boje



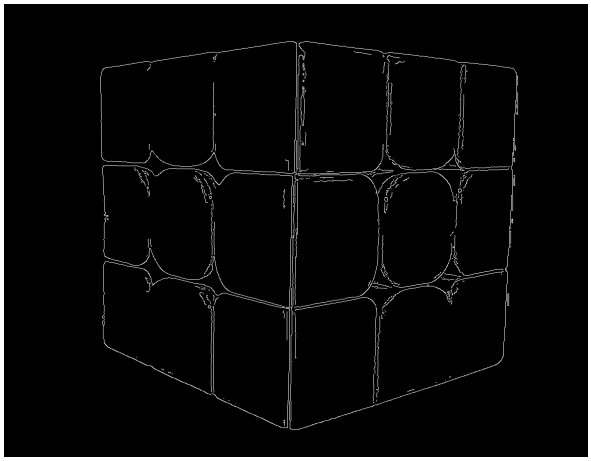
Slika 3.4 HSV tablica (izvor: Stack Exchange, https://i.sstatic.net/gyuw4.png)

Postavljanjem HSV minimuma i maksimuma neke boje, poput plave u kôdu 3.3, zapravo određujemo kvadar boje, odnosno HSV vrijednosti koje spadaju pod tu boju po našoj definiciji. HSV vrijednosti korištene u aplikaciji dobivene su kroz ručno testiranje u raznim uvjetima osvjetljenja, bilo to žuto ili bijelo umjetno svjetlo ili pak prirodno svjetlo, da bi se osigurala bolja kvaliteta skeniranja.

Nakon odabira HSV vrijednosti koje najbolje odgovaraju za šest boja Rubikove kocke, vrši se obrada svake sličice koju snima kamera, ograničena na 30 sličica po sekundi. Uz ograničenje broja slika po sekundi koje program obrađuje, ograničen je i dio ulazne slike kamere koji se skenira na središnji kvadrat dimenzija 300 piksela, indiciran grafikom koja predstavlja prostor skeniranja. Prije praćenja i detekcije boja izračunat je prag boja (engl. *threshold*) za svaku od šest boja kocke, čiji je cilj odvojiti skenirane piksele koji su potrebni za praćenje od onih koji nisu potrebni, točnije odvojiti boje koje program prati od ostalih. Praćene boje u pragu su prikazane bijelom bojom, dok su ostale prikazane crnom bojom. Računanje slike praga je jednostavno. Vrijednost boje svakog piksela slike uspoređena je s vrijednosti praga, u slučaju da vrijednost spada u zadani raspon, piksel je obojan bijelo, a u suprotnome je obojan u crno. Na slici 3.5 prikazan je izgled izračunatog praga boja za fotografiju jabuke, gdje je jasno vidljiva jabuka naspram pozadine, a na slici 3.6 prikazan je prag Rubikove kocke.



Slika 3.5 Izračunat prag slike jabuke, preuzeto s [24]



Slika 3.6 Izračunat prag slike Rubikove kocke, preuzeto s [25]

Nakon podjele boja provjerava se je li njihova površina veća od zadane minimalne površine s ciljem uklanjanja smetnji, odnosno šuma koji nastaje zbog ograničene kvalitete kamere ili refleksije. Zatim se za svaki detektirani objekt, ako ih je devet, računaju *x* i *y* koordinate na ekranu, te se preko tih koordinata zaključuje u koju poziciju dvodimenzionalnog polja je potrebno spremiti koju vrijednost, što je vidljivo u kôdu 3.4. Polje upisa detektirane boje izračunato je provjerom u kojoj od 9 segmenata zone koja detektira boju pripadaju njene koordinate.

```
if (area > MIN_OBJECT_AREA)
{
    ColorObject colorObject = new ColorObject();
    colorObject.setXPos((int)(moment.get_m10() / area) +
offsetX);
    colorObject.setYPos((int)(moment.get_m01() / area) +
offsetY);
```

```
        colorObject.setType(theColorObject.getType());
        colorObject.setColor(theColorObject.getColor());

        colorObjects.Add(colorObject);
        colorObjectFound = true;

        int gridX = (colorObject.getXPos() - offsetX) * 3 /
threshold.cols();
        int gridY = (colorObject.getYPos() - offsetY) * 3 /
threshold.rows();
        if (gridX >= 0 && gridX < 3 && gridY >= 0 && gridY < 3)
        {
            detectedColors[gridY, gridX] =
colorObject.getType().Substring(0, 1);
        }
    }
    else
    {
        colorObjectFound = false;
    }
```

