Sadržaj

Uvod 5

1. Alati i tehnologije 6

1.1. Visage|SDK 6

1.1.1. Visage Tracker Unity Plugin 6

1.1.2. Face Tracker 2 7

1.2. Unity3D 7

1.3. Photoshop 8

2. Karikatura 9

3. Razvoj i komponente aplikacije 11

3.1. Model savršenog lica 11

3.2. Točke lica 12

3.3. Izrada maske karikature 14

3.4. Razmještaj osnovnih dijelova lica 16

3.4.1. Oči 16

3.4.2. Nos 17

3.4.3. Usta 19

4. Aplikacija 21

5. Problemi i potencijalna poboljšanja 24

Zaključak 25

Literatura 26

Sažetak 27

Summary 28

Uvod

U svakodnevim medijima često se susrećemo s karikaturama koje prikazuju šaljivi likovni prikaz osobe koji preuveličava i deformira istaknute osobine portretiranog pojedinca. Nerijetko se koriste za ismijavanje ljudi naglašavajući njihove fizičke mane, ali prvobitni cilj portret karikature je napraviti zabavnu ilustraciju na kojoj je lako uočljivo koju osobu ona predstavlja [1]. Poznati karikaturisti tvrde da se stvaranje karikature temelji na naglašavanju uočenih razlika između osobe i norme. Norma naravno nije propisana, već je subjektivna i svaki umjetnik je drugačije shvaća. Ona je zapravo viđenje općenitih ili čak savršenih proporcija lica.

Unatoč svojeg šaljivog sadržaja, karikature lica su problematične jer ih je teško proizvesti. Potrebni su izuzetno nadareni umjetnici da prenesu bit karikature, a uz to cijeli proces može dugo potrajati. Nadalje, dobiveni rezultati bit će, kao i svaka ručno izrađena umjetnost, donekle nedosljedni.

Računalni vid je područje umjetne inteligencije koje obuhvaća metode za obradu, analiziranje i razumijevanje slike. Koristi se za stjecanje zanimljivih i prakičnih informacija iz fotografija. Područje primjene seže od industrijskih sustava za strojni vid, na primjer, pregledavanje boca na proizvodnoj liniji, do istraživanja umjetne inteligencije i računala ili robota koji mogu shvatiti stvarni svijet oko njih. [citat] Temelj koji najbolje opisuje računalni vid je stvaranje automatskog sustava koji simulira ljudski vid i način na koji ljudi percipiraju okolinu.

Spajanjem ideje o nastanku karikature i korištenjem računalnog vida za prepoznavanje ljudskog lica i njegovih karakterističnih crta, nastao je ovaj rad. Svrha ovog završnog rada je izrada aplikacije koja u realnom vremenu preko lica iscrtava masku karikature.

Aplikacija se temelji na prepoznavanju karakterističnih točki lica subjekta i izračunavanju koliko svaka od karakterističnih točaka lica odudara od zadanog modela savršenog lica. Potom pomiče određene točke na maski u mjeri izračunatog odudaranja te nastaje izobličena maska koja predstavlja karikaturu. S ciljem da karikatura bude što uvjerljivija, potrebno je obratiti pozornost i na razmještaj osnovnih dijelova lica, poput pozicije očiju, nosa i usana.

1. Alati i tehnologije

U svrhu rada napravljena je aplikacija u mehanizmu za igre Unity3D uz korištenje funkcionalnosti razvojnog okruženja Visage|SDK. Za implementaciju bilo je potrebno modificirati FaceTracker2 i napisati Unity skripte za što je korišten Microsoft Visual Studio 2015. Adobe Photoshop je odabran grafički računalni program za razvijanje teksure maske. U nastavku su opisane glavne komponente potrebne za izradu rada i njihove funkcionalnosti koje se koriste u aplikaciji.

1.1. Visage|SDK

Visage | SDK je razvojni alat koji obuhvaća skup tehnologija temeljenima na računalnom vidu. Omogućuje praćenje, analiziranje i prepoznavanje ljudskih lica u stvarnom vremenu. Tehnologije su podijeljene u tri specijalizirana paketa koje mogu na slikama, videima ili u realnom vremenu određivati sljedeće značajke: FaceTrack koji je specijaliziran za detekciju ljudskog lica i crta lica uključujući položaj glave, FaceAnalysis koji detektira spol i emocije i FaceRecognition koji mjeri sličnost lica i prepoznaje identitete. Visage | SDK je dostupan na svim glavnim platformama pa tako podržava i Windows i Unity platformu. Od navedenih paketa korišten je FaceTrack u sklopu dodatka VisageTrackerUnityPlugin koji omogućuje korištenje alata za analizu lica u sklopu Unity3D sustava.

1.1.1. Visage Tracker Unity Plugin

VisageTrackerUnityPlugin je C# omotač koji omogućuje upotrebu različitih funkcionalnosti iz Visage | SDK biblioteke. Projekt je dizajniran za integriranje u druge programe i ne može se samostalno pokrenuti. Korišten je u sklopu Unity3D projekta kao alat koji omogućuje VisageTracker, odnosno tragač, koji sadrži funkcionalnosti poput praćenja lica, odnosno njegove translatacije i rotacije i dobivanje karakterističnih točaka lica.

VisageTracker je alat koji može pratiti poziciju glave, crte lica i smjer pogleda za više lica u slici, video zapisu, kamere ili drugih izvora. Svaku sliku, odnosno kadar potrebno je prosljeđivati uzastopno `VisageTracker :: track ()` metodi koja vraća sljedeće

značajke kao dio `FaceData` objekta za zadani kadar: 3D poziciju glave, izraz lica, informacije o smjeru pogleda, informaciju je li oko zatvoreno, radijus irisa, značajke lica i još neke druge. `FaceData` objekt koristi se kao spremnik za sve rezultate praćenja i otkrivanja lica. Za ovaj rad najvažnija metoda tog objekta je metoda `faceModelVertices` koja vraća popis koordinata vrhova 3D modela lica, koje će se kasnije mijenjati i uz određenu teksturu stvarati karikaturu.

1.1.2. Face Tracker 2

FaceTracker2 je demo primjer u sklopu razvojnog okružejna koji demonstrira mogućnosti Visage | SDK poput praćenja značajki glave i lica u stvarnom vremenu na više lica iz video datoteke, kamere ili slike [3]. Temelji se na `VisageSDK::VisageTracker` koji je pisan u C++ programskom jeziku. FaceTracker2 uključuje kod koji implementira upisivanje podataka praćenja u tekstualnu datoteku. Ova funkcionalnost je iskorištena tako da iz slike modela savršenog lica zapisuje popis koordinata vrhova 3D modela lica koji se kasnije uspoređuje s istim podacima korisnika aplikacije.

