Prepoznavanje značajki lica u ozbiljnim igrama

Jurić, Maja

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 524

PREPOZNAVANJE ZNAČAJKI LICA U OZBILJNIM IGRAMA

Maja Jurić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 524

PREPOZNAVANJE ZNAČAJKI LICA U OZBILJNIM IGRAMA

Maja Jurić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 524

Pristupnica:	**Maja Jurić (0036522077)**
Studij:	Računarstvo
Profil:	Znanost o podacima
Mentor:	izv. prof. dr. sc. Jurica Babić

Zadatak: **Prepoznavanje značajki lica u ozbiljnim igrama**

Opis zadatka:

Prepoznavanje značajki lica ima ključnu ulogu u preciznom određivanju karakterističnih dijelova lica, što je korisno u aplikacijama poput onih za prepoznavanje lica i medicinsku dijagnostiku. U ozbiljnim igrama, ova tehnologija može unaprijediti korisničko iskustvo prilagodbom zasnovanom na analizi izraza lica. Vaš je zadatak ostvariti rješenje za prepoznavanje značajki lica u ozbiljnim igrama. U okviru navedenog zadatka potrebno je istražiti postojeća rješenja za prepoznavanje značajki lica te na temelju analize predložiti model rješenja koji se sastoji od minimalno tri dijela: (i) modula za prepoznavanje značajki lica na temelju video ulaza iz kamere uređaja; (ii) modul za upravljanje igrom korištenjem prepoznatih značajki lica, te (iii) modul za proširenu stvarnost temeljen na prepoznavanju značajki lica. Posebnu pozornost potrebno je staviti na dizajn rješenja koji će osigurati ispravan rad na računalima bez grafičke kartice s operacijskim sustavom Windows. Rješenje je potrebno implementirati i eksperimentalno evaluirati na prototipu vlastite ili postojeće ozbiljne igre zelene tematike.

Rok za predaju rada: 28. lipnja 2024.

Zahvala

Zahvaljujem mentoru izv. prof. dr. sc. Jurici Babiću i asistentici Ani Radović na trudu i stručnoj pomoći tijekom izrade diplomskog rada.

Veliko hvala roditeljima. Hvala vam što ste uvijek uz mene. Hvala na podršci i svim savjetima, ne samo tijekom pisanja ovog rada, već i kroz cijelo moje obrazovanje. Moja zahvalnost ne može se pretočiti u riječi.

Hvala prijateljima na razumijevanju, pomoći i, prije svega, dobrom društvu. Bez vas studiranje ne bi bilo ni upola toliko ugodno.

Na kraju, posebno zahvaljujem Lovri na strpljenju i motivaciji.

Sadržaj

Uvod

Prepoznavanje značajki lica je područje računalnog vida koje se bavi detekcijom specifičnih točaka na licu. Ova tehnologija omogućuje prepoznavanje identiteta, analizu izraza lica, praćenje pokreta lica te stvaranje proširene stvarnosti na licu korisnika. Ozbiljne igre (igre dizajnirane s ciljem edukacije) mogu posebno profitirati od korištenja novih tehnologija, poput prepoznavanja značajki lica, jer one povećavaju interaktivnost i angažiranost korisnika.

Postoji nedostatak računalnih igara koje koriste prepoznavanje značajki lica u obrazovne svrhe, stoga je cilj ovog rada osmisliti i predstaviti programsko rješenje koje će iskoristiti mogućnosti koje pruža prepoznavanje značajki lica. Programsko rješenje mora biti modularno kako bi se moglo lako uklopiti u postojeće ozbiljne igre. U sklopu razvijenog programskog rješenja nalaze se četiri modula: modul za prepoznavanje značajki lica, modul za detekciju gesti i izraza lica, modul za upravljanje igrom te modul za proširenu stvarnost.

U prvom poglavlju definiran je pojam prepoznavanja značajki lica, navedene su najpopularnije tehnologije za prepoznavanje značajki lica te su istaknuti izazovi s kojima se te tehnologije susreću. U drugom poglavlju istraženo je korištenje novih tehnologija u ozbiljnim igrama te je dan pregled rješenja koje koriste prepoznavanje značajki lica kao novu tehnologiju. Također su istaknute prednosti i nedostatci primjene prepoznavanja značajki lica u tim rješenjima. U trećem poglavlju predstavljen je model rješenja koje koristi prepoznavanje značajki lica te je konceptualno opisana njegova arhitektura. U četvrtom poglavlju detaljno je opisan razvoj svakog modula programskog rješenja. U petom poglavlju evaluirani su razvijeni moduli kroz njihovu integraciju s postojećom ozbiljnom igrom *Eko Zeko*, navedene su prednosti i nedostatci razvijenog rješenja te prijedlozi za budući rad. Na kraju slijede zaključak i popis literature.

1. Prepoznavanje značajki lica

1.1. Definiranje važnih pojmova

Prepoznavanje značajki (engl. *landmark detection)* jedan je od temeljnih problema računalnog vida, a definira se kao proces pronalaženja značajnih dijelova slike [1]. Značajnim dijelovima smatraju se specifične točke ili objekti na slici koji nose važne informacije i mogu se prepoznati u različitim uvjetima bez obzira na npr. promjene u osvjetljenju, skali, kutu snimanja i slično.

Prepoznavanje značajki lica je specifična primjena prethodno definiranog zadatka prepoznavanja značajki. Prepoznavanje značajki lica pokušava detektirati ključne točke na licu osobe. Formalno rečeno, algoritam za prepoznavanje značajki lica kao ulaz prima sliku te kao izlaz daje koordinate prepoznatih točaka na licu [2].

Prema istraživanju [3], točke koje detektiraju algoritmi za detekciju značajki lica mogu se podijeliti u dvije skupine: dominantne i sporedne točke. Dominantne točke su specifične za određene dijelove lica, poput kutova očiju, usana, vrha glave, brade i nosa. Sporedne točke su interpolirane između dominantnih točaka i opisuju konture dijelova lica ili cijelog lica.

1.2. Izazovi u prepoznavanju značajki lica

Postoji nekoliko izazova koji otežavaju prepoznavanje značajki lica, a pojavljuju se zbog prirode podataka [3].

Prvi izazov polazi iz prirode objekta od interesa – lica. Ljudska lica su izuzetno raznolika. Postoje varijacije u oblicima lica i međusobnim odnosima karakterističnih točaka na licu. Dodatne komplikacije nastaju zbog različitih orijentacija glave i izraza lica na slikama.

Drugi izazov čine okolnosti snimanja slike koje često uzrokuju pojavu šuma u podacima. Na primjer, promjene u osvjetljenju mogu stvoriti sjene i izobličenja na licu što otežava precizno prepoznavanje značajki.

Treći izazov predstavlja prekrivanje lica. Djelomično ili potpuno prekrivanje lica drugim objektima ili dijelovima tijela (poput ruku ili kose) te ekstremni položaji glave (naprimjer ako je glava jako nagnuta u nekom smjeru) mogu dovesti do nedostatka ključnih informacija potrebnih za detekciju karakterističnih točaka lica.

Unatoč ovim izazovima, postoje algoritmi i tehnologije računalnog vida koji uspješno izvršavaju zadatak prepoznavanja značajki lica.

1.3. Zadatci temeljeni na prepoznavanju značajki lica

Mnogi drugi zadatci računalnog vida temelje se na prepoznavanju značajki lica, odnosno uzimaju rezultate dobivene prepoznavanjem značajki lica kao predmet daljnje analize kako bi ostvarili neki specifičan zadatak.

1.3.1. Prepoznavanje lica

Prepoznavanje lica (engl. *facial recognition*) koristi detektirane značajke lica za identifikaciju pojedinca na temelju slike ili videozapisa [4].

Nakon što se detektira lice, koristi se prepoznavanje značajki lica kako bi se dobile ključne točke na licu osobe. Zatim se analiziraju dobivene točke, točnije računaju se razne mjere koje proizlaze iz njihovih geometrijskih odnosa (npr. udaljenost između očiju, oblik čeljusti, konture usana, očiju, obraza i dr.). Dobivene vrijednosti spajaju se u jedan vektor koji se naziva predložak lica (engl. *faceprint*). Dalje se koriste algoritmi koji uspoređuju dobiveni predložak lica s licima iz baze podataka na temelju čega se onda može odrediti koja osoba se nalazi na slici. Primjer prepoznavanja lica može se vidjeti na Slika 1.1.

Tehnologija prepoznavanja lica nalazi široku primjenu u različitim sektorima [5]. Prepoznavanje lica počelo je zamjenjivati lozinke čime se otežava neovlašteni pristup korisničkim računima, uslugama i osobnim uređajima. Tehnologiju prepoznavanja lica redovito koristi policija koja tako prikuplja fotografije uhićenika i uspoređuje ih s lokalnim, državnim i saveznim bazama podataka za prepoznavanje lica. Jednom kada se fotografija uhićenika snimi, ona se dodaje u baze podataka koje se pretražuju prilikom svake sljedeće kriminalističke pretrage. U bankarstvu se koristi za autentifikaciju transakcija pogledom u telefon ili računalo, čime se eliminira potreba za jednokratnim lozinkama ili dvofaktorskom autentifikacijom. Na nekim aerodromima implementiran je sustav prepoznavanja lica koji putnicima omogućava brži prolazak kroz automatizirane terminale čime se smanjuje vrijeme čekanja i poboljšava sigurnost.

Sve ove primjene pokazuju kako prepoznavanje lica može poboljšati sigurnost, učinkovitost i korisničko iskustvo u različitim dijelovima života.

Slika 1.1. Rezultat prepoznavanja lica [4]

1.3.2. Prepoznavanje izraza lica

Prepoznavanje izraza lica je proces analize i interpretacije emocionalnih stanja pojedinca [6]. Korištenjem značajki lica, ovaj proces identificira emocije kao što su sreća, tuga, ljutnja, strah, iznenađenje i druge. Analiza izraza lica pruža uvid u trenutačno emocionalno stanje i promjene u raspoloženju tijekom vremena.

Prepoznavanje izraza lica koristi se u sigurnosnim sustavima vozila za praćenje vozačevog stanja budnosti i emocionalnog stanja kako bi se spriječile nesreće uzrokovane umorom ili distrakcijom. Mnoga klinička stanja povezana su s pojavom razlika u izrazima lica, poput poremećaja autističnog spektra i Parkinsonove bolesti pa prepoznavanje izraza lica ima potencijal za njihovo rano dijagnosticiranje.

1.3.3. Proširena stvarnost

Proširena stvarnost (engl. *augmented reality*, AR) je tehnologija koja spaja virtualni i fizički svijet. AR tehnologija koristi kamere, senzore i druge uređaje za praćenje fizičkog okruženja te onda prekriva dijelove stvarnog svijeta ili dodaje na njega čime omogućuje korisnicima da dožive digitalno poboljšanu verziju stvarnosti [7].

Korištenjem detekcije značajki lica, AR objekti mogu se precizno postaviti na određene točke lica te pratiti pokrete i izraze lica u stvarnom vremenu.

Mnogi kanali online prodaje koriste ovu tehnologiju za virtualno isprobavanje proizvoda poput šminke, naočala i odjeće, čime pružaju korisnicima mogućnost da vide kako bi određeni proizvod izgledao na njima prije kupnje. Također, kombinacija proširene stvarnosti

i prepoznavanja značajki lica sve je više prisutna na društvenim mrežama gdje se koriste razni filteri i efekti na fotografijama i videozapisima.

1.4. Radni okviri i biblioteke za prepoznavanje značajki lica

Postoji mnogo alata za prepoznavanje značajki lica, no za ovaj rad su razmotreni i u nastavku opisani samo oni najrašireniji.

1.4.1. Dlib

Dlib je biblioteka otvorenog koda (engl. *open source*) koja se koristi za razne probleme računalnog vida. Sadrži komponente za rad s mrežama, dretvama, linearnom algebrom, strojnim učenjem, rudarenjem podataka, analizom teksta, numeričkom optimizacijom i mnogim drugim zadacima [8]. Osim toga, koristi se i za obradu i analizu slika [9].

1.4.2. OpenCV

OpenCV jedna je od najpopularnijih biblioteka otvorenog koda za računalni vid [10]. Podržava razne programske jezike, uključujući Python, Javu i C++, što je čini izuzetno fleksibilnom i pristupačnom za velik broj korisnika. Dobro je dokumentirana, a zahvaljujući velikom broju korisnika u zajednici je dostupna odlična podrška pa je lakše doći do potrebnih odgovora i resursa.

OpenCV nudi niz algoritama i funkcija za razne zadatke unutar računalnog vida poput filtriranja i segmentacije slika, prepoznavanja objekata, kolorizacije i drugih. Također nudi i mogućnosti prepoznavanja značajki lica [11].

OpenCV omogućuje primjenu u stvarnom vremenu što je čini idealnom za projekte koji zahtijevaju trenutnu obradu i analizu podataka.

1.4.3. Mediapipe

Google je 2018. predstavio Mediapipe, radni okvir otvorenog koda za razvoj rješenja za probleme strojnog učenja [12]. Mediapipe nudi mnoge mogućnosti u području računalnog vida kao što su detekcija objekata, klasifikacija slika, segmentacija slika, detekcija gesti, poza tijela kao i prepoznavanje značajki lica.

Mediapipe radi po principu protočnih struktura (engl. *pipeline*). Podaci prolaze kroz niz povezanih komponenti (tzv. kalkulatora). Svaki kalkulator na neki način obrađuje podatke prije nego ih preda sljedećoj komponenti.

Jedna od glavnih karakteristika Mediapipe-a je njegova modularnost. Postojeće komponente Mediapipe-a mogu se kombinirati s vlastito napisanim komponentama čime se omogućava rješavanje specifičnih problema jer je moguće nadograditi one osnovne mogućnosti Mediapipe-a, koje su već implementirane u radnom okviru.

MediaPipe koristi OpenCV. Korištenjem OpenCV-a, MediaPipe može lako dodavati značajke poput snimanja, obrade i prikaza videozapisa u svoje tokove podataka. MediaPipe također surađuje s TensorFlowom, Googleovim alatom za strojno učenje, kako bi jednostavno mogao dodati prethodno trenirane i prilagođene modele u svoje komponente.

2. Korištenje novih tehnologija u ozbiljnim igrama

2.1. Ozbiljne igre

Sve definicije ozbiljnih igara na koje se može naići imaju za zajednički nazivnik sljedeće:

ozbiljna igra = korisna funkcionalnost + zabavna igra

Za dobru ozbiljnu igru nije važno samo to da ima korisnu funkcionalnost i zabavan aspekt, nego je bitan način njihove kombinacije koja igraču daje maksimalnu korist i ozbiljnog i zabavnog aspekta igre [13]. Ozbiljne igre povećavaju angažman korisnika i olakšavaju učenje kroz interaktivne i dinamične pristupe.

2.2. Prednosti korištenja novih tehnologija u ozbiljnim igrama

Primjena novih tehnologija u ozbiljnim igrama može stvoriti ozbiljne igre koje su učinkovitije u postizanju svojih ciljeva jer iskorištavaju nove načine prenošenja informacija. Također, mogu više poticati na igranje jer je korisnicima korištena nova tehnologija često nepoznata pa time i interesantna.

Naprimjer, proširena stvarnost može obogatiti iskustvo učenja dodavanjem digitalnih tvorevina u stvarni svijet. AR može omogućiti učenicima da vizualiziraju složene koncepte (npr. molekule ili organe) umjesto da samo gledaju njihove slike u udžbeniku.

Umjetna inteligencija (engl. *artificial intelligence*, AI) može se koristiti za personalizaciju iskustva u ozbiljnim igrama. AI može analizirati napredak igrača pa po tome prilagođavati sadržaj igre kako bi odgovarao različitim potrebama i brzinama učenja.

Prepoznavanje značajki lica može se koristiti u igrama za prilagođavanje sadržaja igre na temelju emocionalnog stanja igrača. Na primjer, igra može detektirati ako je igrač frustriran ili zbunjen te mu ponuditi smjernice ili prilagoditi težinu zadataka.

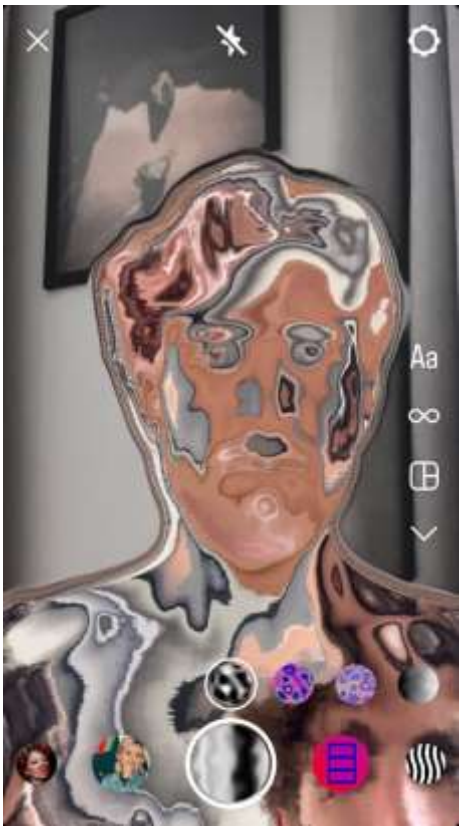
2.3. Pregled postojećih rješenja

U ovom podpoglavlju navode se postojeća rješenja i primjene tehnologije prepoznavanja značajki lica koje su relevantne za primjenu iste u ozbiljnim igrama. Na kraju se provodi

analiza navedenih rješenja kako bi se identificirali načini korištenja tehnologije te njene prednosti i nedostaci što pruža temelj za daljnji razvoj u tom području.