Kôd 3.4 Zapisivanje detektiranih boja u polje

Kada je neki obojani objekt pronađen, iscrtava se točka u njegovom središtu uz obojani tekst imena skenirane boje, koji će biti iznimno vidljiv u slučaju pogrešnog skeniranja. Kôd crtanja objekta vidljiv je u kôdu 3.5. Pozivom metode crtanja objekta na ulaznu sličicu se ispisuje točka i tekst, te nakon ispisivanja svih detektiranih boja sličica se postavi kao ono što korisnik vidi na uređaju.

```
private void drawObject(List<ColorObject> theColorObjects,
Mat frame)
{
    for (int i = 0; i < theColorObjects.Count; i++)
    {
        Imgproc.circle(frame, new
Point(theColorObjects[i].getXPos(),
theColorObjects[i].getYPos()), 10,
theColorObjects[i].getColor());
```

```
        Imgproc.putText(frame, theColorObjects[i].getType(),
new Point(theColorObjects[i].getXPos(),
theColorObjects[i].getYPos() - 20), 1, 2,
theColorObjects[i].getColor(), 2);
    }
}
```

Kôd 3.5 Iscrtavanje detektiranog objekta

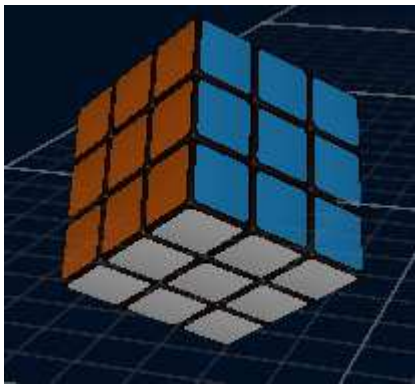
Zapisivanjem i praćenjem stranica kocke u stvarnom vremenu upotpunjena je interpretacija stranica kocke. Polja koja predstavljaju stanje kocke dobiju vrijednosti detektirane pomoću metode postavljanja stranice vidljive u kôdu 3.6. Spremajući vrijednost skenirane stranice, provjerava se vrijednost središnjeg elementa, što određuje koja stranica će biti promijenjena. Korisniku je rezultat skeniranja vidljiv preko postavljenih sličica u boji na korisničkom sučelju, te ako je neka od vrijednosti netočna, korisnik može ponovno skenirati istu stranicu.

```
tempFace = scanner.GetDetectedFace();
faceScannerVisual.SetActive(true);
if (tempFace[1, 1] != facesToScan[currentFaceIndex])
{
    //throw error
    return;
}
switch (tempFace[1, 1])
{
    case "y":
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                yellowFace[i, j] = tempFace[i, j];
            }
        }
        break;
}
```