1.2. Unity3D

Unity3D je *game engine* čiji je primarna funkcija izrada računalnih igara, a pogodan je i za izradu simulacija ili različitih vizualizacija. Posebice je popularan jer je prilagođen za razvoj na raznim platformama poput PC, konzola, web stranica i razvoj iOS i Android mobilnih igara. Programeri ga vole jer je jednostavan za korištenje zbog C# skriptiranja, ali i umjetnici ga vole, jer dolazi s moćnim alatima za animaciju koji pojednostavljuju stvaranje vlastitih 3D scena ili izradu 2D animacija od nule.

Uz Visage|SDK raspoloživ je primjer Unity3D projekta VisageTrackerUnity Demo, koji olakšava upotrebu Unity sustava novim korisnicima i uvodi ih u korištenje funkcija poput funkcije praćenja lica, analize lica i prepoznavanja lica. Projekt sadrži Unity3D scenu koja omogućava ispobavanje naočala i prikaz maske teksture tigra na licu u realnom vremenu [4]. Dijelovi ovog demo projekta su korišteni i nadograđeni kako bi se ubrzala izrada vlastite aplikacije.

1.3. Photoshop

Adobe Photoshop, ili skraćeno Photoshop, je grafički računalni program, razvijen i izdan od strane američke tvrtke Adobe Systems. Ovo je najpoznatiji računalni program za obradu slike [2]. Ovaj uređivač slike je korišten za izradu teksture koja imitira naslikano lice.

2. Karikatura

Karikatura je svakodnevno prisutna u medijima te je postala i uobičajena pojava. Najčešće se odražava kao usputna zabava koja na šaljivi način prikazuje osobe i događaje iz društvene aktulnosti. Često uz sebe nosi i skrivenu poruku koja karakterizira satirične karikature. Definiciju karikature je sljedeća: „Karikatura je crtež, plastički prikaz ili opis koji, pretjerujući u prikazivanju prirodnih obilježja (preuveličavanje, umanjivanje i iskrivljavanje), pojavu subjekta čini smiješnom ili apsurdnom radi zabave ili kritike, odnosno predstavlja duhoviti (ironički, satirički, metaforički i sl.) komentar određene situacije“ [7].

Izraz karikatura potječe iz talijanskog glagola *caricare* što u prijevodu znači pretjerivanje ili natovarivanje. Također postoji i poveznica s riječima *carattere*, na talijanskom označava karakter i *cara* što na španjolskom znači lice [6]. Dakle, riječ karikatura u biti znači nabijeni portret.

Karikatura se referira na stvarnost kojoj nudi „deformirano zrcalo“ [8]. Ta deformiranost, odnosno miješanje sličnosti i odstupanja od stvarnog izgleda prikazanog motiva daje karikaturi komičnu notu. Karikatura u suštini naglašava razliku od uzora i izruguje mu se sve do pretjerivanja u funkciji naglašavanja te razlike [8].

Mnoge karikature mogu prikazivati cijelo tijelo, no sami temelj i središte karikature je lice koje je i glavni fokus ovog rada. Prilikom stvaranja karikature lica karikaturist kao polazište uzima prenaglašavanje ili djelomično transformiranje portretnih crta lica. Ključni elementi lica su oči, nos i usta jer se naglašavanjem samo tih elemenata može postići prepoznatiljiva karikatura.

Budući da je za stvaranje karikature potreban izuzetni talent karikaturista i određeno vrijeme, pojavila se potražnja za računalnim sustavima koji bi automatski stvarali prikaze karikatura. Korištenjem tehnika računalne grafike postoje već takvi sustavi koji nude alate za iskrivljivanje slike posebno stvorene za brzo stvaranje karikatura. Računalnim grafičkim programima mogu se kreirati i složenije metode stvaranja slika tako da se primjerice koriste mnogo detaljnijim teksturama nego što bi se slika stvorila tradicionalnim metodama.

Prijelomna točka u formalnom definiranju karikature u području računarstva bio je rad Susan Brennan iz 1982. godine [10]. U njezinom sustavu karikatura je formalizirana kao

proces pretjerivanja razlika u odnosu na prosječno lice. Na primjer, ako osoba ima istaknutije oči od prosječne osobe, u njegovoj karikaturi oči će biti puno veće od normalnih. Ova ideja je implementirana na način da se u sustav učita frontalna slika osobe te se generira skica lica snimanjem položaja 165 značajnih točaka na skeniranoj fotografiji lica. Te se značajke zatim uspoređuju s odgovarajućim značajkama prosječnog ili srednjeg lica, što se izračunava prosjekom značajki svih ostalih lica u bazi podataka. Određeno lice se zattim karikira jednostavnim oduzimanjem od određenog lica odgovarajuće točke na srednjem licu i skaliranjem dobivene razlike za faktor. U suštini, premještanjem značajki lica koje odudaraju od prosjeka, generirana je karikatura. Ovaj sustav je i temelj nad kojim je izrađena ova aplikacija.

Drugi pristup izgradnji ovakvih sustava može počivati i na tehnikama strojnog učenja. Tako primjerice Liang [9] tvrdi da se izgled karikature razlikuje ovisno o karikaturistu i da ga se ne može eksplicitno definirati. U svojem radu koristi tehnike automatskog učenja koje oponašaju stilove određenih karikaturista.

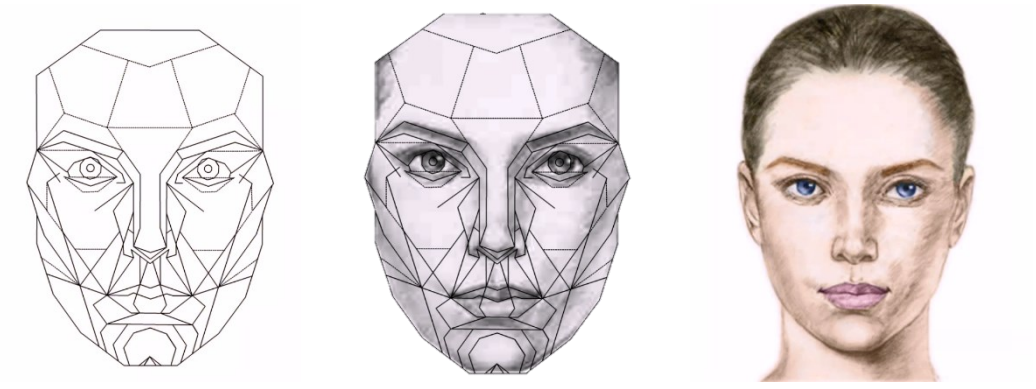
Rezultati dobiveni ovakvim automatskim grafičkim sustavima svkako još nisu iste kvalitete kao radovi umjetnika. Na primjer, mnogobrojni sustavi su ograničeni na samo frontalne poze, dok mnoge ili čak većina ručno izrađenih karikatura sadrže prikaze cijele osobe.