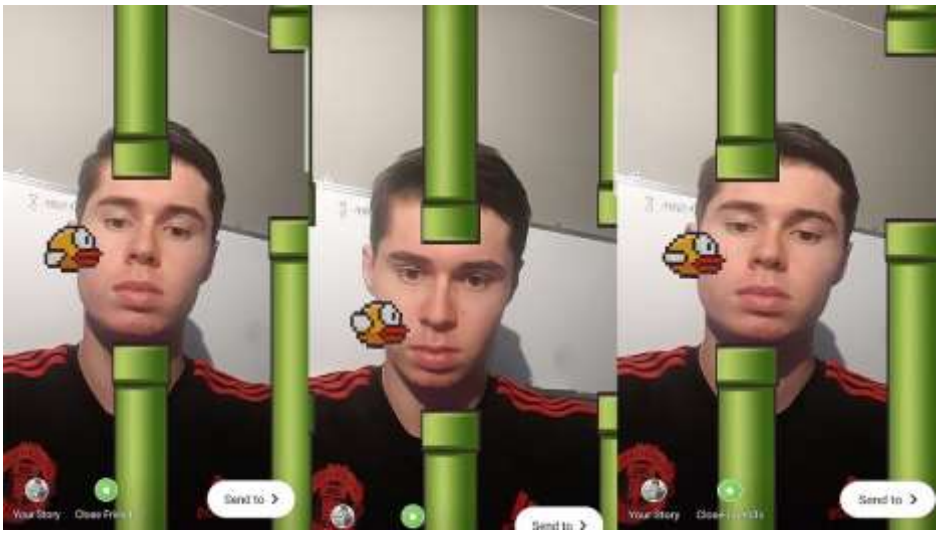
2.3.1. Primjena na društvenim mrežama

Instagram je jedna od vodećih društvenih mreža koja neprestano evoluira kako bi korisnicima pružila nove mogućnosti i iskustva. Jedna od značajnih karakteristika Instagrama su filteri koji omogućuju korisnicima da transformiraju svoje fotografije i videozapise na kreativne načine [14]. Redovito se pojavljuju novi filteri. Filteri, naravno, zahtijevaju prepoznavanje značajki lica za ispravan rad. Često filteri kombiniraju i proširenu stvarnost zajedno s prepoznavanjem značajki lica kako bi na neke dijelove lica postavili različite objekte ili na dio lica primijenili neki efekt. Jedan takav filter može se vidjeti na Slika 2.1.



Slika 2.1. Instagram filter [14]

Na Instagramu su dostupne i različite igre koje koriste prepoznavanje značajki lica za interakciju s korisnicima. Korisnik u takvim igrama može, naprimjer, pomicanjem glave kontrolirati lika ili objekte na ekranu. Primjer jedne takve igre može se vidjeti na Slika 2.2 gdje korisnik pomicanjem glave kontrolira lik ptičice koja izbjegava prepreke.



Slika 2.2. Instagram igra[1]

Trend korištenja takvih filtera i igara pokazuje kako se prepoznavanje značajki lica sve više koristi za poboljšanje korisničkog iskustva na društvenim mrežama.

2.3.2. Project Gameface

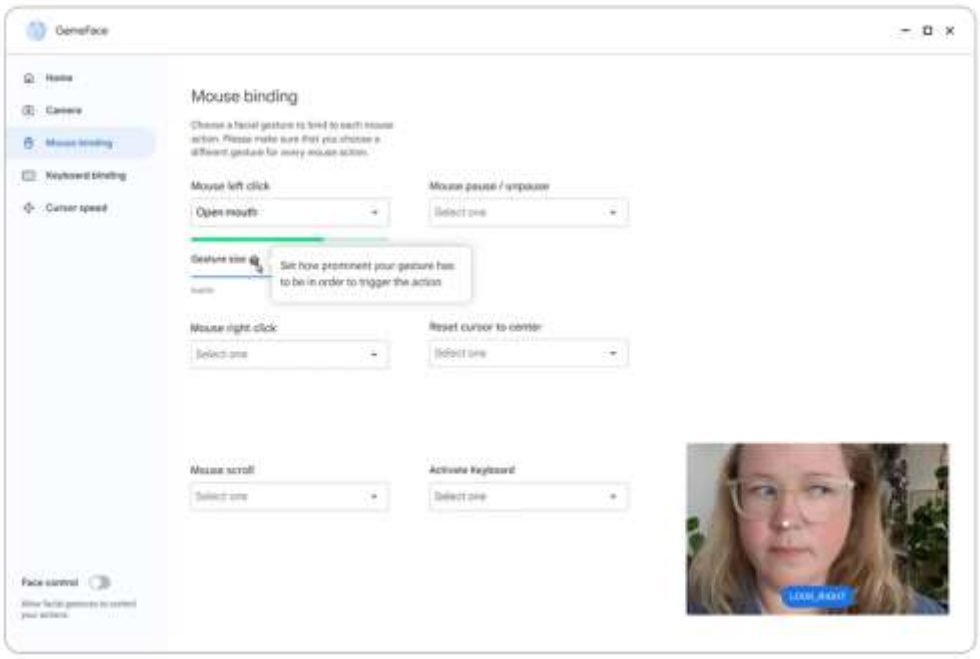
Google je 2023. predstavio *Project Gameface* – softver otvorenog koda za upravljanje kontrolama unutar igre. Korisnici ga mogu kontrolirati bez korištenja ruku, samo pomoću pokreta glave i izraza lica [15].

Project Gameface inspirirao je Lance Carr, kvadriplegični *gaming streamer*, koji uživo putem interneta prenosi svoje igranje videoigara [16]. Lance je prije projekta *Gameface* za igranje igara koristio poseban miš koji prati pokrete glave kako bi kontrolirao pokazivač na ekranu. Međutim, 2021. godine Lanceova kuća je izgorjela u požaru pa je sva njegova gaming oprema izgubljena, uključujući i skupi miš koji prati pokrete glave bez kojeg on više nije mogao igrati igre. Nakon te tragedije, Google je surađivao s Lanceom kako bi osmislili univerzalni miš koji zadovoljava potrebe korisnika poput Lancea.

Project Gameface funkcionira tako da detektira pokrete glave i izraze lica putem web kamere. Dalje se pokreti i izrazi prevode u pomicanje pokazivača i akcije na ekranu. Jedna od posebnih značajki projekta *Gameface* je mogućnost prilagođavanja. Korisnik može prilagoditi koje se geste mapiraju na koje akcije te može prilagoditi intenzitet geste koju

[1] Slika preuzeta s: https://forum.unity.com/threads/face-tracking.1234570/

mora izvesti kako bi se pokrenula odgovarajuća akcija (Slika 2.3). Ova značajka omogućuje korisnicima da prilagode osjetljivost prepoznavanja gesta pa tako sustav odgovara njihovim specifičnim potrebama ili ograničenjima. To osigurava da čak i male geste mogu učinkovito kontrolirati pokazivač ili pokrenuti neku akciju.



Slika 2.3. Prilagođavanje gesti [15]

2.3.3. AR Zombie

U [17] opisana je igra *AR Zombie* namijenjena za korištenje na tabletima. Ova igra koristi detekciju i prepoznavanje lica.

Prije početka igre, korisnik mora u sustav dodati slike ljudi koji će biti zombiji. Nakon pokretanja igre, korisnik se kreće stvarnim okruženjem i snima okolinu koristeći svoj tablet. Putem video kamere na tabletu, igra analizira ljude u okolini korisnika. Sustav koristi prepoznavanje lica (engl. *face recognition*) za identifikaciju pojedinaca. Ako sustav prepozna neku osobu koja je prethodno definirana kao zombi, na njegovo lice superponira lice zombija. Zadaća korisnika je ubiti prepoznate zombije čime se osvajaju bodovi.

AR Zombie je primjer kako tehnologija prepoznavanja lica u igrama može pomoći stvoriti uzbudljivo iskustvo za korisnika.

2.3.4. Prilagođavanje igre

U [19] istražena je uporaba sustava procjene angažiranosti u procesu e-učenja pomoću detekcije značajki lica. Opisani sustav analizira reakcije korisnika kako bi ocijenio je li nastavni materijal na korisnikovoj razini razumijevanja, jesu li fokusirani i slično.

Kao pokazatelj razine angažiranosti učenika koristi se treptanje. Češće i sporije treptanje povezuje se s lutanjem misli i manjkom angažiranosti, dok se manji broj treptaja i kraće trajanje treptaja povezuju s većom razinom angažiranosti.

Za prepoznavanje značajki lica, sustav koristi biblioteku Dlib, s posebnim naglaskom na analizu točaka oko očiju radi praćenja obrasca treptanja. Na temelju procjene razine angažiranosti može se prilagoditi težina nastavnog materijala.

Ovaj rad ilustrira potencijal kombiniranja tehnologije prepoznavanja značajki lica s konceptima edukacijske psihologije. Može se stvoriti rješenje koje se prilagođava korisniku stvarajući time učinkovitije iskustvo e-učenja.

2.3.5. Analiza postojećih rješenja

Analizom postojećih rješenja otkriveno je da se prepoznavanje značajki lica može primijeniti na različite načine.

Može se koristiti u igrama za upravljanje likovima pomoću položaja ili izraza lica. Dodatno se može koristiti i za upravljanje cijelom igrom po uzoru na projekt *Gameface*. Projekt *Gameface* pokazuje kako implementacija mogućnosti kontroliranja igre pomoću značajki lica može igru učiniti više pristupačnom tako da se njome mogu koristiti svi korisnici bez obzira na njihova ograničenja.

Prepoznavanje značajki lica također se može iskoristiti za stvaranje iskustava proširene stvarnosti tako što se razni 3D objekti mogu staviti na detektirane točke lica. Također, tehnologija prepoznavanja značajki lica može se koristiti za prilagođavanje igre korisniku tako što se prati njegova angažiranost.

Analizom postojećih rješenja uočeni su i nedostatci u primjeni tehnologije prepoznavanja značajki lica. Naime, većina rješenja usmjerena je na zabavu, iako postoji veliki potencijal za korištenje ove tehnologije u obrazovne svrhe. Nadalje, većina rješenja koja koristi prepoznavanje značajki lica namijenjena je za korištenje na mobilnim uređajima. Postoji nedostatak takvih rješenja za desktop računala.

Na temelju istraživanja postojećih rješenja, može se osmisliti rješenje koje bi iskoristilo neke od spomenutih mogućnosti tehnologije prepoznavanja značajki lica te koje bi ispravilo neke od nedostataka u primjeni te tehnologije.

3. Model rješenja zasnovanog na prepoznavanju značajki lica

U ovom poglavlju predstavljen je model rješenja koje se oslanja na prepoznavanje značajki lica te su postavljeni zahtjevi koje takvo rješenje mora zadovoljavati. Unutar modela rješenja definirani su moduli od kojih se rješenje sastoji, a koji se temelje na nekim mogućnostima koje nudi prepoznavanje značajki lica. Na kraju, konceptualno se opisuje arhitektura takvog rješenja.

3.1. Zahtjevi

Cilj je osmisliti programsko rješenje koje će na različite načine iskoristiti mogućnosti koje nudi tehnologija prepoznavanja značajki lica kako bi se poboljšalo korisničko iskustvo u igranju ozbiljnih igara.

Većina rješenja koja koristi prepoznavanje značajki lica razvijena su za mobilne uređaje, stoga je ovdje naglasak na razvoju rješenja koje bi se koristilo na računalima.

Rješenje moraju moći koristiti svi korisnici bez obzira na tehničke sposobnosti ili opremu pa se ne postavljaju posebni zahtjevi za tehničkim specifikacijama računala (rješenje mora ispravno raditi i na računalima bez grafičke kartice).

Također, rješenje će biti modularno kako bi se lako integriralo u postojeće igre prema potrebama i zahtjevima korisnika.

3.2. Koncept

Rješenje se sastoji od 4 osnovna modula.

3.2.1. Modul za prepoznavanje značajki lica

Modul za prepoznavanje značajki lica koristi neki od alata za prepoznavanje i praćenje značajki lica korisnika kako bi dobio koordinate značajnih točaka lica. Modul je predviđen za rad na računalima bez potrebe za specijaliziranom hardverskom podrškom pa tako algoritam koji se koristi mora biti optimiziran za rad na procesorima (CPU) kako bi se

osigurao ispravan rad čak i na računalima bez grafičkih kartica. Također, algoritam mora biti predviđen za rad u stvarnom vremenu.

3.2.2. Modul za prepoznavanje orijentacije, gesti i izraza lica

Modul za prepoznavanje orijentacije te gesti i izraza lica obrađuje podatke dobivene iz prethodnog modula kako bi detektirao orijentaciju, geste i izraze lica. Na temelju dobivenih koordinata točaka lica, modul radi izračune kako bi detektirao pozu u kojoj se nalazi lice korisnika, kako je ono orijentirano, radi li korisnik neke geste (poput namigivanja) te kakav izraz lica ima korisnik (npr. otvorena usta ili podignute obrve).

3.2.3. Modul za upravljanje korištenjem prepoznatih značajki lica

Modul za upravljanje obrađuje podatke dobivene iz modula za prepoznavanje orijentacije, gesti i izraza lica. Geste i izrazi lica mapiraju se na specifične akcije unutar igre. Na temelju detektiranih gesti i izraza, pokreću se odgovarajuće akcije unutar igre poput upravljanja likom, interakcije s korisničkim sučeljem ili drugih složenijih interakcija.

3.2.4. Modul za proširenu stvarnost koja koristi prepoznate značajke lica

Modul za proširenu stvarnost koristi prepoznate značajke lica kako bi omogućio prikazivanje AR elemenata koji se dinamički prilagođavaju korisnikovom licu. Modul omogućava postavljanje virtualnih objekata na unaprijed određena mjesta na licu korisnika.

3.3. Arhitektura

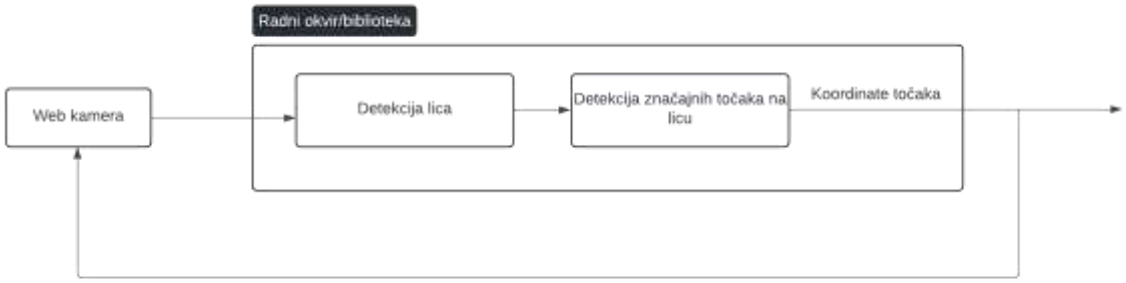
U nastavku poglavlja opisana je i grafički prikazana arhitektura svakog modula[2].

3.3.1. Arhitektura modula za prepoznavanje značajki lica

Arhitektura modula za prepoznavanje značajki lica prikazana je na Slika 3.1. Modul za prepoznavanje značajki lica kao ulazne podatke koristi kontinuirani tok slika iz web kamere računala na koji onda primjenjuje odgovarajuće algoritme iz neke od dostupnih biblioteka

[2] Skice arhitekture izrađene su uz pomoć alata *Lucid*: https://lucid.co/

ili radnih okvira kako bi detektirao značajke lica. Izlazni rezultat modula za prepoznavanje značajki lica su koordinate tih točaka. Ove koordinate mogu se dalje koristiti za razne zadaće. Modul neprekidno uzima ulazne slike iz web kamere i detektira točke lica.