Kôd 3.6 Postavljanje vrijednosti stranice žute boje

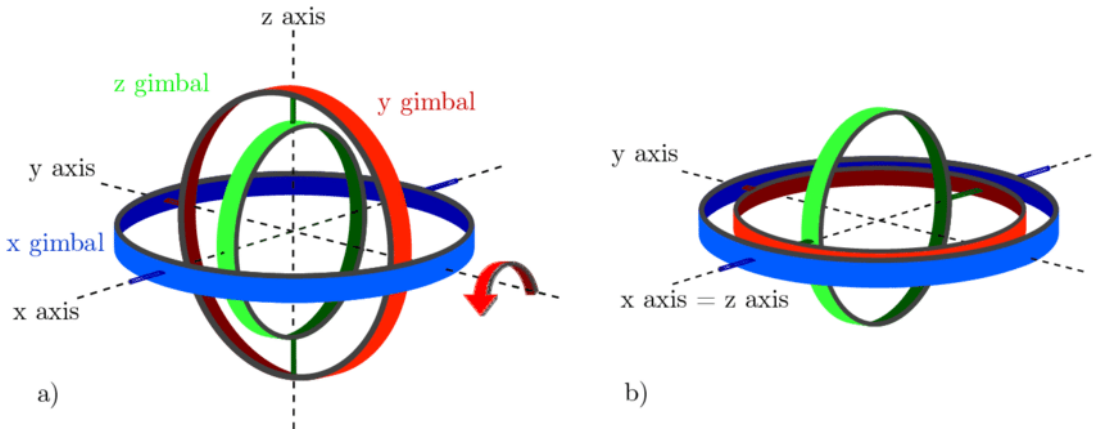
3.5.2. Virtualna kocka

Pasivni način prikaza aplikacije, u svrhu vizualizacije, ima virtualnu 3D Rubikovu kocku koja prikazuje trenutno stanje kocke u odabranom koraku rješavanja. Korisnik može rotirati kocku po želji, te se kocka mijenja ovisno o učinjenom potezu, a ne ovisno o učitanom stanju, što omogućuje implementaciju kompletno virtualnog rješavanja. 3D model kocke modeliran je s pomoću aplikacije Blender i vidljiv je na slici 3.7.



Slika 3.7 3D model Rubikove kocke

Kocka se sastoji od 26 vidljivih kockica i 1 središnje kockice, te 6 skrivenih ploha. Svaka od 26 vidljivih kockica ima na sebi jednu, dvije ili tri boje, ovisno o tipu dijela kocke (brid, vrh). Geometrijski centar svih kocki je u centru Rubikove kocke, poput stvarne igračke. Rotacija Eulerovim kutovima zasniva se na rotaciji objekta oko koordinatnih osi trodimenzionalnog sustava za neka tri kuta. Kvaternioni su proširenje kompleksnih brojeva na četverodimenzionalni prostor. Često se koriste za rotaciju objekata u trodimenzionalnom prostoru jer pružaju numerički stabilan, efikasan i predvidiv način za takve transformacije. Konkretno, rotacija objekta oko zadanog normaliziranog vektora koristi se jedinstvenim kvaternionom koji predstavlja tu rotaciju. Rotiranje kocke je učinjeno uz pomoć kvaterniona umjesto Eulerovih kutova zbog problema zaključavanja gimbal stabilizatora. Navedeni problem nastaje kada se dvije od tri osi rotacije usklade, što dovodi do gubitka jedne osi rotacije kao što je prikazano na slici 3.8b. Posljedica tog problema očita je pri višestrukoj rotaciji jedne od kockica. Nakon što se neke dvije osi usklade, sljedeća rotacija može izbaciti kockicu iz modela ili ju rotirati oko same sebe.



Slika 3.8 a) slobodni Eulerovi kutovi; b) zapeli Eulerovi kutovi – poravnate osi *x* i *z* [26]