3. Razvoj i komponente aplikacije

3.1. Model savršenog lica

Polazišna točka ovog rada bilo je definiranje osnovnog modela lica, odnosno glave koja će označavati normu. Kao što je ranije navedeno, norma je osnova svih karikatura i zato se važnost dobro odabrane norme ne može podcijeniti. Budući da ovo nije prvi rad na ovu temu, postoje već ranije nastali radovi koji za odabir osnovnog modela koriste prosječno lice. Da bi stvorio pravi prosjek, bilo je potrebno izmjeriti dimenzije lica odgovarajuće velike skupina uzoraka. U ovom radu umjesto prosječnog lica, koristio se model savršenog lica kojeg je stvorio Dr. Stephen R. Marquardt, koji je u svojem istraživanju pokušao znanstveno analizirati matematiku savršene ljepote lica [5]. Njegov rad vuče inspiraciju iz antičkog doba, gdje su Stari Grci smatrali da sva ljepota potječe od zlatnog reza kojeg često nalazimo u prirodi i stvarima koje smatramo lijepima.

Zlatni rez je matematički omjer 1,618: 1. Broj 1.618 još je poznat i pod nazivom *Phi*. Budući da su svi zlatni omjeri jednaki jedni drugima, dva zlatna omjera prikazana kao međusobno jednaka nazivaju se zlatnom sredinom. Na temelju ovih omjera stvorena je mreža geometrijskih likova koji opisuju savršeno lice. Daljnim razvojem mreže, odnosno crtanjem dijelova lica, nastaje i prikaz savršenog lica (Slika 3.1).



Slika 3.1 Proces nastanka savršenog lica

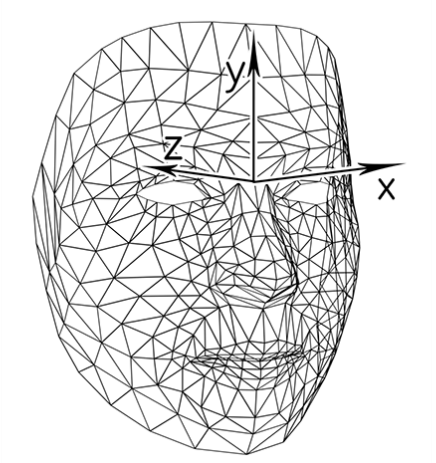
FaceTracker2 iz skupine primjera u sklopu razvojnog okruženja Visage|SDK je korišten kao alata koji je analizirao model lica sa treće slike. Bilo je potrebno modificirati

FaceTracker2, na način da prilikom zapisivanja dobivenih podataka u tekstualnu datoteku, zapisuje koordinate vrhova modela lica. Za navedeni postupak korištena je metoda `faceModelVertices` koja vraća 357 koordinata vrhova koji opisuju zadano lice.

3.2. Točke lica

Kao što je rečeno u prijašnjem odlomku, za detektiranje svih točaka lica, koristi se funkcionalnost Visage|SDK uz VisageTrackerUnityPlugin koji omogućava VisageTracker alatu. U sklopu VisageTracker alata poziva se metoda `VisageTracker :: track ()` nad određenim kadrom te metoda vraća podatke, osim o širini i visini kadra i formatu, podatke obavljene analize lica. Ovi podaci spremljeni su u strukturu naziva `FaceData` koja sadrži podatke o translataciji glave u odnosu na kameru, orijentaciji i nagibu glave, smjeru pogleda i mnoge druge.

Budući da stvaramo karikaturu lica, potrebno je stvoriti 3D model lica koji se isctava na licu korisnika u realnom vremenu. Model je jednostruka teksturirana 3D mreža trokuta. Osnovna ideja izrade modela je detekcija točaka lica te sastavljanje iz njih mreže trokuta. Zatim je potrebno na svaki trokut iscrtati zadanu teksturu. Time se dobiva model u funkciji maske koja točno rekreita dijelove lica sa slike.

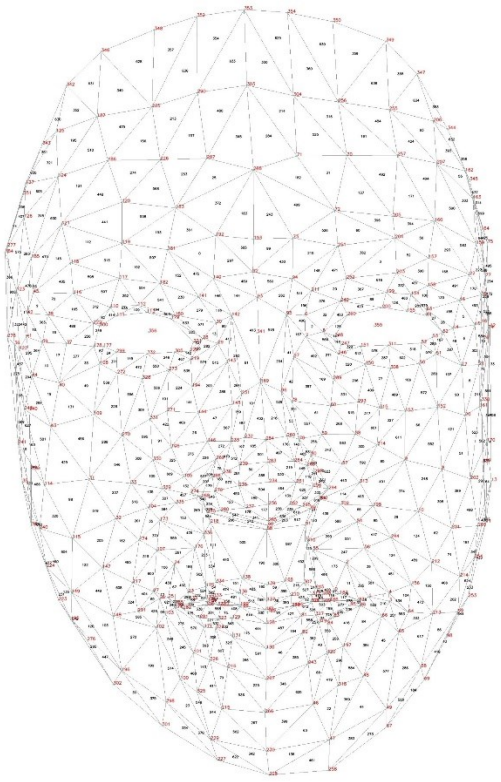


Slika 3.2 Ugrađeni 3D model lica

VisageTracker ima već ugrađeni 3D model lica koji se može korisititi, te je prikazan na slici (Slika 3.2). 3D model lica sastoji se od 3 komponente: popisa koordinata vrhova modela, liste trokuta modela i popis koordinata teksture. Zadana konfiguracija koristi *jk_300 model* koji se sastoji od 357 vrhova i 640 trokuta. Svim komponentama se može pristupiti kroz `FaceData` strukturu koja sadrži metode za njihovo dobivanje. `FaceData ::`

`faceModelVertexCount()` je metoda koja vraća broj vrhova te `FaceData :: faceModelVertices()` vraća popis koordinata tih vrhova. Na isti način se koriste i metode `FaceData :: faceModelTriangleCount()` i `FaceData :: faceModelTriangles()` koje vraćaju broj i popis svih trokuta. Te zadnje metoda `FaceData :: faceModelTextureCoords()` koja daje koordinate teksture za 3D model lica.