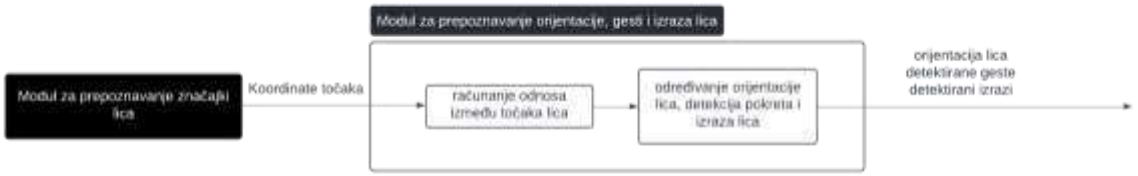
Slika 3.1. Arhitektura modula za prepoznavanje značajki lica

3.3.2. Arhitektura modula za prepoznavanje orijentacije, gesti i izraza

Arhitektura modula za prepoznavanje orijentacije, gesti i izraza lica prikazana je na Slika 3.2.

Modul za prepoznavanje orijentacije, gesti i izraza lica oslanja se na modul za prepoznavanje značajki lica kako bi dobio koordinate točaka lica. Na temelju dobivenih koordinata, modul radi izračune kako bi detektirao orijentaciju lica te geste i izraze lica. Ovo može uključivati računanje udaljenosti i kutova između ključnih točaka kako bi se odredili različiti izrazi lica poput otvorenih/zatvorenih usta, podignutih obrva i sl. Također se mogu pratiti geste lica poput pomicanja glave lijevo-desno, gore-dolje, treptanja i slično.

Ovaj modul kao izlaz daje informaciju o detektiranoj orijentaciji lica kao i o tome koje su geste i izrazi lica detektirani.



Slika 3.2. Arhitektura modula za prepoznavanje orijentacije, gesti i izraza lica

3.3.3. Arhitektura modula za upravljanje igrom korištenjem prepoznatih značajki lica

Arhitektura modula za upravljanje igrom korištenjem prepoznatih značajki lica prikazana je na Slika 3.3.

Modul za upravljanje igrom korištenjem prepoznatih značajki lica oslanja se na modul za prepoznavanje orijentacije, gesti i izraza lica kako bi dobio informaciju o detektiranim gestama i izrazima lica. Modul aktivira akcije koje su prije pokretanja mapirane na pojedine izraze i geste lica. Na primjer, podizanje obrva igrača može pokrenuti određenu radnju u igri poput pucanja, dok mrštenje može uzrokovati zaustavljanje ili promjenu smjera kretanja lika.

Slika 3.3. Arhitektura modula za upravljanje igrom

3.3.4. Arhitektura modula za proširenu stvarnost koja koristi prepoznavanje značajki lica

Arhitektura modula za proširenu stvarnost koja koristi prepoznavanje značajki lica prikazana je na Slika 3.4.

Modul za proširenu stvarnost temeljen na prepoznavanju značajki lica oslanja se na modul za prepoznavanje značajki lica, kako bi dobio koordinate točaka lica, te na modul za prepoznavanje orijentacije, gesti i izraza lica kako bi dobio orijentaciju lica.

Na temelju dobivenih koordinata, modul postavlja AR objekte na odgovarajuće pozicije na licu korisnika. To, naravno, podrazumijeva da se prije pokretanja mora odrediti na koje karakteristične točke lica će se postaviti AR objekti. Na temelju detektirane orijentacije, modul rotira AR objekte u skladu s licem.

Nakon postavljanja AR objekata, modul može izvršiti dodatne prilagodbe kako bi se objekti bolje uklopili s licem korisnika. Ovo može uključivati pomak, skaliranje i rotiranje AR objekta. Postavljeni AR objekt dalje nastavlja pratiti položaj i orijentaciju lica.

Slika 3.4. Arhitektura modula za proširenu stvarnost

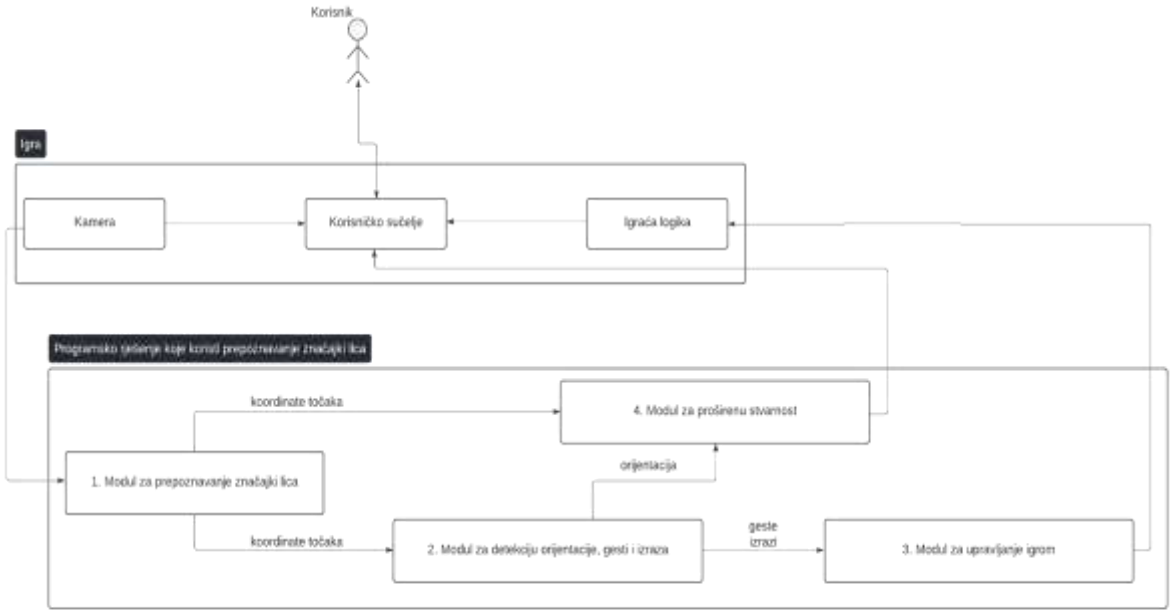
3.3.5. Arhitektura cjelokupnog programskog rješenja

Na Slika 3.5 prikazana je cjelokupna arhitektura programskog rješenja.

Web kamera kontinuirano snima korisnika i tako modulu za prepoznavanje značajki lica dostavlja ulazne podatke. Modul za prepoznavanje značajki lica nakon obrade tih podataka prosljeđuje detektirane točke lica u modul za detekciju orijentacije, izraza i gesti te modul

za proširenu stvarnost. Modul za detekciju orijentacije, izraza i gesti šalje modulu za upravljanje igrom informacije o detektiranim gestama i izrazima lica. Modul za upravljanje igrom je u interakciji s logikom igre i pokreće odgovarajuće akcije u igri. Modul za proširenu stvarnost na detektirane točke lica postavlja AR objekte koje prikazuje na korisničkom sučelju.

Nakon razvoja navedenih modula, također treba razviti dodatne komponente koje bi omogućavale uspješnu integraciju tih modula s postojećim programskim rješenjima.



Slika 3.5. Arhitektura cjelokupnog programskog rješenja

4. Razvoj rješenja

U ovom poglavlju detaljno je opisan razvoj rješenja za prepoznavanje značajki lica u igrama. Za izvedbu rješenja odabrana je razvojna platforma *Unity*. *Unity* je popularan alat za razvoj igara s dobrom podrškom za rad s proširenom stvarnosti, što je ključno za ovaj projekt. Dodatno, *Unity* ima veliku zajednicu korisnika i opsežnu dokumentaciju, što olakšava rješavanje problema.

4.1. Tehnologije i alati u Unityju

Prije kretanja u implementaciju projekta, potrebno je istražiti trenutno dostupne tehnologije, alate i dodatke (engl. *plugin*) u Unityju koji bi se mogli upotrijebiti za razvoj sustava koji koristi prepoznavanje značajki lica. Ovaj korak je ključan za odabir odgovarajuće tehnologije (ili kombinacije tehnologija) koja bi mogla ostvariti sve zahtjeve navedene u poglavlju *3.1. Zahtjevi*.

4.1.1. ARFoundation

ARKit i ARCore su pluginovi koje su razvili Apple i Google, a namijenjeni su za razvoj proširene stvarnosti na njihovim platformama, odnosno ARKit za iOS uređaje te ARCore za Android uređaje. ARFoundation je Unityjev radni okvir koji integrira ova dva plugina kako bi omogućio razvoj AR aplikacija koje se mogu implementirati na oba operacijska sustava [19].

ARFoundation je besplatan za korištenje i dolazi s obilnom dokumentacijom te primjerima koda i projekata što olakšava početak razvoja vlastite AR aplikacije. Ovaj radni okvir je široko prihvaćen u zajednici pa postoji dobra podrška što omogućuje pronalazak odgovora na pitanja i rješavanje problema tijekom razvoja.

AR Foundation nudi mnogo gotovih funkcionalnosti koje su korisne u razvoju AR aplikacija:

1. Detekcija ravnih ploha (engl. *plane detection*) omogućuje detektiranje i iscrtavanje ravnih ploha, poput podova ili stolova, u okolini korisnika. Na Slika 4.1 se vidi kako AR Foundation iscrtava detektirane ravne plohe.

Slika 4.1. Detekcija ravnih ploha [19]

2. Praćenje slike (engl. *image tracking*) koristi se za praćenje unaprijed definiranih slika u okolini korisnika što dalje omogućuje interakciju s virtualnim sadržajem na temelju prepoznatih slika, na primjer postavljanje 3D modela na detektiranu sliku. Na Slika 4.2 se na prepoznate slike brojeva postavljaju 3D modeli tih brojeva.

Slika 4.2. Postavljanje 3D objekta na detektiranu sliku[3]

3. Praćenje određenih točaka u prostoru (engl. a*nchor tracking*) omogućuje aplikaciji da prati proizvoljne točke u stvarnom svijetu što je korisno za pozicioniranje virtualnih objekata u prostoru, kao što je crvena kockica

[3] Slika preuzeta s: https://github.com/Unity-Technologies/arfoundation-demos

postavljena na poziciju na Slika 4.1. Objekt (na slici crvena kockica) ne pomiče se s pomicanjem uređaja, nego ostaje stacionarna u točki u kojoj je postavljena.

AR Foundation nudi i mogućnost prepoznavanja značajki lica, ali mnoge funkcionalnosti ograničene su na određenu platformu. Na primjer, ARKit nudi funkcionalnost praćenja izraza lica kao što su namigivanje ili mrštenje, ali ta funkcionalnost ne postoji u AR Core-u. S druge strane, ARCore omogućuje detekciju specifičnih dijelova lica kao što su usta ili obrve, ali ta opcija nije dostupna u ARKitu.

Važno je napomenuti da se rješenja implementirana u ARFoundationu ne mogu u potpunosti testirati unutar Unityjeve razvojne okoline (engl. *Unity Editor*). Za testiranje funkcionalnosti AR-a unutar razvojnog okruženja potrebno je koristiti AR Foundation Remote[4] aplikaciju koja se može preuzeti s *Unity Asset Store*-a. AR Foundation Remote omogućuje programerima da simuliraju AR okruženje na mobilnim uređajima izravno iz Unityja, bez potrebe za stvaranjem zasebne izvršne datoteke (engl. *build*) za svaku promjenu, što značajno ubrzava testiranje rješenje jer nije potrebno svaki put generirati izvršnu datoteku da bi se programsko rješenje ispitalo.

AR Foundation Remote nije besplatan alat, nego se radi o jednokratnoj kupnji koja omogućuje korištenje ovog alata tijekom razvojnog procesa

Također, kako AR Foundation spaja ARKit i ARCore, očito je usmjeren na razvoj za mobilne uređaje pa ne podržava razvoj za desktop.

4.1.2. MARS

MARS (engl. *Mixed and Augmented Reality Studio*) je Unityjeva vlastita ekstenzija koja je specijalizirana za razvoj AR aplikacija [20]. MARS developerima pruža visoku razinu apstrakcije pa je intuitivan za kreiranje vlastitih AR iskustava. Jedna od ključnih mogućnosti MARS-a je podrška za detekciju značajki lica koja omogućava precizno praćenje i interakciju s elementima lica korisnika [21].

[4] AR Foundation Remote: https://assetstore.unity.com/packages/tools/utilities/ar-foundation-remote-2-0-201106

Tijekom rada s MARS-om dostupan je njegov vlastiti simulator koji se unutar Unityjeve razvojne okoline može koristiti za testiranje razvijenog rješenja. To značajno ubrzava proces razvoja jer nema potrebe za stalnim generiranjem izvršnih datoteka (engl. *building*).

Nema sumnje da je MARS moćan i široko korišten alat za razvoj AR iskustava, no može se koristiti isključivo uz plaćanje licence na godišnjoj razini.

4.1.3. Python i Unity

Python je poznat po svojim mnogim mogućnostima u području računalnog vida (i, specifično, u prepoznavanju značajki lica) tako da je ideja kombiniranja Pythona s Unityjem prirodna.

U nastavku su navedena dva paketa koja to pokušavaju ostvariti.

4.1.3.1 Python for Unity 2.0.1

Paket Python for Unity omogućava Python skriptama pristup svim funkcionalnostima unutar Unityeve razvojne okoline kao i pokretanje Python skripti iz C#-a [22]. Iako se isprva ovaj paket čini kao odlična ideja, njegova primjena nije zaživjela pa tako nema dobru potporu i nije u širokoj primjeni. Velika mana ovog paketa je što je isključivo dizajniran za korištenje unutar Unityeve razvojne okoline pa ne podržava generiranje izvršnih datoteka. To znači da je veza između C# i Pythona ograničena samo na razvoj unutar Unityeve razvojne okoline bez daljnje mogućnosti distribucije kao zasebnog izvršnog programa.

4.1.3.2 UnityPython

Paket UnityPython omogućava integraciju Pythona kao skriptnog jezika unutar Unityja. UnityPython omogućuje pisanje i izvršavanje Python skripti izravno unutar Unityja. Ovaj paket koristi IronPython, implementaciju Pythona za .NET, koji omogućuje interakciju Python i C#-a, kojeg koristi Unity. U trenutku pisanja ovog rada, UnityPython više nije održavan. Odluka o prekidu ažuriranja i podrške prvenstveno je posljedica prestanka podrške za *Python 2*, koji je dostigao kraj svog životnog ciklusa 1. siječnja 2020. [23].

4.1.4. OpenCV

Kao što je spomenuto u poglavlju *1.4.2. OpenCV*, OpenCV jedna je od najpopularnijih biblioteka otvorenog koda za računalni vid, zato je prirodno tražiti načine kako ju koristiti u Unity okruženju. Postoji nekoliko dodataka za integraciju OpenCV-a u Unityju.

4.1.4.1 OpenCV+Unity

OpenCV+Unity je besplatan plugin koji omogućava korištenje OpenCV biblioteke unutar Unity okruženja [24]. Iako je odličan za testiranje i razvoj unutar Unityeve razvojne okoline, ovaj dodatak ne podržava generiranje izvršnih datoteka, što znači da se aplikacije koje ga koriste ne mogu distribuirati kao samostalni izvršni programi. Unatoč tom ograničenju, OpenCV+Unity pruža mogućnosti za eksperimentiranje s računalnim vidom tijekom faze razvoja.

4.1.4.2 OpenCVForUnity

OpenCVForUnity je plaćeni plugin koji omogućava korištenje OpenCV biblioteke unutar Unity okruženja [25]. Nudi mogućnost testiranja rješenja direktno u Unityevoj razvojnoj okolini prije generiranja izvršne datoteke. Plugin dolazi s brojnim primjerima koji mogu pomoći korisnicima da brzo savladaju i integriraju OpenCV funkcionalnosti u svoje projekte. Postoji i besplatna trial verzija u kojoj se mogu isprobavati funkcionalnosti koje dolaze s OpenCVForUnity pluginom, ali bez mogućnosti generiranje izvršne datoteke.

4.1.5. Banuba

Banuba je tvrtka za razvoj softverskih rješenja za detekciju lica i korištenje proširene stvarnosti na licu [26]. Banuba nudi softver za praćenje lica putem web ili mobilne kamere.

Softver se može koristiti za stvaranje AR filtera. Korisnici mogu na svoje lice dodati vizualne efekte ili virtualne objekte. Softver također omogućuje animiranje avatara ili maski koje kopiraju izraze lica korisnika pa tako, ako korisnici ne žele pokazati svoje pravo lice u videochatu/igri, mogu se pojaviti kao digitalni avatari.

Međutim, alat Banuba nije besplatan.

4.1.6. Zappar

Zappar je platforma koja želi poslovnim subjektima omogućiti jednostavno stvaranje AR iskustava što bi im moglo pomoći u reklamiranju njihovih proizvoda.

ZapWorks je Zapparov proizvod koji omogućava da se jednostavno razvijaju AR iskustva za web [27]. ZapWorks može raditi i u Unity okruženju, ali samo za web ili mobilne uređaje. ZapWorks nudi opcije za rad s prepoznavanjem značajki lica, ali su one uglavnom usredotočene za razvoj AR iskustava na temelju prepoznatog lica.