Svaka od šest skrivenih ploha kocke nalazi se u jednoj od šest stranica kocke s ciljem da iščita stanje odabrane stranice. Plohe imaju komponentu sudarača (*collider*) koja detektira kojih su 9 kockica, koje također imaju komponente *collider* i *rigidbody*, na strani koja treba biti rotirana. Središnja kockica određene strane odabrana je kao roditelj ostalih kockica. Rotacijom roditeljske središnje kockice, ta transformacija vrši se i nad osam kockica-djece, kao što je vidljivo u kôdu 3.7. Ovakvo rješenje dopušta jednostavan i predvidiv način modeliranja virtualne kocke koji može biti korišten u više različitih svrha. Virtualna kocka ima implementirane metode koje predstavljaju svaki mogući potez koji korisnik može učiniti s kockom.

```
GameObject middleChild = null;
foreach (GameObject child in Uplane.GetCollidingObjects())
{
    if (middleChild == null)
    {
        if (child.GetComponent<MiddlePiece>())
        {
            middleChild = child;
            break;
        }
    }
}
foreach (GameObject child in Uplane.GetCollidingObjects())
{
    if (child != middleChild)
    {
        child.transform.parent = middleChild.transform;
    }
}
Quaternion currentRotation =
middleChild.transform.localRotation;
Quaternion additionalRotation = Quaternion.Euler(0, 0, 90);
```

```
middleChild.transform.localRotation = currentRotation *
additionalRotation;
foreach (GameObject child in Uplane.GetCollidingObjects())
{
    if (child != middleChild)
    {
        child.transform.parent = gameObject.transform;
    }
}
```

Kôd 3.7 Rotacija gornje plohe kocke

Virtualna kocka prikazana je u pasivnom prikazu aplikacije gdje ju korisnik dodirom može okretati. U slučaju da korisnik pritisne na virtualnu kocku s dva prsta kocka se ponovno postavlja na početnu rotaciju (kôd 3.8).

```
Vector3 speed = new Vector3(0, 0, 0);
float boost = 4;
bool isRotating = false;
[SerializeField] Camera cam;
float prevX;
float prevY;

void Update()
{
    if (Input.touchCount > 0 && Input.touches[0].phase ==
TouchPhase.Began)
    {
        Ray ray =
cam.ScreenPointToRay(Input.touches[0].position);
        RaycastHit hit;

        if (Physics.Raycast(ray, out hit))
        {
            if (hit.transform.tag.Equals("CubeModel"))
            {
                isRotating = true;
                prevX = Input.touches[0].position.x;
                prevY = Input.touches[0].position.y;
            }
            else
            {
                isRotating = false;
            }
        }
    }

    if (isRotating && Input.touchCount == 1)
    {
```

```
        speed = new Vector3(Input.touches[0].position.y -
prevY, prevX - Input.touches[0].position.x, 0);
        prevX = Input.touches[0].position.x;
        prevY = Input.touches[0].position.y;
    }
    else
    {
        speed = new Vector3 (0, 0, 0);
    }

    if (Input.touchCount == 2)
    {
        ResetRotation();
    }

    transform.Rotate(speed * Time.deltaTime * boost,
Space.World);
}

void ResetRotation()
{
    transform.rotation = Quaternion.identity;
}
```