Točni izgled *jk_300 model* koji sadrži 357 vrhova i 640 trokuta je prikazan na slici (Slika 3.3).



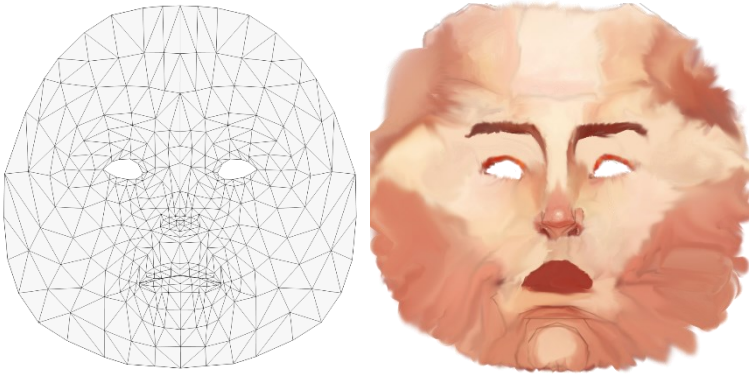
Slika 3.3 Vrhovi modela *jk_300*

Kao ishodište modela, uzima se točka 93 koja se nalazi između očiju.

Kako bi teksturirana mreža točno prekrivala lice korisnika, potrebno je pratiti i poziciju glave, odnosno translataciju i rotaciju glave. Ovi podaci dobivaju se pozivom metoda `VisageTracker :: getHeadTranslation()` i `VisageTracker :: getHeadRotation()`. Za svaku sliku, odnosno kadar, ažurira se položaj mreže tako da se pomak translatacije i rotacije pribroji početnoj poziciji mreže.

3.3. Izrada maske karikature

U sklopu VisageTrackerUnity Demo projekta, nalazi se predložak za izradu tekstura. Predložak je potrebno učitati u neki program za obradu slika, npr. Photoshop, te preko njega naslikati željeni uzorak. Uređeni predložak se spremi kao nova slika te nastaje nova tekstura. Za korištenje teksture u Unity3D-u, potrebno je učitati željenu teksturu i primijeniti je na određen materijal. U ovom projektu, korišten je materijal uvezen iz demo projekta naziva *Head*, uz shader *Unlit/Teksture* gdje je kao baza korištena teksura sa slike (Slika 3.4) izrađena u Photoshopu. Kad zadana tekstura pravilno prekriva mrežu trokuta modela, rezultat je maska koja prekriva lice.



Slika 3.4 Predložak za izradu tekstura i izrađena tekstura

Sljedeći korak u stvaranju maske karikature, je proces izobličenja 3D modela lica. Postupak započinje učitavanjem koordinata vrhova točaka lica zadanog modela savršenog lica. Dobivene točke kao i sve ostale točke koje prestavljaju točke lice spremljene su u strukturu `Vector3[]` zbog lakšeg rukovanja podacima. Zatim se u drugu sturkturu spremaju koordinate vrhova lica u trenutnom kadru, koji ćemo zvati trenutni model.

Glavni korak u procesu je izračun razlika koordinata vrhova modela savršenog lica i trenutnog lica. Razlika se računa za svaku koordinatu mreže modela, odnosno za svaku os svake koordinate. Pojednostavljeno, za jedan vrh modela, razlika na osi x je razlika u vrijednosti x-koordinate između pozicije tog vrha na savršenom modelu i pozicije tog vrha na trenutnom modelu.

Zatim slijedi proces izobličavanja trenutnog modela, tako da svaki vrh trenutnog modela dodatno pomaknemo u prostoru po svakoj osi za dobivenu razliku pomnoženu

skalarom naziva *empCar* koji je skraćenica od emphasis of caricature, u prijevodu isticanje karikature, što označava skalara naglašavanja karikature.

```
for (int j = 0; j < VertexNumber; j++) {
      float diffX = perfectVertices[j].x - Vertices[faceIndex][j].x;
      float diffY = perfectVertices [j].y - Vertices[faceIndex][j].y;
      float diffZ = perfectVertices [j].z - Vertices[faceIndex][j].z;
                if (diffX >= 0) {
                    resultVertices[j].x -= diffX * empCar;
                }else{
                    resultVertices[j].x += -diffX * empCar;
                }if (diffY >= 0) {
                    resultVertices[j].y -= diffY * empCar;
                }else{
                    resultVertices[j].y += -diffY * empCar;
                }if (diffZ >= 0) {
                    resultVertices[j].z -= diffZ * empCar;
                }else{
                    resultVertices[j].z += -diffZ * empCar;
                }
      }
```

Kôd 3.1 – Metoda za pomak vrhova modela

Kao što je vidljivo u kodu (Kôd 3.1) gdje `perfectVertices` označava vrhove modela savršenog lica, `Vertices[faceIndex]` označava vrhove trenutnog modela, `resultVertices` označava vrhove rezultante mreže i `VertexNumber` broj vrhova, razlika koordinata vrhova savršenog modela i trenutnog modela se računa za svaki vrh te mreže. Ako je razlika na x osi veća od nule, to znači da je vrh trenutnog modela bliže x osi nego vrh savršenog modela, stoga će se x koordinata tog vrha smanjiti za razliku pomoženu skalarom, odnosno vrh će se približiti osi x. Suprotno, ako je razlika manja od nule, vrh će se udaljiti od osi x. Na ovaj način, naglašavajući razlike između savršenog i trenutnog modela, na korak smo bliže stvaranju karikature.

Kad je skalar *empCar* = 0, nema deformacija i rezultat je trenutni model izgledom isti kao i lice korisnika, ali prekriven maskom. Povećavanjem skalara vidljive su promjene na modelu te se generira karikatura. Kad je skalar veće vrijednosti, npr. *empCar* > 2, vrhovi mreže se počinju presijecati i model više ne poprima izgled lica. U aplikaciji je zato *empCar* određen na vrijednost *empCar*=1.4.