Zapparovi proizvodi plaćaju se na mjesečnoj bazi.

4.1.7. MediapipeUnity plugin

MediapipeUnity je besplatan plugin koji integrira Googleov Mediapipe radni okvir u Unity [28].

MediapipeUnity plugin omogućuje razvoj rješenja za Windows, Android, iOS i druge platforme. Iako nije najbolje dokumentiran, sve više ljudi se njime koristi, što znači da se putem zajednice mogu pronaći odgovori na razna pitanja.

MediapipeUnity plugin omogućuje korištenje svih komponenti Mediapipe-a te dodavanje vlastitih komponenti u protočnu strukturu (engl. *pipeline*). Pruža mogućnosti kao što su detekcija lica, stvaranje mreže lica (engl. *face mesh*), detekcija zjenica, detekcija ruku, detekcija poze, detekcija objekata i mnoge druge. Uz pomoć detektiranih točaka koje je moguće dobiti, mogu se graditi i AR iskustva.

MediapipeUnity plugin nudi gotove primjere koji olakšavaju početak implementacije željenih funkcionalnosti.

4.1.8. Usporedba

Pregled alata za prepoznavanje značajki lica u Unityju pruža uvid u raznolikost opcija, svaka sa svojim specifičnim karakteristikama i ograničenjima. Tablica 1 prikazuje usporedbu spomenutih alata s obzirom na faktore koji su bitni pri odabiru.

Tablica 1 Usporedba alata prepoznavanje značajki lica u Unityju

ime alata	**cijena**	**mogućnost buildanja za desktop**	**dodatne bilješke**	**ispunjava zahtjeve navedene u *3.1. Zahtjevi***
ARFoundation	besplatno (+opcionalni jednokratni trošak aplikacije AR Foundation Remote)	DA	mogućnost simulacije bez buildanja uz dodatan trošak aplikacije AR Foundation Remote	NE
MARS	oko 560€ godišnje	DA	ne postoji dodatan trošak za simuliranje rješenja, dostupan simulator unutar MARS ekstenzije	DA, ako ne postoji druga, besplatna, alternativa
PythonForUnity	besplatan	NE	nema dobru potporu	NE
UnityPython	besplatan	DA	nije više održavan	NE
OpenCV+Unity	besplatan	NE	detektira manji broj točaka lica u usporedbi s	NE

			nekim drugim rješenjima	
OpenCVForUnity	95€	DA (samo plaćena verzija)	detektira manji broj točaka lica u usporedbi s nekim drugim rješenjima, postoji besplatna opcija, ali ona ima ograničene mogućnosti	DA, ako ne postoji druga, besplatna, alternativa
Banuba	cijena na upit (besplatan probni period)	DA	dobra opcija za razvoj AR iskustava, nije široko korišten, postoji manjak podrške	NE, zbog manjka dostupnih informacija
Zappar	cijena na upit (besplatan probni period)	DA	opcija za razvoj AR iskustava	NE, zbog manjka dostupnih informacija
Mediapipe Unity plugin	besplatno	DA	daje veliki broj točaka, ima gotove komponente za rad s prepoznavanjem značajki lica	DA

ARFoundation, iako besplatan, fokusiran je isključivo na razvoj za mobilne uređaje, što ga čini neprikladnim izborom za projekt koji zahtijeva podršku za desktop računala. MARS, Banuba i Zappar su alati koji se plaćaju, ali se može iskoristiti besplatan trial period kako bi se bolje razumjelo kako su neke funkcionalnosti u tim alatima izvedene. Također, u tim je alatima veći naglasak na stvaranje AR iskustava, a ne na implementaciju drugih mogućnosti koje su potrebne za razvoj projekta opisanog u poglavlju *3. Model rješenja* , pa, bez plaćanja za isprobavanje tih alata, ne zna se imaju li sve funkcionalnosti potrebne za izvedbu projekta.

MediapipeUnity plugin se nameće kao najbolji izbor. Besplatan je, podržava razvoj za desktop, pruža širok spektar funkcionalnosti te nudi gotove komponente za prepoznavanje značajki lica što pomaže u počinjanju projekta koji koristi prepoznavanje značajki lica. Nadalje, mogu se dodavati vlastite komponente kako bi se prilagodile specifičnim zahtjevima i zadacima projekta.

Naravno, izbor MediapipeUnity plugina prikladan je specifično za ovaj projekt s njegovim specifičnim zahtjevima. Za druge projekte, ovisno o njihovim potrebama, resursima, ciljanim platformama i drugim zahtjevima, možda je potrebno istražiti i koristiti druge alate.

4.2. Implementacija osnovnih funkcionalnosti

Za razvoj programskog rješenja korištena je platforma za razvoj igara Unity. Unity je odabran zbog svoje popularnosti, opsežne dokumentacije i snažne podrške za rad s proširenom stvarnosti. Programsko rješenje pisano je u jeziku *C#* za što je korištena razvojna okolina *Microsoft Visual Studio 2017*.

4.2.1. Priprema Unity okruženja

Za prepoznavanje značajki lica koristi se MediapipeUnity plugin. Ovaj plugin dostupan je za preuzimanje na GitHub-u[5], popularnoj platformi za pohranu i dijeljenje koda. Potrebno je preuzeti cijeli direktorij kao ZIP datoteku te ju onda raspakirati negdje na računalu.

Za uvoz plugina u Unity projekt, potrebno je u Unityevoj razvojnoj okolini odabrati *Assets* → *Import Package* → *Custom Package* te onda u okviru koji se otvori pronaći i odabrati

[5] Mediapipe Unity plugin: https://github.com/homuler/MediaPipeUnityPlugin

raspakirani *MediapipeUnity* plugin. U okviru koji se potom otvori u Unityevoj razvojnoj okolini, potrebno je označiti sve opcije i kliknuti na *Import*.

4.2.2. Prepoznavanje značajki lica

Mediapipe nudi gotove klase za rad s prepoznavanjem značajki lica što znači da je samo potrebno razumjeti kako te klase komuniciraju i kako iz njih dohvatiti koordinate točaka lica za daljnju uporabu.

4.2.2.1 Klasa GraphRunner

Glavna uloga klase *GraphRunner* je inicijalizacija i obrada podataka. Klasa *GraphRunner* omogućuje inicijalizaciju grafa Mediapipe-a, komponente koja prima i obrađuje ulazne podatke i stavlja ih u tok za obradu. Klasa *GraphRunner* mora se dalje specijalizirati za specifične potrebe.

GraphRunner je apstraktna klas što znači da sve klase koje nasljeđuju GraphRunner moraju implementirati metode *StartRun* i *AddTextureFrameToInputStream,* definirane u klasi *GraphRunner*, kako bi ih specijalizirale. U metodi *StartRun* definira se što se radi pri inicijalizaciji klase. U metodi *AddTextureFrameToInputStream* definira se što se radi s ulaznim podatkom prije nego ga se proslijedi dalje u tok.

4.2.2.2 Klasa *FaceMeshGraph*

Klasa *FaceMeshGraph* je specijalizacija klase *GraphRunner* za detekciju značajki lica. Ova klasa omogućuje postavljanje parametara za detekciju lica, pokretanje i zaustavljanje grafa te obradu rezultata.

Klasa *FaceMeshGraph* nasljeđuje *GraphRunner* klasu. Dodatno, ova klasa nudi mogućnost definiranja maksimalnog broja lica koja će se pratiti te minimalne vrijednosti povjerenja za detekciju i praćenje lica. Klasa *FaceMeshGraph* definira nekoliko rukovatelja događajima (engl. *event handlers*). Rukovatelji događajima općenito omogućuju dodavanje i uklanjanje metoda koje će se automatski pozivati kada se određeni događaj dogodi. U klasi *FaceMeshGraph* definirani su sljedeći rukovatelji događajima:

1. *OnFaceDetectionsOutput*
2. *OnMultiFaceLandmarksOutput*
3. *OnFaceRectsFromLandmarksOutput*

4. *OnFaceRectsFromDetectionsOutput.*

Rukovatelji događajima omogućuju dodavanje i uklanjanje metoda koje će se pozivati kada su rezultati detekcije dostupni. Konkretno, kod dobivanja značajki lica, koristi se *OnMultiFaceLandmarksOutput* rukovatelj događajima na koji se onda dodaju metode koje će biti pozvane kada se detektiraju točke lica. Sve te metode dobit će kao argument detektirane točke lica koje će moći dalje koristiti.

4.2.2.3 Klasa *ImageSourceSolution*

Klasa *ImageSourceSolution* odgovorna je za upravljanje izvorom podataka, konkretno koristi se za upravljanje grafom za obradu slika. Ova klasa omogućuje postavljanje, pokretanje, pauziranje i zaustavljanje grafova koji se koristi za obradu slika.

Klasa *ImageSourceSolution* je apstraktna klasa parametrizirana s bilo kojom klasom koja nasljeđuje *GraphRunner* klasu. *ImageSourceSolution* klasa kontinuirano dohvaća okvire s web kamere te predaje te podatke (slike) klasi *GraphRunner* za daljnju obradu tih podataka.

Klasa *ImageSourceSolution* ima polje u koje se može predati panel na kojem se može prikazivati slika korisnika što kasnije može biti korisno za stvaranje proširene stvarnosti koja superponira virtualne objekte na sliku korisnika.

4.2.2.4 Dohvaćanje koordinata točaka lica

Metode se mogu pretplatiti na rukovatelj događajima *OnMultiFaceLandmarksOutput* što znači da će biti pozvane svaki put kada se okine događaj, odnosno kada se prepoznaju točke lica. Dakle, klasa *FaceMeshGraph* svaki put kada detektira značajke lica pokreće sve metode koje su dodane na rukovatelj događajima *OnMultiFaceLandmarksOutput*. Događaj *OnMultiFaceLandmarksOutput* prosljeđuje listu podataka tipa *NormalizedLandmarkList* u sve metode koje su pretplaćene na njega, po jedan *NormalizedLandmarkList* za svako detektirano lice.

Klasa *NormalizedLandmarkList* predstavlja strukturu podataka u koju su pohranjene normalizirane koordinate točaka lica. Izraz „normalizirane“ odnosi se na to da su u rasponu [0, 1]. Da bi se te vrijednosti koristile u Unityju, potrebno ih je skalirati prema visini i širini prikaza lica.

Za dohvaćanje koordinata točaka lica i njihovu daljnju obradu, treba se definirati vlastita klasa koja će naslijediti *ImageSourceSolution* klasu parametriziranu s *FaceMeshGraph*-om kako bi klasa bila specijalizirana za obradu podataka dobivenih od *FaceMeshGraph* klase.

Potrebno je implementirati metodu *OnStartRun*, koja služi za inicijalizaciju. U njoj se treba pretplatiti na rukovatelj događajima *OnMultiFaceLandmarksOutput* kako bi se mogle dobiti koordinate točaka lica.

4.2.3. Detektiranje gesti i izraza lica

Nakon što se dobiju koordinate točaka lica, promatrajući njihove geometrijske odnose, mogu se detektirati izrazi i geste lica. Svaka detektirana točka nalazi se u 3D prostoru, pri čemu x-os označava smjer lijevo-desno, y-os označava smjer gore-dolje, a z-os smjer naprijed-natrag. Točke u *NormalizedLandmarkList* uvijek su u istom poretku što osigurava da se ista točka uvijek može dohvatiti pomoću istog indeksa. Mediapipe nudi grafički prikaz[6] koji jasno označava koja je točka na kojem indeksu.

Nenamjerna gesta ili promjena pozicije lica ne bi smjela aktivirati neku akciju. Zato je implementirano da se položaj ili gesta registrira samo ako se drži određeni vremenski period, koji se može specificirati u inspektoru u Unityjevom razvojnom okruženju.

U nastavku je opisano kako su određeni nagibi lica u x, y i z smjeru što je kasnije korisno za postavljanje i rotiranje AR objekata tako da oni prate orijentaciju lica. Također, opisano je kako su detektirane neke geste i izrazi lica. Detektirane geste i izrazi lica kasnije se mogu koristiti za upravljanje igrom tako da se, nakon detektiranja geste ili izraza lica, u igri pokrene odgovarajuća akcija. Cilj je implementirati mogućnost detekcije što većeg broja gesti i izraza kako bi se kasnije omogućila fleksibilnost u mapiranju detektiranih gesti i izraza na različite kontrole unutar igre.

Logika za detekciju orijentacije lica te gesti i izraza lica nalazi se u klasi *FaceEvents*.

4.2.3.1 Detektiranje nagiba lica s obzirom na x-os (engl. *pitch*)

Detektiranje nagiba lica s obzirom na x-os (engl. *pitch*) odnosi se na usmjerenost lica osobe prema gore ili prema dolje.

[6] Mediapipe indeksi točaka lica

Za određivanje kuta koriste se koordinate brade i vrha glave. Specifični indeksi točaka koje se moraju dohvatiti mogu se iščitati iz grafičkog prikaza koji nudi Mediapipe koji prikazuje koji je indeks svake detektirane točke lica. Ovdje je kao točka brade uzeta točka na indeksu 152, a kao točka vrha glave točka na indeksu 10.

Kut se izračunava kao arkus tangens razlike između njihovih z koordinata te razlike između njihovih y koordinata. Kut se pretvori iz radijana u stupnjeve množenjem s 180 i dijeljenjem s π. Pseudokod za računanje kuta s obzirom na x-os dan je u Kod 4.1.

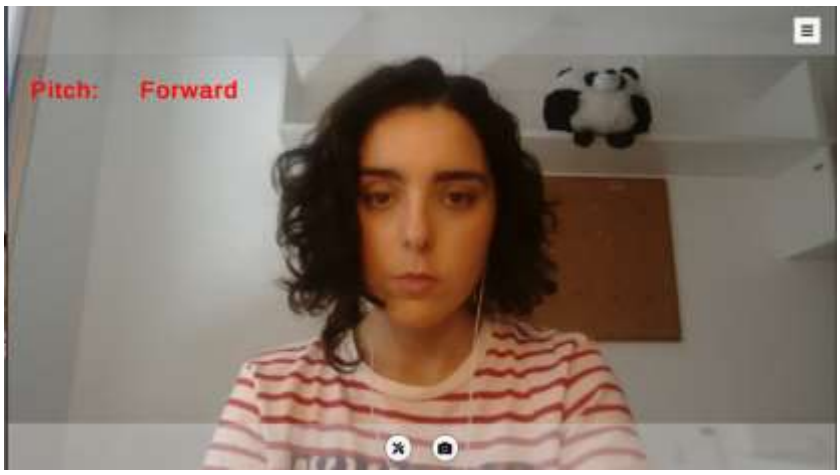
```
deltaZ = brada.Z – vrh_glave.Z
deltaY = brada.Y – vrh_glave.Y
kut = arkustangens(deltaZ / deltaY)* 180 / π
```

Kod 4.1. Računanje kuta s obzirom na x-os

Poza lica s obzirom na x-os određuje se na temelju kuta. Ako je kut veći od unaprijed postavljenog praga, smatra se da lice gleda prema gore (Slika 4.3), ako je kut manji od negativnog unaprijed postavljenog praga, smatra se da lice gleda prema dolje (Slika 4.4). U slučaju da se kut nalazi unutar intervala [-prag, prag], smatra se da lice gleda prema naprijed (Slika 4.5).

Slika 4.3. Lice gleda prema gore

Slika 4.4. Lice gleda prema dolje



Slika 4.5. Lice gleda prema naprijed

4.2.3.2 Detektiranje nagiba lica s obzirom na y-os (engl. *yaw*)

Detektiranje nagiba lica s obzirom na y-os (engl. *yaw*) odnosi se na usmjerenost lica osobe prema lijevo ili prema desno.

Za određivanje kuta koriste se koordinate lijevog i desnog kraja lica. Specifični indeksi točaka koje se moraju dohvatiti mogu se iščitati iz grafičkog prikaza koji nudi Mediapipe koji prikazuje koji je indeks svake detektirane točke lica. Ovdje je kao lijevi kraj lica uzeta točka na indeksu 352, a kao desni kraj lica točka na indeksu 123.