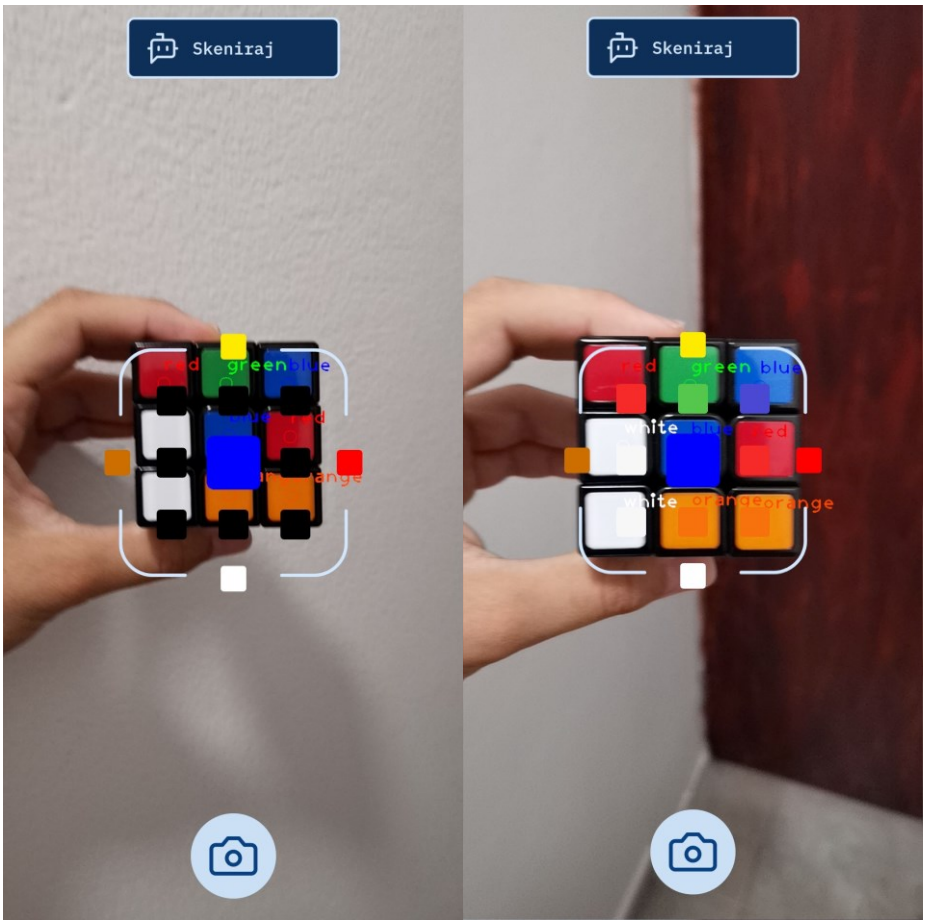
Kôd 3.8 Rotacija kocke dodirom

3.6. Pregled rezultata

Pokretanjem aplikacije na pametnom telefonu pojavljuje se početni ekran prikazan na slici 3.9. Na ekranu ponuđen je gumb fotoaparata koji pokreće skeniranje kocke i gumb na kojemu piše uputstvo korisniku „Skeniraj.“ U prikazu na slici, aplikacija se nalazi u aktivnom stanju.

Slika 3.9 Početni ekran aplikacije ARPrime

Pritiskom na gumb fotoaparata, aplikacija započinje sa skeniranjem kocke. Korisniku su prikazane grafike koje opisuju potrebnu orijentaciju kocke, stranicu koja treba biti skenirana, te ispis rezultata skeniranja stranice (slika 3.10). Grafika koja prikazuje stranicu koja treba biti skenirana može biti pritisnuta. Pritiskom na gumb, mijenja se stranica koja treba biti skenirana, zajedno s grafikama koje su relevantne. Kocka je skenirana u stvarnom vremenu, ali da se osigura najbolja kvaliteta rezultata skeniranja, korisnik mora sam pritisnuti gumb fotoaparata. U slučaju da je jedna od boja pogrešno skenirana, korisnik treba nastavljati stiskati gumb fotoaparata dok nije zadovoljan rezultatom.



Slika 3.10 Prikaz skeniranja stranice kocke

Nakon skeniranja svih stranica, gumb pri vrhu ekrana mijenja stanje u „Riješi.“ Pritiskom na gumb, generira se rješenje kocke u obliku poteza. Potezi su vidljivi u pasivnom načinu prikaza (slika 3.11), zajedno s 3D modelom koji predstavlja trenutno stanje kocke. Model kocke moguće je rotirati dodirom. Poteze je moguće listati, te vraćati se na prijašnji potez u slučaju da je korisnik pogriješio. Kao što je spomenuto ranije, pasivni način prikaza se aktivira kada se mobilni telefon odloži, odnosno postavi paralelno s tlom.