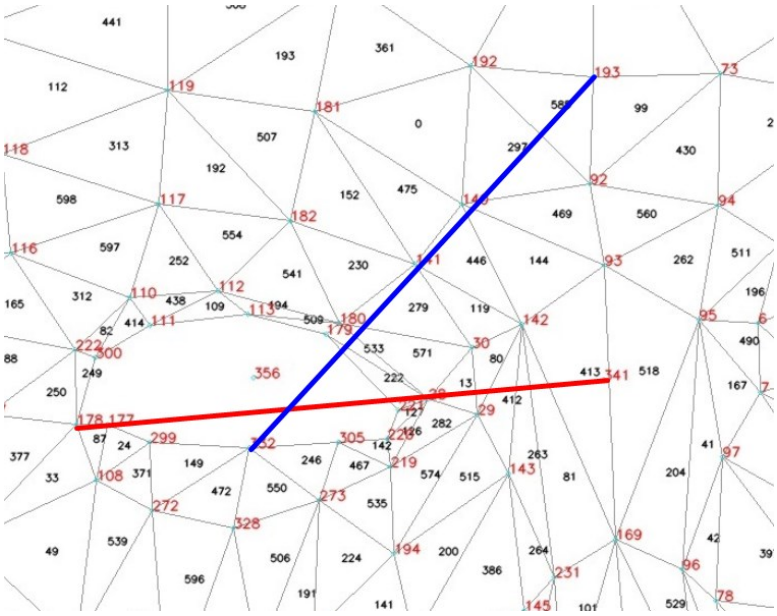
3.4. Razmještaj osnovnih dijelova lica

Kao što je ranije spomenuto, da bi karikatura bila što uvjerljivija, nije dovoljno samo naglasiti crte lica, potrebno je obratiti pozornost i na razmještaj osnovnih dijelova lica, poput pozicije očiju, nosa i usana. To relativno pomicanje obilježja sačinjava drugi korak u ovom sustavu izrada karikature. Ono omogućuje stvaranje ekstremnije karikature uz minimaliziranje skalara naglašavanja karikature i stoga sprječava stvaranje mreže koja je previše deformirana.

Pozicija osnovnih dijelova lica se računa na sličan način kao i u prošlom koraku, gdje se računa razlika zadanih udaljenosti između savršenog i trenutnog modela te se ovisi o dobivenoj razlici pomiču dijelovi lica.

3.4.1. Oči

Za određivanje pozicije očiju potrebno je odrediti pomak svakog oka po x i y osi.



Slika 3.5 Prikaz vrhova lijevog oka

Lijevo oko određuje skup vrhova: 28, 108, 110, 111, 112, 113, 177, 178, 179, 180, 219, 220, 221, 222, 272, 273, 299, 300, 305, 328 i 332. Translatacijom ovih vrhova pojedinačno postižemo translataciju skupa točaka koji definiraju lijevo oko. Prvi korak je translatacija po x osi. Za nju je presudno odrediti udaljenost vanjskog ruba oka, označeno na slici (Slika 3.5) vrhom 178 i središnje točke između očiju koja se nalazi na x osi, označene vrhom 341. Na slici (Slika 3.5) je vidljiva ta udaljenost označena crvenom linijom. Zadanu

udaljenost računamo za savršeni i trenutni model. Ako je zadana udaljenost dobivena na trenutnom modelu manja od one dobivene na savršenom modelu, znači da je lijevo oko korisnika bliže središnjoj točki nego na savršenom modelu. Posljedično, kako bi naglasili tu razliku lijevo oko potrebno je pomaknuti bliže središnjoj točki, odnosno x osi. Oko translatiramo prema x osi tako da x vrijednost svakog vrha koji određuje lijevo oko uvećamo za apsolutnu vrijednost razlike udaljenosti ta savršeni i trenutni model. Naravno, kako bi mogli naglasiti taj pomak, potrebno je razliku pomnožiti skalarom, koji je ovdje označen kao *empEye.*

```
if (distanceLeftEye_RealX < distanceLeftEye_MeanX){

      resultVertices[pos].x += (distanceLeftEye_MeanX -
      distanceLeftEye_RealX) * empEye;

}
```

Suprotno, ako je zadana udaljenost dobivena na trenutnom modelu veća od one dobivene na savršenom modelu, lijevo oko je potrebno odmkanuti od x osi na isti način.

Za translataciju po y osi potrebno je izračunati udaljenost između sredine donjeg kapka, označenog vrhom 332 i središnje točke modela koja je označena točkom 193. Ova udaljenost je na slici označena plavom linijom. Na isti način lijevo oko translatiramo po y osi za apsolutnu vrijednost razlike udaljenosti dobivenih iz trenutnog i savršenog modela, ali skalar *empEye* je podijeljen s 5 kako bo translatacija bila što uvjerljivija.

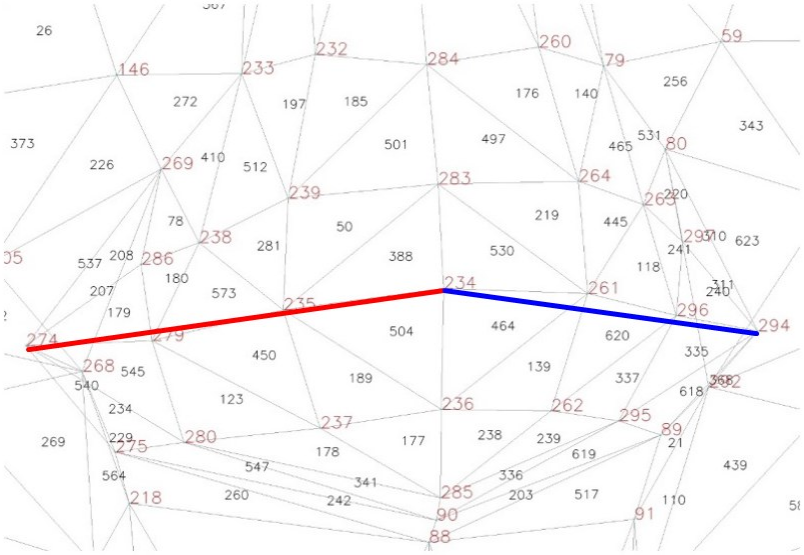
Desno oko je određeno skupom vrhova: 5, 50, 51, 52, 53, 61, 62, 63, 150, 151, 154, 155, 199, 200, 246, 247, 248, 249, 250, 308 i 311. Prilikom translatacije po x osi računa se udaljenost između vrhova 62 i 341, te prilikom translatacije po y osi između vrhova 311 i 193. Na potpuno identičan način već opisan postižemo translataciju i desnog oka.

Skalar *empEye* je skraćenica od emphasis of eye i označava naglašavanje pozicije očiju. Proizvoljno je postavljen tako da iznosi *empEye*=5.

3.4.2. Nos

Prilikom karikiranja izgleda nosa, u ovom projektu njegov izgled određuju duljina i širina nosa. Duljina je definirana kao pozicija nosa na y osi, dok je širina definirana kao udaljenost nosnica od x osi.