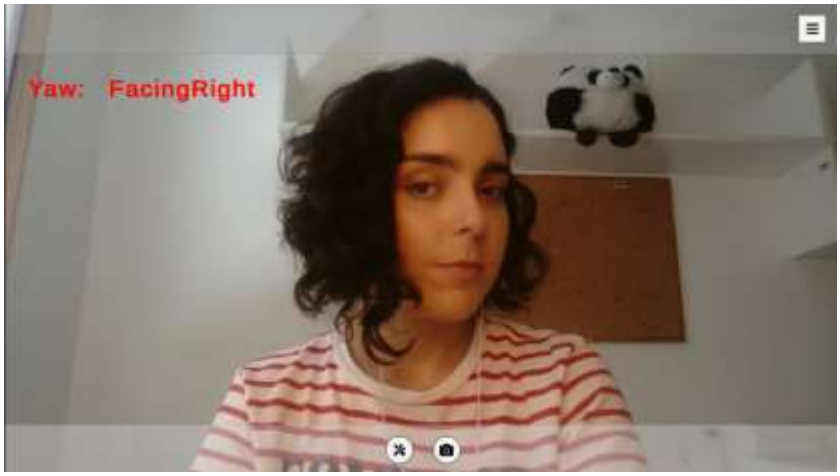
Kut se izračunava kao arkus tangens razlike između njihovih z koordinata te razlike između njihovih x koordinata. Kut se pretvori iz radijana u stupnjeve množenjem s 180 i dijeljenjem s π. Pseudokod za računanje kuta s obzirom na y-os dan je u Kod 4.2.

```
deltaZ = desni_obraz.Z – lijevi_obraz.Z
deltaX = desni_obraz.X – lijevi_obraz.X
kut = arkustangens(deltaZ / deltaX)* 180 / π
```

Kod 4.2. Računanje kuta s obzirom na y-os

Poza lica s obzirom na y-os određuje se na temelju kuta. Ako je kut veći od unaprijed postavljenog praga, smatra se da lice gleda prema lijevo (Slika 4.6), ako je kut manji od negativnog unaprijed postavljenog praga, smatra se da lice gleda prema desno (Slika 4.7). U slučaju da se kut nalazi unutar intervala [-prag, prag], smatra se da lice gleda prema naprijed (Slika 4.8).

Slika 4.6. Lice usmjereno ulijevo



Slika 4.7. Lice usmjereno udesno

Slika 4.8. Lice usmjereno naprijed

4.2.3.3 Detektiranje nagiba lica s obzirom na z-os (engl. *roll*)

Detektiranje nagiba lica s obzirom na z-os (engl. *roll*) odnosi se na nagib lica osobe ulijevo ili udesno.

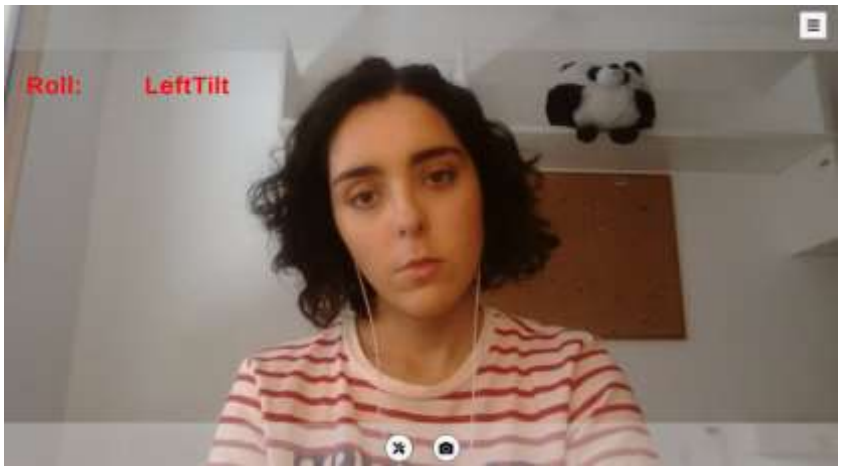
Za određivanje kuta koriste se koordinate brade i vrha glave. Specifični indeksi točaka koje se moraju dohvatiti mogu se iščitati iz grafičkog prikaza koji nudi Mediapipe koji prikazuje koji je indeks svake detektirane točke lica. Ovdje je kao točka brade uzeta točka na indeksu 152, a kao točka vrha glave točka na indeksu 10.

Kut se izračunava kao arkus tangens razlike između njihovih y koordinata te razlike između njihovih x koordinata. Kut se pretvori iz radijana u stupnjeve množenjem s 180 i dijeljenjem s π. Pseudokod za računanje kuta s obzirom na z-os dan je u Kod 4.3.

```
deltaY = vrh_glave.Y - brada.Y
deltaX = vrh_glave.X - brada.X
kut = arkustangens(deltaY / deltaX)* 180 / π
```

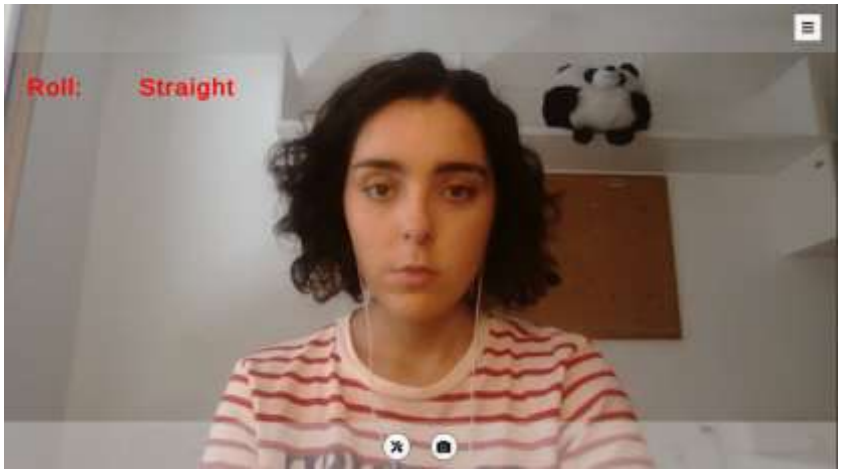
Kod 4.3. Računanje kuta s obzirom na z-os

Poza lica određuje se na temelju kuta. Ako je kut veći od unaprijed postavljenog praga, smatra se da je lice nagnuto ulijevo (Slika 4.9), ako je kut manji od negativnog unaprijed postavljenog praga, smatra se da je lice nagnuto udesno (Slika 4.10). U slučaju da se kut nalazi unutar intervala [-prag, prag], smatra se da lice nije nagnuto (Slika 4.11).



Slika 4.9. Lice nagnuto ulijevo

Slika 4.10. Lice nagnuto udesno



Slika 4.11. Lice nije nagnuto

4.2.3.4 Detektiranje namigivanja i treptanja

Za detekciju namigivanja i treptanja treba se računati udaljenost između točaka na gornjoj i donjoj granici lijevog i desnog oka. Kao mjera udaljenosti koristi se euklidska udaljenost. Euklidska udaljenost daje najkraći razmak između dvije točke u prostoru, a računa se kao korijen od zbroja kvadrata razlika njihovih koordinata po osima [29]. Korištenje euklidske udaljenosti, umjesto korištenja npr. razlike između samo y koordinata, osigurava da se dobije razmak između gornje i donje točke bez obzira na orijentaciju lica. Matematička formula za euklidsku udaljenost dviju točaka, p i q, dana je izrazom (1).

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_1)^2 + (p_3 - q_3)^2} \quad (1)$$

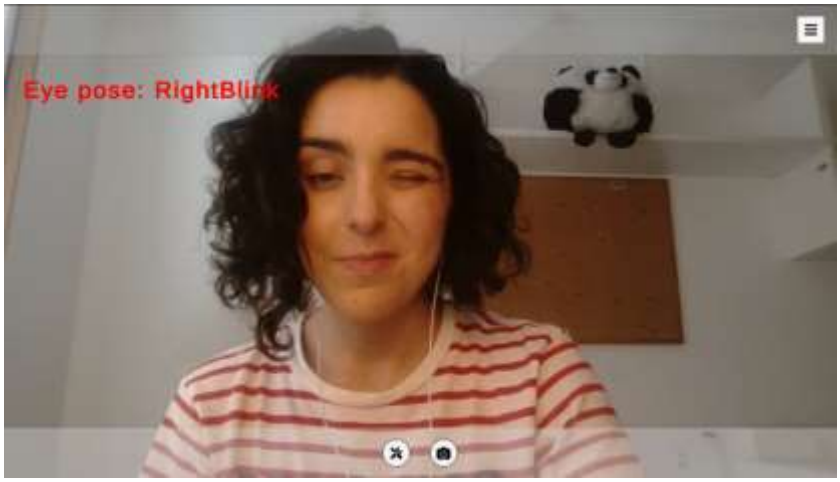
Za računanje udaljenosti potrebno je dohvatiti gornju i donju točku lijevog i desnog oka. Specifični indeksi točaka koje se moraju dohvatiti mogu se iščitati iz grafičkog prikaza koji nudi Mediapipe koji prikazuje koji je indeks svake detektirane točke lica. Ovdje je kao gornja točka lijevog oka uzeta točka na indeksu 386, a donja točka lijevog oka na indeksu 374. Kao gornja točka desnog oka uzeta je točka na indeksu 159, a kao donja točka desnog oka uzeta je točka na indeksu 145.

Pseudokod za detekciju namigivanja i treptanja dan je u Kod 4.4. Prvo se računa euklidska udaljenost za lijevo i za desno oko. Ako je ta udaljenost manja od unaprijed definiranog praga za namigivanje, onda je detektiran desni (Slika 4.12) ili lijevi namig (Slika 4.13). Ako je udaljenost točaka i za lijevo i za desno oko manja od unaprijed definiranog praga, onda je detektiran treptaj (Slika 4.14).

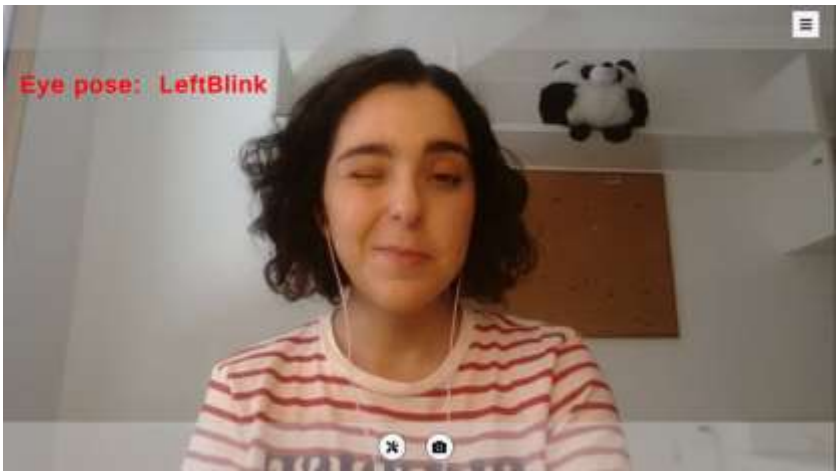
```
lijevo_oko = Euklidska_udaljenost(lijevo_oko_gore,
lijevo_oko_dolje)
desno_oko = Euklidska_udaljenost(desno_oko_gore, desno_oko_dolje)

ako (lijevo_oko < prag){
      ako (desno_oko < prag){
            detektiran treptaj
      } inače{
            detektiran lijevi namig
      }
}inače ako (desko_oko < prah){
      detektiran desni namig
}
```

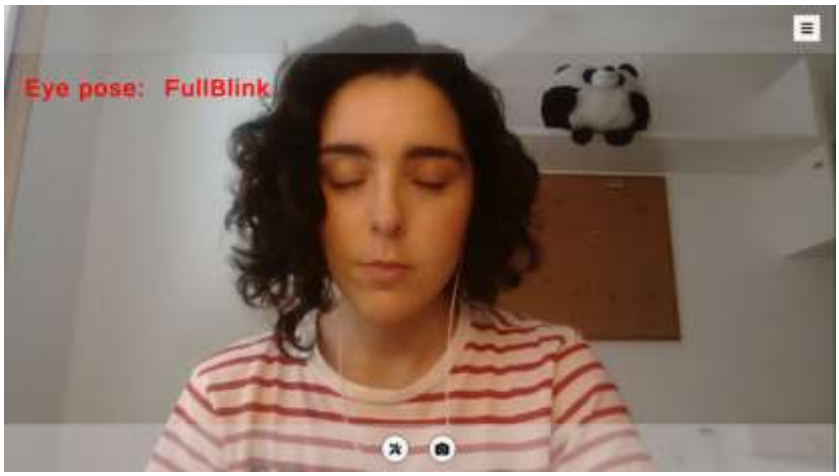
Kod 4.4. Detekcija namiga i treptaja



Slika 4.12. Desni namig



Slika 4.13. Lijevi namig



Slika 4.14. Treptaj

4.2.3.5 Detektiranje otvaranja usta

Za detekciju otvaranja usta potrebno je dohvatiti gornju i donju točku na otvoru usta, odnosno donju točku gornja usne i gornju točku donje usne, kako bi se izračunala njihova udaljenost. Kao mjera udaljenosti, kao i kod detekcije namigivanja i treptanja, koristi se euklidska udaljenost. Specifični indeksi točaka koje se moraju dohvatiti mogu se iščitati iz grafičkog prikaza koji nudi Mediapipe koji prikazuje koji je indeks svake detektirane točke lica. Ovdje je kao gornja točka uzeta točka na indeksu 13, a kao donja točka točka na indeksu 14.

Pseudokod za detekciju otvorenih/zatvorenih usta dan je u Kod 4.5. Prvo se računa euklidska udaljenost gornje i donje točke na otvoru usta. Ako je udaljenost veća od unaprijed definiranog praga, onda su detektirana otvorena usta (Slika 4.15), inače, usta smatraju zatvorenim (Slika 4.16).

```
usta = Euklidska_udaljenost(gornja_tocka_usta, donja_tocka_usta)

ako (usta > prag){
      detektirana otvorena usta
}inace{
      usta zatvorena
}
```

Kod 4.5. Detekcija otvorenih usta

Slika 4.15. Otvorena usta

Slika 4.16. Zatvorena usta

4.2.3.6 Aktiviranje događaja

Događaji (engl. *events*) omogućuju objektima da šalju obavijesti drugima kada se ispune određeni uvjeti, kada se dogodi neki događaj ili promjena stanja [30]. Drugi objekti mogu reagirati na te obavijesti tako da se pretplate na njih.

Događaji razdvajaju dijelove koda koji detektiraju promjene stanja od dijelova koda koji na njih reagiraju čime se povećava održivost koda. Također, omogućavaju komponentama da komuniciraju bez da su one čvrsto povezane.

Klasa *FaceEvents* koristi različite događaje kako bi signalizirala promjene stanja lica, poput treptanja očiju (*OnLeftBlink*, *OnRightBlink*¸ *OnBlink*), otvaranja usta (*OnMouthOpened*) ili promjena položaja glave (*OnHeadPoseChanged*). Da bi se događaji aktivirali, koristi se operator *?.Invoke*, koji prvo provjerava postoji li pretplatnik na događaj, a zatim poziva sve pretplaćene metode.

4.2.4. Upravljanje igrom

Cilj je omogućiti upravljanje igrom pomoću detektiranih gesti i izraza lica. Klasa *FaceEvents* ima zadaću detektirati geste i izraze lica koji se onda dalje mogu koristiti za upravljanje igrom.

U nastavku je opisano kako je izvedena komunikacija između klase *FaceEvents*, koja obavještava o detektiranim gestama i izrazima lica, i logike igre.

4.2.4.1 Pretplaćivanje na događaje

Klasa *FaceEvents* detektira geste, izraze i orijentaciju lica te aktivira odgovarajuće događaje, poput *OnMouthOpened* kada korisnik otvori usta ili *OnLeftBlink* kada korisnik namigne lijevim okom. Za pokretanje metoda koje reagiraju na detektirane pojave definiraju se nove skripte u kojima se metode s odgovarajućom logikom pretplaćuju na događaje definirane u klasi *FaceEvents*. Metode koje se žele pretplatiti na događaj, tako da se pozovu kada se on aktivira, mogu to napraviti koristeći operator '+='.

Naprimjer, za upravljanje igračem potrebno je stvoriti novu skriptu u kojoj će se geste lica mapirati na akcije igrača kao što su otvaranje usta za skakanje, ili okretanje glave lijevo ili desno za pokretanje lika u tim smjerovima.

4.2.4.2 Rukovanje operacijama na glavnoj dretvi

Za interakciju s osnovnim elementima igre (objektima (engl. *game objects*), komponentama i korisničkim sučeljem) Unity koristi glavnu dretvu (engl. *main thread*). Dakle, na glavnoj dretvi se izvršavaju sve operacije povezane s iscrtavanje objekata, ažuriranjem stanja objekata te interakcijom s korisničkim sučeljem [31]. To uključuje manipulaciju objektima, tj. stvaranje, uništavanje, aktiviranje i deaktiviranje objekata. Također, transformacije, odnosno promjene pozicije, rotacije i skale, moraju biti obavljene na glavnoj dretvi. Osim toga, bilo kakve promjene vezane uz korisničko sučelje, poput promjene teksta, veličine ili boje elemenata, moraju biti izvršene na glavnoj dretvi. Pokušaj izvođenja tih operacija izvan glavne dretve izazvat će grešku.

Glavni razlog zašto se ove operacije moraju izvoditi na glavnoj dretvi i zašto Unity javlja grešku ako se one izvode izvan glavne dretve je taj što bi istovremeni pristup iz više dretvi mogao dovesti do nekonzistentnog stanja i nepredvidivih rezultata što bi kasnije dovelo do problema s performansama i padova sustava. Dakle, da bi se osigurala stabilnost aplikacije u Unityju, potrebno je na glavnoj dretvi izvoditi operacije koje manipuliraju osnovnim elementima igre.