Slika 3. Pasivni način prikaza u aplikaciji

Prateći upute za generirane korake, korisnik će doći do rješenja kocke. U slučaju da učini više grešaka ili da se ne može vratiti na put rješenja, korisnik može ponovno skenirati kocku koja će generirati novo rješenje. Rješenje neće uvijek biti generirano od početka, nego ako ima već riješen prvi ili drugi sloj, samo će se nastaviti tamo gdje je korisnik stao.

3.7. Problemi i poboljšanja

Svaka složena aplikacija koja koristi napredne tehnologije se suočava s nizom izazova i problema koji mogu utjecati na njezinu funkcionalnost, performanse i korisničko iskustvo. Međusobna nekompatibilnost osnovnih komponenata proširene stvarnosti bila je vremenski zahtjevna. Često su se rješenja nalazila unutar skrivenih postavki, čije mijenjanje bi stvaralo druge probleme. Programsko rješenje razvijeno za definirane funkcionalne zahtjeve je prilično delikatno i ograničeno, te ovisi o mnogim uvjetima. Aplikacija „ARPrime“ zavisi o kvalitetnom ulazu slike Rubikove kocke, koja je ograničena

na isključivo jednu stranicu. Problem istovremenog skeniranja triju stranice kocke odjednom tijekom razvoja se pokazao nerješivim zbog nekonzistentnosti osvjetljenja na skeniranim stranicama te puno većeg utjecaja šuma i težine zapisivanja iščitanih boja u polja koja predstavljaju stranice kocke. Problem je bio i u praćenju kocke u prostoru, također zbog promjene osvjetljenja, točnije refleksije svjetla preko naljepnica kocke prema kameri, što bi učinilo da praćeni objekti često nestaju. Ograničavanje zone detekcije je znatno ubrzalo brzinu skeniranja i točnost skeniranih podataka.

Moguća poboljšanja, odnosno dodatci aplikaciji bi prvenstveno bili u unaprjeđenju skeniranja, točnije u rješavanju problema navedenih u prijašnjem odlomku. Uz poboljšanje skeniranja, povećavanje broja dostupnih metoda rješavanja bi znatno povećalo korisnost aplikacije. Uz veći broj dostupnih algoritama bilo bi poželjno nadodati i kratka uputstva za svaki ponuđeni algoritam.

Dodavanje profila, ljestvica bodova, te drugih motivatora poput nagrada za svakodnevno rješavanje unutar aplikacije povećalo bi kvalitetu i zabavu pri učenju slaganja Rubikove kocke, te bi služilo korisnicima kao indikator njihovog napretka.

Zaključak

U ovom radu korištena je tehnologija proširene stvarnosti za razvoj aplikacije za pametne telefone koja pomaže korisniku sastaviti Rubikovu kocku. Uz opis procesa razvoja, te problema i rješenja pri razvoju aplikacije, napravljen je kratak pregled povijesti proširene stvarnosti i opisana je matematička osnova Rubikove kocke. Za razvoj aplikacije bilo je potrebno upoznati se s razvojnim okruženjem Unity, aplikacijom za 3D modeliranje Blender i bibliotekom OpenCV.

Razvoj aplikacije započeo je definiranjem funkcionalnih i nefunkcionalnih zahtjeva čija je implementacija bila ključna za kvalitetan rezultat završnog rada. Pomoću definiranih zahtjeva odabrana je tehnologija OpenCV za računalni vid kao najbolje rješenje za programsku implementaciju. Jednostavno, ali efikasno spremanje stanja kocke, te mogućnost programske manipulacije tim stanjem je bila neophodna za kvalitetnu implementaciju algoritma rješavanja Rubikove kocke. Virtualni prikaz uputa omogućava korisnicima jednostavniju vizualizaciju i bolje razumijevanje poteza. Razvijena aplikacija korisniku daje jednostavne upute koje dovode do rješenja metodom koju i sam može naučiti. Rezultat rada je aplikacija koja uspješno ispunjava odabrane funkcionalne i nefunkcionalne zahtjeve.