Pomak u širinu implementiran je pojedinačno za lijevu i desnu nosnicu. Na slici je prikazan skup točaka vrhova koji određuju nos. Linija točaka od vrha 284 do vrha 88 predstavlja središte nosa i dijeli nos na lijevu i desnu stranu. Širina nosa, odnosno udaljenost kraja nosnice do središta je određena za lijevu stranu kao udaljenost između vrhova 274 i 234, na slici (Slika 3.6) označena crvenom bojom te za desnu stranu kao udaljenost između vrha 294 i 234, na slici (Slika 3.6) označena plavom bojom. Vrh 234 je označen kao središte nosa.

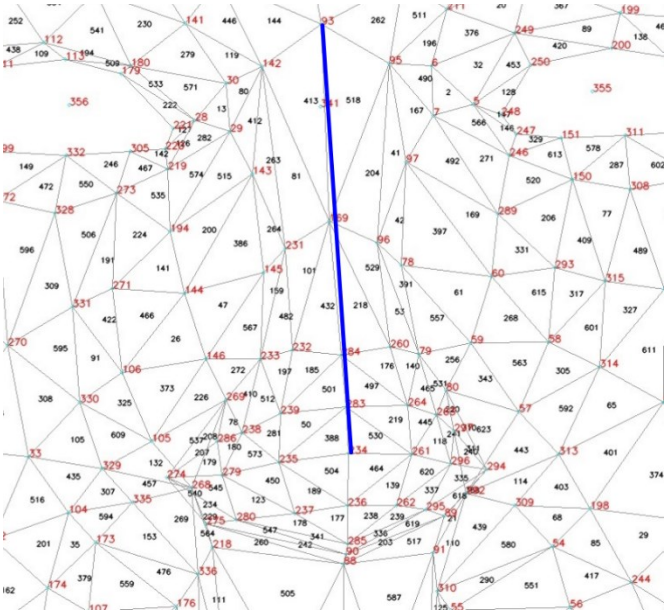


Slika 3.6 Prikaz vrhova širine nosa

Pomak točaka po x osi, provodi se istim postupkom kao i prije navedenom. Potrebno je usporediti zadane duljine za lijevu i desnu stranu nosa za oba modela, savršeni i trenutni model. Zatim se x koordinate vrhova koji sačinjavaju lijevu i desnu stranu nosa, ovisi s koje su strane središta, udaljavaju ili približavaju x koordinati središta. Pomak je opet određen kao razlika zadanih udaljenosti dobivenih iz oba modela, pomnoženi skalarom *empNoseWidth.* Ovaj skalar je u projektu proizvoljno postavljen na *empNoseWidth*=6 da daje zadovoljavajuće karikirane rezultate.

Duljina nosa ustvari predstavlja poziciju vrhova koji sačinjavaju nos, koji su vidljivi na slici, na osi y. Translatacija se dakle vrši po y osi, a uvjetovana je s udaljenosti od središta nosa do središta između očiju koja je na slici (Slika 3.7) označena plavom bojom. Točnije, potrebno je izračunati udaljenost između vrhova 234 i 93. Te zatim oviseći o tome koja od udaljenosti dobivenih na dva modela je veća, sve vrhove nosa translatirati gore ili dolje po

y osi. Iznos translatacije je opet određen kao razlika dvaju dobivenih udaljenosti pomnožen skalarom *empNoseLenght* koji ovdje iznosi *empNoseWidth*=1.5.

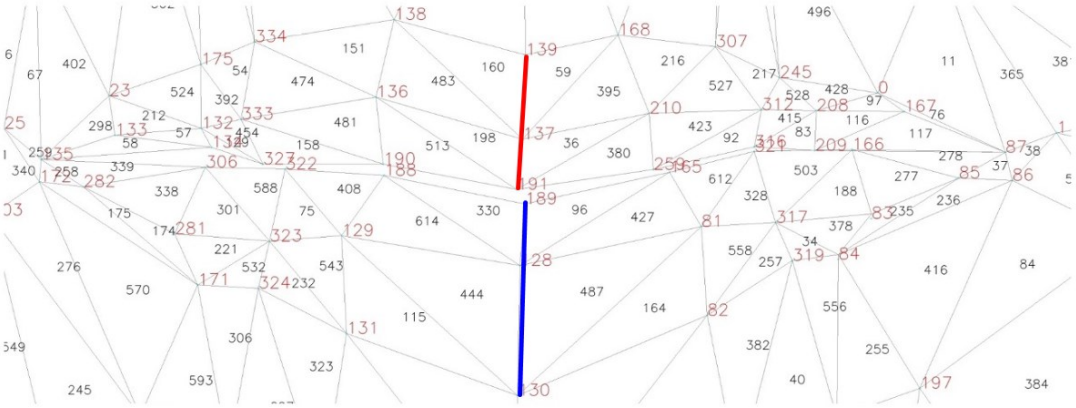


Slika 3.7 Prikaz vrhova dužine nosa

3.4.3. Usta

Zadnja stavka koja se dodatno translatira su usne, odnosno rub gornje i donje usna posebno te se tako naglašava debljina usana. Točke vrhova koji sačinjavaju usne, odnosno vrhovi koji definiraju rub usana, translatiraju se po y osi.

Rub gornje usne određen je vrhovima: 23, 175, 334, 138, 139, 168, 307, 245, 0 i 167. Udaljenost za koju će se ti vrhovi pomicati je zadana kao udaljenost između vrhova 139 i 191 (Slika 3.8). Istim postupkom se izračunava razlika udaljenosti na oba modela te se vrhovi ruba gornje usne translatiraju.



Slika 3.8 Prikaz vrhova usana

Rub donje usne je definiran vrhovima: 172, 171, 324, 131, 130, 82, 319, 84 i 86. Debljina donje usne je određena kao udaljenost između vrhova 189 i 130 (Slika 3.8). Razlika ove udaljenosti dobivene iz oba modela označava pomak vrhova koji određuju donju usnu po y osi.