Međutim, Mediapipe detekciju značajki lica provodi na sporednoj dretvi što znači da se metode koje će upravljati igrom ne smiju direktno pretplatiti na događaje koje detektira *FaceEvents* jer će se one na taj način automatski izvršavati na istoj dretvi iz koje su i pozvane. Umjesto toga, metode koje se pretplaćuju na događaje iz klase *FaceEvents* trebaju

prvo prebaciti izvođenje logike na glavnu dretvu prije nego što pokrenu izvršavanje potrebnih koraka.

Za tu svrhu napisana je klasa *UnityMainThread*. Ova klasa koristi red (engl. *queue*) za pohranu akcija koje treba izvršiti na glavnoj dretvi. Klasa *UnityMainThread* omogućava da se iz bilo koje dretve dodaju metode u red metodi koje će se izvršiti na glavnoj dretvi.

U svojoj *Update* metodi, koja se poziva jednom za svaki kadar (engl. *frame*) na glavnoj dretvi, klasa uzima sve metode iz reda i izvršava ih. Kako se metoda *Update* izvršava na glavnoj dretvi, sve metode koja ona pozove također se izvršavaju na glavnoj dretvi. Time se omogućava da operacije koje uključuju manipulaciju Unity objektima ili korisničkim sučeljem, koje bi inače izazvale greške ako se izvode izvan glavne dretve, budu sigurno izvršene.

4.2.5. Proširena stvarnost

Na detektirano lice, na specifične točke lica, koje se dobiju od modula za prepoznavanje značajki lica, postavljaju se 2D ili 3D objekti koji onda prate i prilagođavaju se promjenama u položaju i orijentaciji lica.

4.2.5.1 Pohrana podataka o AR objektima

Za potrebe pohrane informacija o AR objektu koji se želi stvoriti u sceni, koristi se specifična struktura podataka u kojoj se mogu definirati sva potrebna svojstva.

Scriptable Object je posebna vrsta klase koja omogućava stvaranje vlastitih podatkovnih struktura [32]. *Scriptable Object* pohranjuje podatke kao resurse (engl. *assets*) u projektu.

Vlastiti *Scriptable Object* kreira se tako da se definira klasa koja nasljeđuje *ScriptableObject* te se koristi *CreateAssetMenu* atribut kako bi se primjerci mogli stvarati u Unityevoj razvojnoj okolini. Primjerci se stvaraju tako da se u Unityevoj razvojnoj okolini odabere *Assets → Create → Custom → vlastiti scriptable objekt.*

Za potrebe stvaranja AR objekta na temelju prepoznatih značajki lica definiran je *Scriptable Object* naziva *ARObjectData* koji pohranjuje podatke o tome koji se objekt želi instancirati, na kojoj točki lica, treba li neki pomak (engl. *offset*) od te pozicije, koliko se skaliranje treba primijeniti na objekt, treba li objekt biti aktivan u sceni te u kojim smjerovima se objekt smije rotirati.

4.2.5.2 Postavljanje AR objekta

Za stvaranje iskustva proširene stvarnosti stvorena je nova klasa – *FaceAR*. Klasi *FaceAR* predaje se lista objekata tipa *ARObjectData* koji sadrže informacije o AR objektu. Te informacije uključuju primjerak objekta koji se treba instancirati u sceni, koju poziciju lica objekt treba pratiti, koliki pomak od te točke treba primijeniti, koliko se skaliranje treba primijeniti na objekt, treba li objekt biti aktivan u sceni te u kojim smjerovima se objekt smije rotirati.

U *Start* metodi, koja se poziva pri pokretanju scene, skripta instancira objekte u sceni, ali ih postavlja na neaktivne. To olakšava da se kasnije, kada se detektiraju točke lica, potrebni objekti samo aktiviraju te da im se mijenja položaj i orijentacija, te da se, ako se lice više ne vidi na slici, objekti samo deaktiviraju.

U *Start* metodi klasa *FaceAR* se pretplati na događaje klase *FaceEvents* koja ju onda obavještava o promjeni položaja i orijentacije lica. Klasa *FaceEvents* kontinuirano šalje informacije o promjeni položaja lica što omogućuje kontinuirano ažuriranje pozicije instanciranih objekata s obzirom na trenutnu poziciju njihove referentne točke. Mijenjanje pozicije AR objekta je jednostavno jer klasa *FaceAR* dobije normalizirane 3D koordinate referentne točke. Kako su koordinate normalizirane, vrijednosti x i y koordinata referentne točke moraju se skalirati sa širinom i visinom prikaza kako bi se dobile koordinate točke na koju se treba postaviti AR objekt.

Klasa *FaceEvents* također obavještava o promjeni orijentacije lica što omogućuje da se AR objekti orijentiraju u skladu s licem. Rotiranje AR objekata s obzirom na neku os radi se samo ako je ono dopušteno u pripadajućem *ARObjectData* objektu.

Kako je već spomenuto u poglavlju *4.2.4.2 Rukovanje operacijama na glavnoj dretvi*, u Unityju postoje operacije koje se ne smiju izvršavati izvan glavne dretve. To uključuje i manipulaciju objektima (stvaranje, aktiviranje i deaktiviranje objekata) te transformacije objekata (promjene pozicije, rotacije ili skale). Znači, metode koje se pretplaćuju na događaje klase *FaceEvents*, a koje su zadužene za promjenu položaja i orijentacije AR objekata u skladu s licem moraju operacije potrebne za izvršavanje svoje zadaće prebaciti na glavnu dretvu kako bi se one mogle izvesti jer u suprotnom Unity daje grešku. Za to se ponovno može iskoristiti klasa *UnityMainThread*. Samo je potrebno, nakon što je detektiran događaj, u red definiran u klasi *UnityMainThread* dodati metodu koja izvršava odgovarajuću logiku kao reakciju na taj događaj.

4.2.5.3 Skaliranje objekta s obzirom na udaljenost od kamere

Veličina objekta treba se mijenjati ovisno o približavanju i udaljavanju od ekrana. Kako se lice približava kameri, objekt bi se trebao povećavati, dok bi se udaljavanjem lica od kamere, objekt trebao smanjivati.

Generalno informacija o položaju objekta u smjeru naprijed-natrag leži u z koordinati, ali, problem kod korištenja Mediapipe-a je da trenutno z koordinata ne nosi stvarnu informaciju o dubini, tj. o udaljenosti od kamere. Međutim, vrijednost z koordinate svejedno nosi neku informaciju jer se može uočiti da se njezina vrijednost povećava kada se lice približava kameri te se smanjuje kada se lice udaljava od kamere.

Zato je bilo potrebno odrediti vlastiti način mjerenja udaljenosti od kamere. Izračunata udaljenost od kamere mogla bi se koristiti kao faktor skaliranja veličine objekta kod približavanja i udaljavanja od ekrana.

Faktor skaliranja je određen eksperimentalno. Za računanje faktora skaliranja koristi se z koordinata točke nosa pomnožena s omjerom visine i širine ekrana te s eksperimentalno određenom konstantom (vrijednost 100). Korištenje vrijednosti z koordinate u računanju faktora skaliranja osigurava da je faktor skaliranja veći kada je lice bliže kameri, a manji kada je lice dalje od kamere.

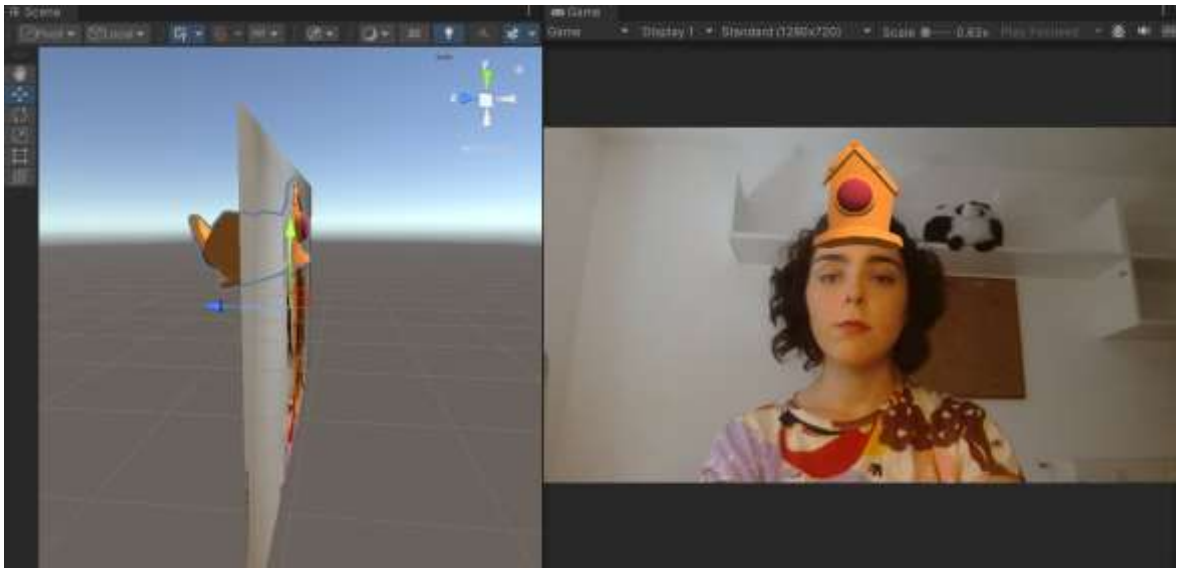
4.2.5.4 Podešavanje kamera

Kod stvaranja nove scene u Unityju, u sceni se automatski nalazi jedna kamera. Na Slika 4.17 je prikazan jedan AR objekt, ovdje kruna, na licu korisnika, točnije, na vrhu glave korisnika.

Slika 4.17. AR kruna na licu korisnika

Iako proširena stvarnost na prvi pogled izgleda u skladu s očekivanjima, postoji problem kod rada s nekim objektima.

Zbog ekstremnih položaja glave, neki objekti nestaju iza platna (engl. *canvas*) na kojem se prikazuje lice korisnika. Na Slika 4.18 prikazan je jedan takav slučaj. Desno, na prikazu igre (engl. *game view*) vidi se kako dijelovi AR objekta, krune, nestaju kada se glava podigne prema gore. U slučaju još veće rotacije glave, objekt u potpunosti nestaje. Lijevo, na prikazu scene (engl. *scene view*) može se uočiti da kruna nestaje iza platna što je razlog da se ona ne prikazuje u cjelosti u igri.



Slika 4.18. AR objekt nestaje iza platna

Kako bi se riješio taj problem, potrebno je u scenu dodati još jednu kameru. Ideja iza toga je da se dvije kamere podese tako da svaka prikazuje različitu skupinu objekata tako da objekti iz različitih skupina onda međusobno nisu u interakciji. Dakle, u sceni trebaju postojati dvije kamere – kamera koja iscrtava sve osim AR objekata (dalje, glavna kamera) te kamera koja iscrtava samo AR objekte (dalje, AR kamera).

Kako bi se ostvario ispravan način rada dviju kamera, važno ih je dobro podesiti. Za to su bitna tri svojstva kamera: *Culling Mask*, *Depth* i *Clear Flags* [33].

Svojstvo *Culling Mask* određuje koje slojeve scene kamera može vidjeti čime se može kontrolirati što kamera prikazuje, a što je od nje sakriveno. Svi AR objekti dodijeljeni su posebnom sloju. U svojstvu *Culling Mask* glavne kamere isključi se sloj kojem pripadaju

AR objekti tako da glavna kamera prikazuje sve osim AR objekata. AR kameri se isključe svi slojevi osim sloja u kojem se nalaze AR objekti tako da ona ne prikazuje ništa osim AR objekata. Dakle, sada glavna kamera ne prikazuje AR objekte, a AR kamera prikazuje samo njih.

Dalje, treba se prilagoditi redoslijed prikazivanja objekata kako bi se AR objekti prikazivali povrh platna na kojem se prikazuje lice korisnika. To se može ostvariti podešavanjem vrijednosti svojstva dubine (engl. *depth*). Dubina određuje redoslijed iscrtavanja kamera u sceni. Kamere s većim vrijednostima dubine iscrtavaju se nakon kamera s manjim vrijednostima što znači da će objekti koje prikazuje ona kamera s većim vrijednostima uvijek biti prikazani povrh objekata koje prikazuje kamera s manjim vrijednostima. Kako bi se osiguralo da se AR objekti iscrtavaju povrh platna na kojem se prikazuje lice korisnika, bitno je AR kameri postaviti vrijednost dubine na veću vrijednost od glavne kamere. Same vrijednosti nisu bitne, nego je samo bitno da AR kamera ima veću vrijednost dubine od glavne kamere. Naprimjer, dubina glavne kamere može se postaviti na 0, a dubina AR kamere na 1, što osigurava da će AR kamera uvijek iscrtavati svoje objekte iznad objekata koje iscrtava glavna kamera.

Konačno, potrebno je prilagoditi kako se prikazuje prazan prostor oko objekata koje prikazuje AR kamera kako bi se osiguralo da se vidi ono što je iza objekata koje ona prikazuje, odnosno da se vidi platno na kojem je prikazano lice korisnika. Svojstvo *Clear Flags* definira kako se prikazuje prazan prostor oko objekata koje kamera prikazuje. Svojstvo *Clear Flags* AR kamere mora biti postavljeno na *Depth Only* jer se time omogućava da se vidi sadržaj koji je prikazan iza nje.

Na slikama Slika 4.19 i Slika 4.20 vidi se kako u sceni s dvije kamere, nakon njihovog podešavanja, objekti ne nestaju iza platna.

Slika 4.19. AR s dvije kamere

Slika 4.20. AR s dvije kamere, nagnuto lice

Međutim, sada postoji problem da se vidi i stražnja strana objekta, odnosno ne daje se iluzija da se kruna omata oko glave.

4.2.5.5 Sakrivanje stražnje strane objekta

Problem koji se pojavljuje je taj da se objekt prikazuje u cjelosti, a mi želimo da se sakrije njegov stražnji dio kako bi se dala iluzija omatanja objekta oko glave. Ovaj problem pojavljuje se samo kod nekih AR objekata, tj. onih za koje želimo da se omataju oko glave. Rješenje za problem dano je u nastavku.

Prvo je potrebno originalni objekt (npr. krunu) zamijeniti s praznim objektom koji ima dvoje djece. Prvo dijete je originalni AR objekt koji se želi postaviti u scenu (npr. kruna), a drugi objekt je geometrijski lik koji ima zadaću aproksimirati lice. Taj dodatni geometrijski lik ne mora savršeno pratiti oblik lica, već samo služi da sakrije stražnji dio AR objekta kada se

mijenja orijentacija glave. Na taj geometrijski lik potrebno je dodati materijal koji bi skrivao stražnji dio AR objekta, ali propuštao prikaz drugih elemenata iza njega (osobite slike lica korisnika).

Za izradu takvog materijala, potrebno je definirati novo sjenilo – *OcclusionShader*. Sjenila (engl. *shader*) se koriste u računalnoj grafici kako bi se definiralo kako će se objekti ili scene prikazivati na ekranu [34]. Glavna svrha sjenila je kontrola iscrtavanja svakog piksela čime se određuje kako svaki piksel reagira na svjetlo, boje, teksture, sjene i druge vizualne efekte. Sjenila omogućuju različite vrste efekata kao što su simulacija materijala (poput metala, stakla, ili tkanine), simulacija svjetla i sjene, animacije i drugo.

U *OcclusionShader* sjenilu je deklariran blok *Pass*. U *Pass* bloku, u kojem se definiraju svojstva iscrtavanja za sjenilo, definira se *Blend Mode* koji određuje način kombiniranja piksela različitih objekata kako bi se proizveo konačan piksel koji se prikazuje na ekranu [35]. *Blend Mode* se definira uz pomoć izvorišnog (engl. *source*) i odredišnog (engl. *destination*) čimbenika. Izvorišni čimbenik je piksel materijala koji se crta, a odredišni čimbenik je piksel koji već postoji u zaslonom spremniku (engl. *frame buffer*) na toj lokaciji. Zasloni spremnik je privremeni spremnik koji se koristi za pohranu podataka o pikselima koji se iscrtavaju na ekranu prije nego što se oni prikažu na ekranu [36].

U *OcclusionShader* sjenilu, u *Pass* bloku je postavljena direktiva 'Blend Zero One' koja određuje kako se boje kombiniraju pri iscrtavanju. 'Zero' znači da se izvorišni čimbenik, tj. boja izvornog piksela, u potpunosti ignorira (pomnožena je s nulom). To znači da boja izvornog piksela nema utjecaja na konačnu boju piksela. 'One' znači da se boja iz odredišnog piksela koristi u potpunosti (pomnožena je s jedinicom). Ovako definirano sjenilo ne doprinosi konačnom izlazu, odnosno, transparentno je, ali može sakrivati objekte koji se nalaze iza njega jer i dalje zadržava informacije o dubinskom spremniku (engl. *depth buffer*).