Literatura

[1] Encyclopædia Britannica: Erno Rubik https://www.britannica.com/biography/Erno-Rubik, pristupljeno 11.6.2024.

[2] Igor Sunday Pandžić, Tomislav Pejša, Krešimir Matković, Hrvoje Benko, Aleksandra Čereković, Maja Matijašević, "Virtualna okruženja: Interaktivna 3D grafika i njene primjene". Zagreb: Element, 2011.

[3] Microsoft HoloLens 2, https://www.microsoft.com/en-us/d/hololens-2/91pnzzznzwcp?activetab=pivot:overviewtab, pristupljeno 11.6.2024.

[4] Meta Quest Pro, https://www.meta.com/quest/quest-pro/, pristupljeno 13.6.2024.

[5] Ilmari Heikkinen, „Writing augmented reality applications using JSARToolKit", 28. veljače 2012., web.dev, https://web.dev/articles/webgl-jsartoolkit-webrtc, preuzeto 12.6.2024.

[6] Pedro Quelhas Brito, Jasmina Stoyanova, "Marker versus Markerless Augmented Reality. Which Has More Impact on Users?", International Journal of Human–Computer Interaction, 34(9), str. 819–833. https://doi.org/10.1080/10447318.2017.1393974, pristupljeno 12.6.2024.

[7] Ivan E. Sutherland, "A head-mounted three dimensional display", Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I, prosinac 1968., str. 757-764, https://dl.acm.org/doi/10.1145/1476589.1476686, preuzeto 11.6.2024.

[8] Bridget Poetker, A Brief History of Augmented Reality (+ Future Trends & Impact), G2, August 9, 2023, https://www.g2.com/articles/history-of-augmented-reality, pristupljeno 11. 6. 2024.

[9] Georg S. W. Klein and David William Murray. "Parallel Tracking and Mapping for Small AR Workspaces." 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (2007): 225-234. https://www.robots.ox.ac.uk/~gk/publications/KleinMurray2007ISMAR.pdf

[10] Rubik's 3x3 cube, https://www.rubiks.com/products/rubiks-3x3, preuzeto 11.6.2024.

[11] Isaac Chen, "Rubik's Cube algorithms for machines — Part 1 of 2 in a quest to understand the Rubik's Cube", Medium, 2. veljače 2021, https://medium.com/nerd-for-tech/common-rubiks-cube-algorithms-for-machines-part-1-of-2-in-a-quest-to-understand-the-rubik-s-cube-3cb9b53f94b5, preuzeto 11.6.2024.

[12] Darko Žubrinić, Polugrupe i grupe (kratki pregled) (predavanje pripremljeno za kolegij "m-brojevi" profesora Maria Esserta, 28. listopada 2020. g.), Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, Zavod za primijenjenu matematiku https://www.fer.unizg.hr/zpm/djelatnici/red_prof/darko/polugrupe, pristupljeno 11.6.2024.

[13] Tomas Rokicki, Herbert Kociemba, Morley Davidson, John Dethridge, "God's Number is 20", Cube20, https://cube20.org/, pristupljeno 11.6.2024.

[14] Conrad Rider, VisualCube, https://cube.rider.biz/visualcube.php, preuzeto 11.6.2024.

[15] My speed cubing page, Jessica Fridrich, https://www.cs.brandeis.edu/~storer/JimPuzzles/RUBIK/Rubik3x3x3/SOLUTIONS/Rubik3x3x3SolutionFridrich.pdf, pristupljeno 13.6.2024.

[16] Dylan Wang (JPerm), CFOP Speedsolving Method, https://jperm.net/3x3/cfop, pristupljeno 12.6.2024.

[17] Gilles Roux, Roux Method, http://grrroux.free.fr/method/Intro.html, pristupljeno 13.6.2024.