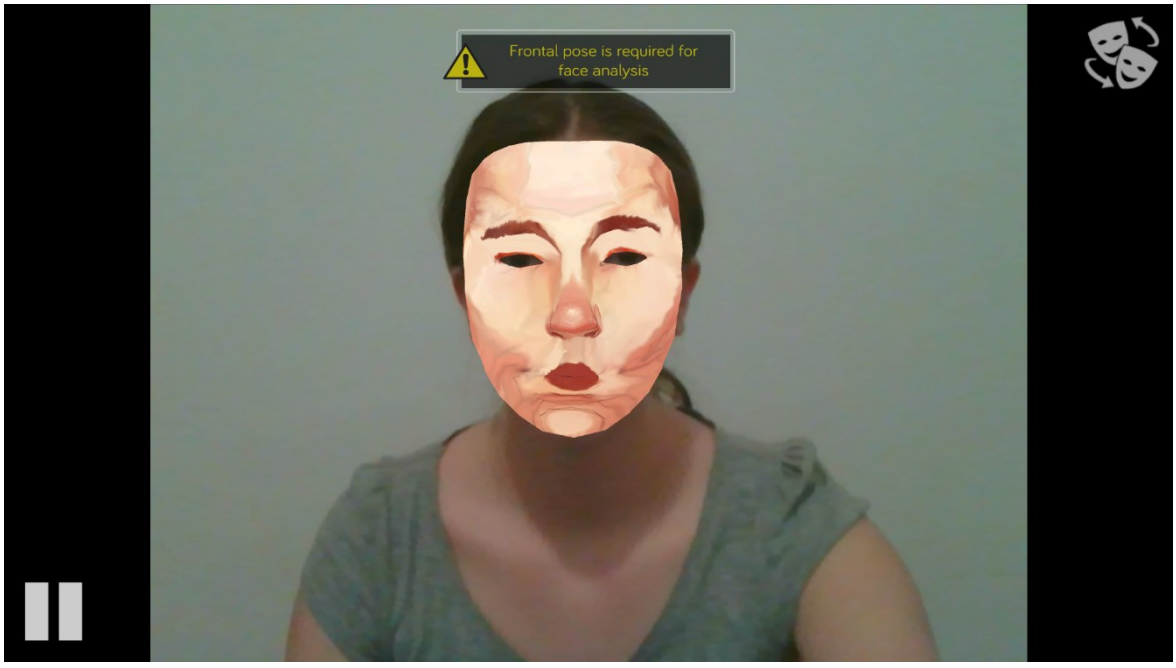
Za oba slučaja, pomak koji je izražen kao razlika zadanih udaljenosti dobivenih iz savršenog i trenutnog modela pomnožen je skalarom *empLip* koji proizvoljno iznosi 3.

4. Aplikacija

Rezultat rada je aplikacija za desktop stvorena u Unity3D razvojnom okruženju. Aplikacija omogućuje korisniku uporabu dvije funkcionalnosti, izrada karikatura i praćenje lica. Sučelje je stvoreno kao Unity scena te se sastoji od različitih komponenti od kojih su neke model lica, kamera, video kamera i ravnina za prikaz.

Praćenje lica se pokreće automatski pokretanjem aplikacije, a može se i zaustaviti pritiskom na tipku u donjem lijevom kutu, gdje se pauzira snimanje kamere.

Prikaz karikature se može podešavati tipkom u gornjem desnom kutu. Početno se u aplikaciji vidi samo slika korisnika bez ikakvih dodataka. Jednim pritiskom na zadanu tipku, preko lica korisnika se pojavljuje maska sa određenom teksturom (Slika 4.1). Ova maska nije izobličena i vrhovi njene mreže se poklapaju s vrhovima očitanog lica. Dakle, maska savršeno prekriva lice i prati njegove konture. Implementacijski ovo stanje ustvari sve skalare naglašavanja postavlja na nulu te ne dolazi do deformiranja početnog modela.



Slika 4.1 Prikaz teksturirane maske

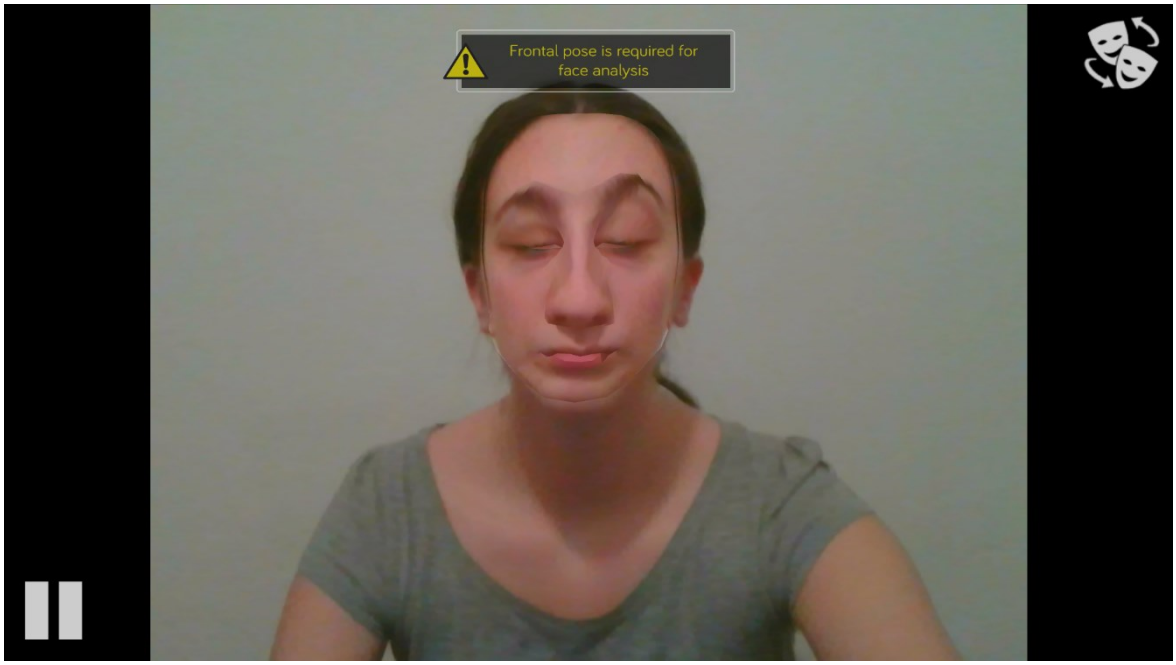
Još jednim pritiskom na taj gumb, preko lica se prikazuje maska karikature (Slika 4.2). Ona je sačinjena od modela kojem je promijenjena mreža vrhova postupkom

opisanim u prijašnjem poglavlju. Tekstura modela ostaje nepromijenjena. Time se dobiva maska karikature koja je ustvari izobličen 3D model lica. Implementacijski u ovom stanju su skalari naglašavanja postavljeni na proizvoljne vrijednosti koje su već navedene.