Dubinski spremnik koristi se kako bi se odredila vidljivost svakog objekta [37]. Prilikom iscrtavanja scene, dubinska vrijednost svakog piksela (udaljenost od kamere) uspoređuje se s vrijednostima pohranjenim u dubinskom spremniku što određuje treba li piksel biti prikazan ili ga je prekrio neki drugi objekt koji je bliže kameri. Blendiranje utječe samo na to kako se boje kombiniraju tijekom iscrtavanja, ali ne utječe na proces ispitivanja dubine. Ispitivanje dubine ostaje zaseban korak u kojem se provjerava dubinski spremnik kako bi se utvrdilo treba li piksel biti nacrtan na temelju njegove dubine u odnosu na već nacrtane piksele. Zato *OcclusionShader* skriva objekte koji se nalaze iza njega, iako je transparentno.

Nakon izrade *OcclusionShader* sjenila, potrebno je u Unityjevoj razvojnoj okolini stvoriti novi materijal te mu dodati to sjenilo. Dalje, materijal je potrebno dodati na geometrijski lik koji aproksimira lice.

Isti postupak treba primijeniti na sve objekte od kojih se očekuje slično ponašanje, odnosno trebaju se djelomično sakriti iza lica.

Slika 4.21 prikazuje stanje proširene stvarnosti kada su u sceni dvije kamere te kada se kao AR objekt koristi objekt opisan u ovom poglavlju. Na slici se vidi da se na ovaj način stvara iluzija da se objekt omata oko glave.

Slika 4.21. Skrivanje stražnjeg dijela krune

4.2.5.6 Rezultati

U prethodnim cjelinama opisana su rješenja za probleme na koje se može naići pri implementaciji sustava za proširenu stvarnost koji koristi prepoznavanje značajki lica. Nakon eksperimentiranja sa skaliranjem s obzirom na udaljenost od kamere, dodavanja druge kamere koja iscrtava samo AR objekte u scenu te prilagođavanja AR objekata tako da daju iluziju omatanja oko glave, proširena stvarnost daje zadovoljavajuće rezultate.

Ovako implementirana proširena stvarnost omogućava instanciranje i 2D i 3D objekata.

Na Slika 4.22 prikazan je 2D objekt – oblačić za govor. U oblačić za govor može se proizvoljno dodati tekst, ili u Unityjevoj razvojnoj okolini ili iz skripte.

Slika 4.22. 2D proširena stvarnost

U nekim slučajevima poželjno je da se AR objekt ne rotira, nego da samo prati lice korisnika. Za to se mogu iskoristiti svojstva iz *ARObjectData* koja daju informaciju u kojem se smjeru objekt smije rotirati. Na Slika 4.23 prikazana je 2D proširena stvarnost ako je onemogućena rotacija u svim smjerovima. Bez obzira na promjenu rotacije lica, AR objekt se ne rotira.

Slika 4.23. 2D proširena stvarnost bez rotacije

Na Slika 4.24 prikazana je 2D proširena stvarnost ako je omogućena rotacija s obzirom na z-os. Oblačić za govor se rotira samo u smjeru naginjanja glave. Pri rotiranju glave s obzirom na x ili y-os, rotacija oblačića za govor se ne mijenja.

Slika 4.24. 2D proširena stvarnost uz rotaciju

Na Slika 4.25 prikazana su dva 3D AR objekta. Šešir je postavljen da prati vrh glave, a brkovi prate vrh usta. Za oba objekta omogućena je rotacija u svim smjerovima kako bi se stvorilo realistično AR iskustvo.

Slika 4.25. 3D proširena stvarnost s dva AR objekta u sceni

4.3. Korištenje implementiranih modula u postojećim igrama

Kako bi se integrirali moduli za prepoznavanje značajki lica u projekt, prvo je potrebno uvesti *MediapipeUnity* plugin prema uputama u *4.2.1. Priprema Unity okruženja*.

Kako bi se olakšala i ubrzala integracija modula za prepoznavanje značajki lica u druge projekte, može se stvoriti *Prefab* objekt. *Prefab* je predložak na temelju kojeg se može raditi

više objekata s istim svojstvima [38]. Neka se *prefab* za integraciju modula za prepoznavanje značajki lica zove *Solution*.

Objekt *Solution* mora sadržavati sve skripte za rad s prepoznavanjem značajki lica. Na *Solution* objektu moraju biti postavljene sljedeće klase: *FaceMeshGraph* (za prepoznavanje značajki lica), *FaceEvents* klasa (za rukovanje događajima vezanim uz prepoznavanje lica), *UnityMainThread* (za rukovanje operacijama na glavnoj dretvi), *FaceAR* (za stvaranje proširene stvarnosti, ako je ona u sceni potrebna) te dodatne klase koje će služiti kao posrednici između modula za prepoznavanje značajki lica i same logike igre.

Solution objekt je lako uključiti u svaki projekt jer je *prefab*, što znači da je unaprijed konfiguriran i može se jednostavno povući i ispustiti u bilo koju scenu unutar *Unity* okruženja.

5. Eksperimentalna evaluacija

Razvijeni moduli koji koriste prepoznavanje značajki lica predstavljaju alat koji značajno može unaprijediti korisničko iskustvo u igrama. Ovi moduli mogu se integrirati u bilo koju igru.

S obzirom da je primjena tehnologija koje se oslanjaju na prepoznavanje značajki lica u obrazovne svrhe još uvijek rijetka, a postoji veliki potencijal u njezinoj primjeni (*2.2. Prednosti korištenja novih tehnologija u ozbiljnim igrama*), za primjer integracije modula u igre odabrana je postojeća ozbiljna igra zelene tematike, *Eko Zeko*, koja je razvijena u sklopu završnog rada [39].

U ovom poglavlju opisan je postupak integracije modula koji koriste prepoznavanje značajki lica u postojeću zelenu ozbiljnu igru *Eko Zeko.* Opcija upravljanja značajkama lica može se uključiti i isključiti u postavkama igre (Slika 5.1).

Slika 5.1. Postavke igre *Eko Zeko*

5.1. Eko Zeko

Eko Zeko je ozbiljna 2D računalna igra koja educira korisnika o ispravnom gospodarenju otpadom. *Eko Zeko* je rezultat završnog rada *Prototip ozbiljne igre za podizanje svijesti o klimatskim promjenama* koji je nastao na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu.

U nastavku ovog poglavlja je opisano kako se igra ponaša kada nije uključena opcija upravljanja značajkama lica.

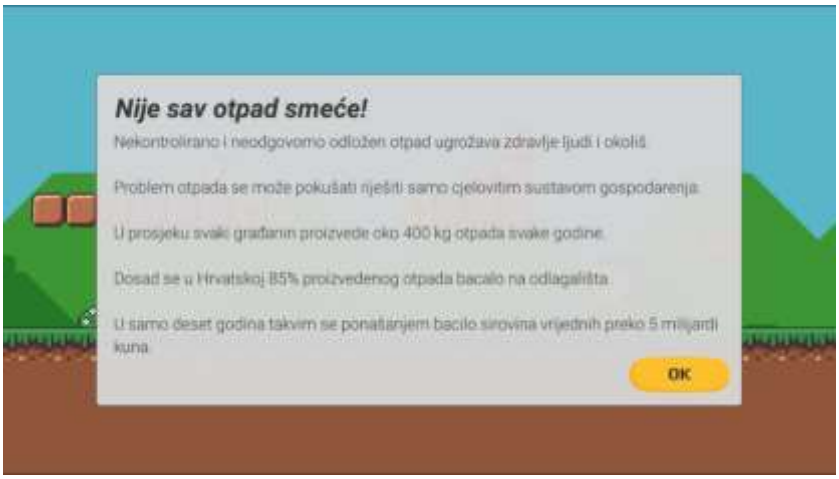
Slika 5.2 prikazuje početni ekran igre *Eko Zeko*.

Slika 5.2. Početni ekran igre *Eko Zeko*

Sama igra podijeljena je u dva dijela. U prvom dijelu igre (Slika 5.3) korisnik tipkovnicom kontrolira lik zečića koji trči i skače po platformama te tako skuplja otpad. Cilj prvog dijela igre je skupiti što više otpada u što kraćem vremenu kako bi se ostvario što bolji rezultat.

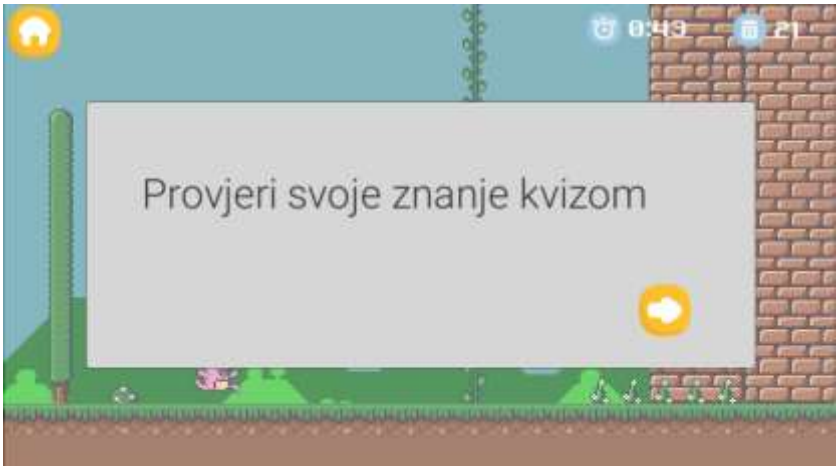
Slika 5.3. Prvi dio igre

Kako se skuplja otpad i prolazi kroz razinu, nailazi se na narančaste kutije s upitnikom. Skokom ispod kutije otvara se ekran s edukacijskim sadržajem. Primjer edukacijskog sadržaja može se vidjeti na Slika 5.4.



Slika 5.4. Edukacijski sadržaj

Drugi dio igre čini kviz koji se pokreće kada korisnik dosegne kraj razine (Slika 5.5). U kvizu se provjerava činjenično znanje o gospodarenju otpadom. Odgovor na svako pitanje iz kviza može se pronaći na jednom od edukacijskih ekrana kojeg se moglo otvoriti izborom neke narančaste kutije.



Slika 5.5. Početak kviza

Primjer jednog pitanja u kvizu može se vidjeti na Slika 5.6.

Slika 5.6. Pitanje u kvizu

Ako korisnik odabere točan odgovor, onda se taj gumb oboja u zelenu boju, pusti se odgovarajući zvuk i omogući prelazak na sljedeće pitanje kako je prikazano na Slika 5.7.

Slika 5.7. Točan odgovor

Ako korisnik ne odabere točan odgovor, gumb se oboja u crvenu boju, pusti se odgovarajući zvuk i prikaže se poruka „Klikom na x, izađi iz kviza i potraži točan odgovor.“ kako je prikazano na Slika 5.8.

Slika 5.8. Netočan odgovor

Može se izaći iz kviza kako bi se potražio točan odgovor, a povratak na nastavak kviza moguć je klikom na gumb s upitnikom kako je prikazano na Slika 5.9.

Slika 5.9. Privremeni izlazak iz kviza

Kada se riješe sva pitanja u kvizu, dolazi se do kraja igre. Ekran na kraju igre (Slika 5.10) prikazuje rezultat te opcije za ponovno pokretanje razine, povratak na početni ekran ili izlazak iz igre.

Slika 5.10. Kraj igre

5.2. Interakcija s korisničkim sučeljem

U postavkama igre može se uključiti i isključiti opcija prepoznavanja značajki lica. Ako je opcija isključena, onda se za interakciju s korisničkim sučeljem koristi miš. Ako je opcija prepoznavanja značajki lica uključena, dodatno se koriste geste i izrazi lica za kretanje i odabir opcija na korisničkom sučelju.

U igri *Eko Zeko*, interakcija s korisničkim sučeljem uz pomoć prepoznavanja značajki lica funkcionira na sljedeći način: uvijek je označen jedan gumb na korisničkom sučelju što se indicira drugačijom bojom gumba (ovdje plavom), lijevim namigom označava se gumb koji prethodi trenutno označenom gumbu, a desnim namigom se označava gumb koji je sljedeći od trenutno označenog gumba. Treptajem se klikće na označeni gumb. Na Slika 5.11 je prikazano stanje igre kada je označen gumb za pokretanje igre.

Slika 5.11. Označen gumb za pokretanje igre

Za implementaciju ovakvog ponašanja, potrebno je prvo na svaki gumb dodati komponentu *ButtonSelect* u kojoj se definira prethodni i sljedeći gumb, kao i izgled gumba kada je označen i kada nije što omogućava da se označeni gumb prikazuje drugačije od ostalih te da se lako modificira njegov izgled. Također, komponenta ima varijablu koja prati je li gumb označen. Za ispravno kretanje po korisničkom sučelju, potrebno je gumbe dobro ulančati u listu tako da nijedan gumb nije nedohvatljiv.

Za kretanje po korisničkom sučelju potrebno je na *Solution* objekt dodati novu klasu (*FaceUIController*) koja sadrži logiku za označavanje gumba. U skripti *FaceUIController* kontinuirano se prate aktivni gumbi u sceni. *FaceUIController* se pretplaćuje na detekciju lijevog namiga, desnog namiga i treptanja.

Kada klasa *FaceEvents* aktivira događaj desni namig, potrebno je izvesti logiku za označavanje sljedećeg gumba. To uključuje promjenu slike trenutno označenog gumba u onu koja znači da gumb nije označen te označavanje sljedećeg gumba i promjenu njegove slike u onu koja znači da je gumb označen. Analogno treba napraviti kada se aktivira događaj lijevi namig. Kada se detektira treptaj, samo je potrebno na trenutno označenom gumbu pokrenuti sve metode koje su navedene u njegovom *OnClick* svojstvu.

Ovako implementirana interakcija s korisničkim sučeljem omogućava korisniku da upravlja sučeljem samo pomoću svojeg lica, bez potrebe za korištenjem miša.

5.3. Upravljanje igračem

U postavkama igre može se uključiti i isključiti opcija prepoznavanja značajki lica. Ako je opcija isključena, koristi se tipkovnica za pokretanje igrača. Ako je opcija prepoznavanja značajki lica uključena, onda se dodatno koriste geste i izrazi lica za pokretanje igrača.

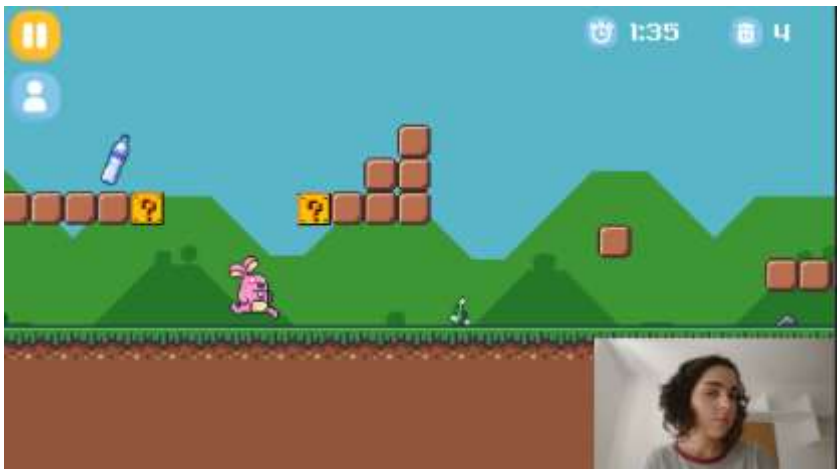
Lik u igri *Eko Zeko* korištenjem prepoznavanja značajki lica kontrolira se na sljedeći način. Lik se kreće udesno kada lice korisnika gleda udesno, kreće se ulijevo kada lice korisnika gleda ulijevo, a skače kada korisnik otvori usta.

Za upravljanje likom u igri potrebno je stvoriti novu skriptu (*PlayerController*) i dodati ju na samog lika. U skripti *PlayerController* potrebno je pretplatiti se na događaje detektirane u *FaceEvents* klasi i povezati ih s odgovarajućim akcijama u igri. Za kretanje igrača lijevo-desno, potrebno je pretplatiti se na događaje koji dojavljuju o promjeni orijentacije lica s

obzirom na y-os (engl. *yaw*), a za skakanje igrača, potrebno je pretplatiti se na događaj koji dojavljuje o otvaranju usta korisnika.

Budući da igra već ima implementiranu logiku za kretanje lika, u skripti *PlayerController* potrebno je samo pozvati odgovarajuće metode kada se detektiraju odgovarajući događaji. Kada se detektira promjena nagiba lica u odnosu na y-os, poziva se funkcija *Move* s argumentom koji određuje smjer kretanja lika, a kada se detektira otvaranje usta, poziva se funkcija *Jump*.