[18] David Singmaster, A step by Step Solution of Rubik's "Magic Cube", 1980., https://web.archive.org/web/20060304183050/http://www.linkedresources.com/teach/rubik/solution.php, pristupljeno 13.6.2024.

[19] Dylan Wang (JPerm), How To Solve The 3x3 Rubik's Cube (Beginner Method), https://jperm.net/3x3, pristupljeno 12.6.2024.

[20] Unity, Unity Technologies, 2024. https://unity.com/, pristupljeno 11.6.2024.

[21] Unity Manual, Intermediate Language To C++ (IL2CPP) Overview, version 2022.3, 15. svibnja 2015., https://docs.unity3d.com/Manual/IL2CPP.html, pristupljeno 12.6.2024.

[22] OpenCV (Open Source Computer Vision Library), https://opencv.org/about/, pristupljeno 12.6.2024.

[23] Alan Jović, Nikolina Frid, "Procesi programskog inženjerstva" (skripta), Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 5. izdanje, Zagreb, rujan 2022. https://www.fer.unizg.hr/_download/repository/Procesi%20programskog%20inzenjerstva_5_izdanje.pdf, pristupljeno 14.6.2024.

[24] Basic Thresholding Operations, OpenCV/Open Source Computer Vision, https://docs.opencv.org/3.4/db/d8e/tutorial_threshold.html, preuzeto 12.6.2024.

[25] Python OpenCV: Canny Edge Detection for Stickerless Rubik's Cube, OpenCV foundation, https://answers.opencv.org/question/230450/python-opencv-canny-edge-detection-for-stickerless-rubiks-cube/, preuzeto 12.6.2024.

[26] Gimbal lock, https://www.researchgate.net/figure/llustrates-the-principle-of-gimbal-lock-The-outer-blue-frame-represents-the-x-axis-the_fig4_338835648, preuzeto iz diplomskog rada Juliana Zeitlhöflera "Nominal and observation-based attitude realization for precise orbit determination of the Jason satellites", Technical University of Munich, lipanj 2019.

Sažetak

Mobilna aplikacija s primjenom proširene stvarnosti za vođeno rješavanje slagalica

Putem ovog završnog rada proučen je koncept proširene stvarnosti (engl. augmented reality, skr. AR), te njezina implementacija kroz povijest i usporedba sličnih, odnosno vezanih tehnologija. Prikazana je matematička osnova Rubikove kocke, kao odabrane slagalice za vođeno rješavanje putem vlastite mobilne aplikacije s primjenom proširene stvarnosti, te su opisani algoritmi za rješavanje kocke. Dan je kratak pregled korištenih tehnologija za razvoj aplikacije i provedena je analiza funkcionalnih i nefunkcionalnih zahtjeva. Prikazana je implementacija postavljenih zahtjeva, te pojednostavljeno su opisane glavne komponente programskog rješenja aplikacije, nazvane „ARPrime“. Završno, dan je osvrt na probleme tokom razvoja aplikacije i moguća unaprjeđenja programskog rješenja u odnosu na postignuti rezultat.

Ključne riječi: proširena stvarnost; AR; proširena stvarnost bez markera; Rubikova kocka; Unity; Android; OpenCV

Summary

Mobile augmented reality application for guided puzzle solving

In this BSc thesis, the concept of augmented reality (AR) is studied, along with its historical overview and its comparison with similar and related technologies. The mathematical foundation of the Rubik's Cube, chosen as the puzzle for guided solving through a custom mobile AR application, is presented, along with the algorithms for solving the cube. A brief overview of the technologies used for application development is given, and an analysis of the functional and non-functional requirements of the application is conducted. The implementation of the specified requirements is presented, and the main components of the application software solution, named "ARPrime," are briefly described. Finally, a reflection on the issues encountered during the development of the application is provided, as well as the potential improvements to the software solution in relation to the achieved result.

Keywords: augmented reality; AR; markerless AR; Rubik's cube; Unity; Android; OpenCV