Na Slika 5.12 prikazan je jedan kadar kretanja lika udesno. Lik se kreće udesno jer je lice korisnika orijentirano udesno.



Slika 5.12. Kretanje lika

Na Slika 5.13 prikazan je skok lika jer su usta korisnika otvorena.

Slika 5.13. Skok lika

Ovako implementirano pokretanje lika u igri omogućava korisniku da njime upravlja samo pomoću svojeg lica, bez potrebe za korištenjem tipkovnice.

5.4. Proširena stvarnost

Za integraciju proširene stvarnosti samo je potrebno da *Solution* objekt na sebi, uz sve ostale klase potrebne za prepoznavanje značajki lica, obvezno ima i klasu *FaceAR*. Klasi *FaceAR* potrebno je predati *ARObjectData* koji određuje koji objekt treba pratiti koju točku lica.

U igri *Eko Zeko*, proširena stvarnost služi kao nagrada korisniku za završetak igre. Na Slika 5.14 prikazana je proširena stvarnost koju korisnik vidi na završnom ekranu igre. Za postizanje ovog efekta, kreiran je *ARObjectData* sa zečjim ušima kao objektom za instanciranje. Zečje uši prate vrh glave. Objekt se rotira u svim smjerovima kako bi što bolje pratio orijentaciju lica i tako stvorio realistično AR iskustvo.

Slika 5.14. Proširena stvarnost - zečje uši

5.5. Osvrt na integrirano rješenje

Korištenje gesti i izraza lica za interakciju s igrom značajno povećava pristupačnost igre jer omogućuje upravljanje igrom bez korištenja ruku što je posebno korisno za korisnike s motoričkim poteškoćama. Ovakav način upravljanja igrom, zajedno s proširenom stvarnosti, čini igru privlačnijom za korisnike. Ova tehnologija također povećava interaktivnost igre čime se povećava i angažiranost korisnika što može doprinijeti boljem usvajanju edukacijskog sadržaja u ozbiljnim igrama.

Međutim, postoje i nedostatci u razvijenom rješenju. Prepoznavanje značajki lica može biti manje precizno u uvjetima slabog osvjetljenja. Ako sustav za prepoznavanje lica ne radi ispravno, korisnici mogu biti frustrirani što onda negativno utječe na njihovo iskustvo i percepciju igre.

Implementirano programsko rješenje koje koristi prepoznavanje značajki lica moglo bi se dalje unaprijediti na razne načine. Uvođenje dodatnih izraza i gesti lica moglo bi povećati fleksibilnost igre jer bi korisnici mogli prilagoditi koje izraze i geste žele koristiti za pokretanje akcija u igri. Također, kao što je implementirano u projektu *Gameface* (*2.3.2 Project Gameface*), trebalo bi dodati mogućnost prilagođavanja koje geste se mapiraju na koje akcije. Osim toga, korisnicima bi trebalo omogućiti prilagodbu intenziteta geste koja aktivira odgovarajući događaj. Dodatno, po uzoru na [19], mogao bi se implementirati sustav koji detektira raspoloženje i razinu angažiranosti korisnika te na temelju toga prilagođava težinu igre.

Razvijeno programsko rješenje trebalo bi se integrirati u postojeće ozbiljne igre. Zato postoji veliki potencijal u suradnji s projektom *Play2Green*.

Play2Green[7] je projekt u okviru programa ERASMUS+ koji se fokusira na razvoj zelenih ozbiljnih igara i njihovu primjenu u obrazovanju. *Play2Green* razvija zelene ozbiljne igre koje koriste nove tehnologije poput umjetne inteligencije i proširene stvarnosti. Cilj *Play2Green* projekta je podizanje svijesti o okolišu i borbi protiv klimatskih promjena, a tehnologija prepoznavanja značajki lica može pomoći u tome. Kako je pokazano na primjeru igre *Eko Zeko,* prepoznavanje značajki lica bi se moglo iskoristiti u zelenim ozbiljnim igrama za upravljanje igrom te za povećavanje zabavnog aspekta igre kroz korištenje proširene stvarnosti. Na taj način povećala bi se interaktivnost i pristupačnost ozbiljnih igara.

[7] Play2Green (Serious Gaming for Universal Access to Green Education) [2022-1-HR01-KA220-HED-000088675], https://sociallab.fer.hr/play2green

6. Zaključak

Prepoznavanje značajki lica je grana računalnog vida koja se bavi detekcijom značajnih točaka lica. Na temelju istraživanja postojećih rješenja koja koriste prepoznavanje značajki lica, može se uočiti da nedostaje programskih rješenja koja koriste ovu tehnologiju u obrazovne svrhe. Također, postojeća programska rješenja uglavnom su namijenjena za korištenje na mobilnim uređajima, stoga je poseban naglasak stavljen na primjenu ove tehnologije u ozbiljnim računalnim igrama. Općenito, integracija novih tehnologija u ozbiljne igre donosi značajne koristi jer povećava interaktivnost i angažiranost korisnika što može značajno pojačati edukacijska vrijednost igara.

U okviru ovog rada razvijeno je programsko rješenje koje demonstrira neke mogućnosti tehnologije prepoznavanja značajki lica. Razvijeno programsko rješenje sastoji se od 4 modula: modul za prepoznavanje značajki lica koji dohvaća koordinate točaka lica, modul za detekciju gesti i izraza lica koji koristi koordinate točaka na licu za identifikaciju gesti, izraza i orijentacije lica, modul za upravljanje igrom koji na temelju detektiranih gesti i izraza lica provodi odgovarajuće akcije unutar igre te, na kraju, modul za proširenu stvarnost koji na lice korisnika superponira 2D i 3D virtualne objekte. Nakon evaluacije postojećih tehnologija za prepoznavanje značajki lica u Unity razvojnom okruženju, za implementaciju modula odabran je MediapipeUnity plugin koji omogućava integraciju Mediapipe radnog okvira u Unity.

Programsko rješenje razvijeno je tako da se može uključiti u bilo koju postojeću igru i tako pružiti mogućnost upravljanja igrom korištenjem značajki lica te stvaranja iskustva proširene stvarnosti na licu korisnika. Moduli su testirani na ozbiljnoj igri *Eko Zeko* koja educira korisnika o važnosti gospodarenja otpadom.

Daljnji razvoj programskog rješenja koji koristi prepoznavanje značajki lica mogao bi uključiti dodatne funkcionalnosti poput detekcije emocija kako bi se procijenila korisnikova koncentracija te, na temelju toga, prilagodila težina edukacijskog materijala. Također, trebala bi se uvesti mogućnost prilagođavanja gesta koje pokreću specifične akcije kako bi korisnik mogao odabrati gestu koja je njemu najintuitivnija. Najvažnije, tehnologija prepoznavanja značajki lica trebala bi se integrirati u što veći broj postojećih ozbiljnih igara kako bi se povećala njihova edukacijska moć, interaktivnost i pristupačnost.

Literatura

[1] P. Antoniadis, *Introduction to Landmark Detection*, Baeldung, (2024, ožujak). Poveznica: https://www.baeldung.com/cs/landmark-detection; pristupljeno 8. lipnja 2024.

[2] N. Kumar Rao B, N. Panini Challa; E. Phalguna Krishna, S. Sreenivasa Chakravarthi *Facial Landmarks Detection System with OpenCV Mediapipe and Python using Optical Flow (Active) Approach.* Proceeding of the 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), (2023)

[3] Y. Wue, Q. Ji, *Facial Landmark Detection: A Literature Survey*, International Journal of Computer Vision, 127, (2019), str. 115-142

[4] *Facial Recognition Technology*, Innovatrics. Poveznica: https://www.innovatrics.com/facial-recognition-technology/; pristupljeno: 8. lipnja 2024.

[5] *What is Facial Recognition,* AWS. Poveznica: https://aws.amazon.com/what-is/facial-recognition/; pristupljeno: 8. lipnja 2024.

[6] S. Sowden, B. Schuster, C. Keating, D. Fraser, J. Cook, *The Role of Movement Kinematics in Facial Emotion Expression Production and Recognition,* (2021)

[7] *What is Augmented Reality (AR)?, SAP.* Poveznica: https://www.sap.com/products/scm/industry-4-0/what-is-augmented-reality.html; pristupljeno: 9. lipnja 2024.

[8] *Dlib*, Dlib C++ Library, (2022, svibanj). Poveznica: http://dlib.net/, pristupljeno: 9. lipnja 2024.

[9] A. Preet Gulati, *Face Detection Using the DLIB Face Detector Model*, Analytics Vidhya, (2022, travanj). Poveznica: https://www.analyticsvidhya.com/blog/2022/04/face-detection-using-the-dlib-face-detector-model/, pristupljeno: 9. lipnja 2024.

[10] *Top 10 Open Source Facial Recognition libraries and tools*, Twine, (2023, studeni). Poveznica: https://www.twine.net/blog/top-10-open-source-facial-recognition-library-and-tools/; pristupljeno: 9. lipnja 2024.

[11] *Top Computer Vision Libraries,* Saiwa, (2024, travanj). Poveznica: https://saiwa.ai/blog/top-computer-vision-libraries/; pristupljeno: 9. lipnja 2024.

[12] A. Ving, *What is Meidapipe?*, Roboflow, (2024, travanj). Poveznica: https://blog.roboflow.com/what-is-mediapipe/; pristupljeno: 10. lipnja 2024.

[13] P. Caserman, K. Hoffmann, P. Müller, M. Schaub, K. Straßburg, J. Wiemeyer, R. Bruder, S. Göbel, *Quality Criteria for Serious Games: Serious Part, Game Part, and Balance*, (2020, srpanj)

[14] P. Penades, *How Instagram and TIKTOK use AI in their filters to boost communities,* FutureJobs, (2023, kolovoz). Poveznica: https://www.futurejobs.ai/blog/how-instagram-and-tiktok-use-ai-in-their-filters-to-boost-communities; pristupljeno: 22. lipnja 2024.

[15] M. De Andres-Clavera, L. Moroney, *Introducing Project Gameface: A hands-free, AI-powered gaming mouse,* (2023, svibanj). Poveznica: https://blog.google/technology/ai/google-project-gameface/; pristupljeno: 10. lipnja 2024.

[16] A. Singh, S. Jin, L. Carr, *Project GameFace makes gaming accessible to everyone*, (2023, lipanj). Poveznica: https://developers.googleblog.com/en/project-gameface-makes-gaming-accessible-to-everyone/; pristupljeno: 10. lipnja 2024.

[17] D. Cordeiro, N. Correia, R. Jesus, *ARZombie: A Mobile Augmented Reality Game with Multimodal Interaction*. Proceedings of the 7th International Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN), (2015)

[18] S. Koyama. *Estimation of Students' Level of Engagement from Facial Landmarks.* Istraživački izvještaj. Zavod za informacijske znanosti, Sveučilište Okayama, 2023.

[19] *ARFoundation*, Unity. Poveznica: https://docs.unity3D.com/Packages/com.unity.xr.arfoundation@5.0/manual/index.html; pristupljeno: 11. lipnja 2024.

[20] *MARS UI overview*, Unity . Poveznica: https://docs.unity3D.com/Packages/com.unity.mars@1.0/manual/UIOverview.html; pristupljeno: 11. lipnja 2024.

[21] *FaceTracking,* Unity. Poveznica: https://docs.unity3D.com/Packages/com.unity.mars@1.0/manual/FaceTracking.html; pristupljeno: 11. lipnja 2024.

[22] *PythonForUnity*, Unity. Poveznica: https://docs.unity3D.com/Packages/com.unity.scripting.python@2.0/manual/index.html; pristupljeno: 11. lipnja 2024.

[23] *UnityPython*, GitHub. Poveznica: https://github.com/exodrifter/unity-python; pristupljeno: 11. lipnja 2024.

[24] *OpenCVplusUnity,* AssetStore. Poveznica: https://assetstore.unity.com/packages/tools/integration/opencv-plus-unity-85928; pristupljeno: 11. lipnja 2024.

[25] *OpenCVForUnity*, AssetStore. Poveznica: https://assetstore.unity.com/packages/tools/integration/opencv-for-unity-21088; pristupljeno: 11. lipnja 2024.

[26] *Introduction to Face AR plugin for Unity*¸ Banuba SDK. Poveznica: https://docs.banuba.com/face-ar-sdk-v1/unity/unity_overview; pristupljeno: 12. lipnja 2024.

[27] *Powering the immersive web,* Zapworks. Poveznica: *https*://zap.works; pristupljeno: 12. lipnja 2024.

[28] *MediaPipeUnityPlugin*, GitHub. Poveznica: https://github.com/homuler/MediaPipeUnityPlugin; pristupljeno: 13. lipnja 2024.

[29] *Euclidian Distance,* Britannica, (2024, veljača). Poveznica: https://www.britannica.com/science/Euclidean-distance; pristupljeno: 20. lipnja 2024.

[30] *Events*, Unity Learn, (2013). Poveznica: https://learn.unity.com/tutorial/events-uh#; pristupljeno: 15. lipnja 2024.

[31] J. Albahari, *Threading in C#*, (2011, travanj). Poveznica: https://www.albahari.com/threading/; pristupljeno: 16. lipnja 2024.

[32] *ScriptableObject*, Unity Documentation (2015). Poveznica: https://docs.unity3D.com/Manual/class-ScriptableObject.html; pristupljeno: 15. lipnja 2024.

[33] *Camera*, Unity Documentation (2015). Poveznica: https://docs.unity3d.com/510/Documentation/Manual/class-Camera.html; pristupljeno: 18. lipnja 2024.

[34] *Shader*, Wikipedia. Poveznica: https://en.wikipedia.org/wiki/Shader; pristupljeno: 21. lipnja 2024.

[35] *ShaderLab Blending,* Unity Documentation (2021). Poveznica: https://docs.unity3d.com/2018.4/Documentation/Manual/SL-Blend.html; pristupljeno: 21. lipnja 2024.

[36] *Framebuffer,* Wikipedia. Poveznica: https://en.wikipedia.org/wiki/Framebuffer; pristupljeno: 21. lipnja 2024.

[37] *Depth Buffer*, Valve Developer Community (2024, siječanj). Poveznica: https://developer.valvesoftware.com/wiki/Depth_buffer; pristupljeno: 21. lipnja 2024.

[38] Prefabs, unity3d.com. Poveznica: https://docs.unity3d.com/Manual/Prefabs.html; pristupljeno: 23. lipnja 2024.

[39] M. Jurić, *Prototip ozbiljne igre za podizanje svijesti o klimatskim promjenama*. Završni rad. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2022.

Sažetak

Naslov: Prepoznavanje značajki lica u ozbiljnim igrama

Prepoznavanje značajki lica je područje računalnog vida koje se fokusira na detekciju specifičnih točaka na licu. Postoji manjak programskih rješenja koja koriste ovu tehnologiju u obrazovne svrhe. Zato je razvijeno je modularno programsko rješenje koje se može uključiti u postojeće igre kako bi pružilo mogućnost rada s prepoznavanjem značajki lica. Rješenje se sastoji od modula za prepoznavanje značajki lica, modula za detekciju orijentacije, gesti i izraza lica, modula za upravljanje igrom te modula za proširenu stvarnost. Za implementaciju prepoznavanja značajki lica u igrama korištena je razvojna okolina Unity i MediapipeUnity plugin, alat otvorenog koda koji omogućuje primjenu algoritama za računalni vid u Unityevoj razvojnoj okolini. Evaluacija rješenja provedena je kroz integraciju s ozbiljnom zelenom igrom *Eko Zeko* te su u radu dokumentirani prijedlozi za poboljšanje. Razvijeno programsko rješenje također ima potencijal za integraciju s projektom *Play2Green* (identifikator projekta: 2022-1-HR01-KA220-HED-000088675), koji razvija zelene ozbiljne igre koje koriste nove tehnologije.

Ključne riječi: prepoznavanje značajki lica, ozbiljne igre, Mediapipe, Unity, proširena stvarnost

Summary

Title: Face Landmark Detection in Serious Games

Face landmark detection is a field of computer vision that focuses on dentecion of specific points on a face. There is a lack of software solutions that use this technology for educational purposes. Therefore, a modular software solution has been developed than can be integrated into existing games to provide the capability of working with face landmark detection. The solution consists of modules for face landmark detection, gesture and facial expression detection, game management via face landmarks and augmented reality. Unity platform and MediapipeUnity plugin, an open-source tool that enables the application of computer vision algorithms in Unity, were utilized to implement the solution. The solution was evaluated through integration with the existing serious green game, *Eko Zeko*. Suggestions for further improvement are documented in the paper. The developed software solution also has the potential for integration with the *Play2Green* (project identifier: 2022-1-HR01-KA220-HED-000088675) project, which develops green serious games that use new technologies.

Keywords: face landmark detection, serious games, Mediapipe, Unity, augmented reality