Slika 4.2 Prikaz karikirane maske

Trećim pritiskom na gumb se mijenja tekstura maske iz slikovnog primjera u teksturu dobivenu putem video kamere. Na ovaj način se zapravo dobiva realistična slika karikature osobe na kojoj su vidljivi detalji lica i način na koje je lice izobličeno (Slika 4.3). U ovom slučaju tekstura je slika koju vidi video kamera i bilo je potrebno promijeniti točke teksture u mreži modela. Točke teksture dobivene su pozivom metode `VisageTracker :: getFaceModelTextureCoords()` koja vraća par koordinata u, v za svaki vrh.

Aplikacija podržava stvaranje maske karikature za najviše dvije osobe istovremeno.



Slika 4.3 Prikaz karikature lica

5. Problemi i potencijalna poboljšanja

Prvi problem koji se javlja kod stvaranja karikature je taj što je početni model savršenog lica prikladniji za žene, odnosno predstavlja žensko lice. Zbog različite fizionomije muškog lica, rezultat će biti veća naglašenost crta lice koje se ne podudaraju s početnim modelom te karikatura može biti prenaglašena. Poboljšanje bi bilo uvođenje detektiranje spola osobe koja koristi aplikaciju i zatim korištenje ispravnog početnog modela.

Također, da bi rezultati bili što realističniji, bilo bi poželjno umjesto modela savršenog lica koristi model koji prikazuje prosječno lice. Za izradu modela prosječnog lica prvo je potrebna velika baza podataka snimaka ljudskog lica, posebno muških i posebno ženskih. Zatim bi se nakon skeniranja tih lica izračunala aritmetička sredina pojedinih karakterističnih točaka lica i stvorio model koji bi zaista prikazivao prosjčeno lice. Problem u ovom pristupu je što je teško nabaviti dovoljno veliku bazu podataka koja će sadržavati slike osoba različite dobi i čak podrijetla.

Veliki nedostatak ove aplikacije je taj što se karikira maska koja se iscrtava preko lica korisnika, a ne samo lice korisnika. Poboljšanje se može ostvariti tako da izradu karikature izvodimo samo na slici, a ne u realnom vremenu gdje bi direktno manipuirali pikselima. Tada bi rezultati bili precizniji i sami detalji karikature bi bili mnogo uočljiviji.

Između ostalog, zbog grešaka u očitavanju točaka lica neki vrhovi maske mogu biti krivo pozicionirani i krivo naglašeni. Do ovog problema dolazi zbog loših uvjeta kod korištenja aplikacije, poput mraka ili prebrzog pomicanja lica prilikom snimanja. Greške su očite i prilikom rotacije glave, gdje sustav ne može pratiti točke lice ako je rotacija glave prevelika. Optimalni uvjeti bi bili korištenje aplikacije uz jako frontalno svijetlo kako bi lice bilo što vidljivije moguće i frontalni položaj glave kako bi očitanja bila što preciznija.

U budućoj nadogradnji bilo bi poželjno implementirati sustav koji je u mogućnosti skenirati i karikirati više od samog lica osobe. Primjerice, uz dodatak skeniranja ušiju, vrata ili čak i cijele osobe.

Zaključak

Ovaj rad predstavlja sustav za automatsko generiranje karikature lica. Sustav je implementiran kao Unity3D aplikacija koja se pokreće na stolnom računalu. Iako neki tehnički aspekti ovog projekta proizlaze iz rada drugih, fokus ovog rada je bilo iscrtavanje karikature u stvarnom vremenu.

Analizirajući metode detaljno opisane od strane nekoliko umjetnika karikaturista, utvrđeno je da, uz očite vještine crtanja, većina koristi niz trikova i tehnika za izradu karikature. Ovaj sustav zasniva se na pristupu koji slijede i crtači, a to je naglašavanje karakterističnih crta lica osobe.

Koncept karikature u osnovi je pretjerivanje i naglašavanje točaka lica koje odudaraju od početnog modela. Kao početni model korišten je model savršenog ženskog lica preuzet iz već provedene studije. Ključno načelo na kojem se bazira rad je izračun razlike točaka lica korisnika aplikacije i početnog modela. Iako se na ovaj način može dobiti poprilično dobar prikaz karikature, potrebno je obratiti pozornost i na relativan položaj osnovnih dijelova lica. Sličnim algoritmom izračunati su potrebni pomaci očiju, nosa i usta.

Stvorena aplikacija u realnom vremenu preko lica korisnika iscrtava masku koja predstavlja 3D model lica. Prikaz maske je omogućen s tri različita prikaza od kojih svaki predstavlja jednu fazu izrade karikature. U prvom prikazu se preko lica korisnika iscrtava maska koja prati crte lice. Slikovna tekstura koja se koristi točno prati položaj glave i točke lica. Ovaj prikaz služi kao pokazatelj rada funkcija metoda za praćenje lica i normalni prikaz maske. Druga faza prikazuje masku koja je deformirana na način da čini karikaturu. Zbog slikovne teksture točno su vidljive zahvaćene promjene na maski. Rezultat trećeg prikaza je cilj ovog rada, koja prikazuje točnu karikaturu osobe. Tekstura je ulaz iz video kamere i iscrtavanjem preko izoblićenog modela stvara se realistični prikaz karikature.

Literatura

[1] Wikipedia.org. (2006). Karikatura. Poveznica: https://hr.wikipedia.org/wiki/Karikatura; pristupljeno 30. svibnja 2021.

[2] Wikipedia.org (2008). Adobe Photoshop. Poveznica: https://hr.wikipedia.org/wiki/Adobe_Photoshop; pristupljeno 30. svibnja 2021.

[3] Visagetechnologies.com. (2021). Windows: FaceTracker2 - Visage Technologies Documentation - visage|SDK Documentation. Poveznica: https://docs.visagetechnologies.com/display/vtd/Windows%3A+FaceTracker2; pristupljeno 30. svibnja 2021.

[4] Visagetechnologies.com. (2021). Windows: VisageTrackerUnityDemo - Visage Technologies Documentation - visage|SDK Documentation. Poveznica: https://docs.visagetechnologies.com/display/vtd/Windows%3A+VisageTrackerUnityDemo; pristupljeno 30. svibnja 2021.

[5] Marquardt Beauty Analysis. (2020). Phi the Key to Beauty - Marquardt Beauty Analysis. Poveznica: https://www.beautyanalysis.com/research/our-research/phi-key-beauty/; pristupljeno 5. lipnja 2021.

[6] Mihalicdictionary.org. (2020). Karikatura. Poveznica: https://hr.mihalicdictionary.org/wiki/Caricature; pristupljeno 8. Lipnja 2021.

[7] Dulibić, Frano (2005a) Definiranje karikature kao likovne vrste. U: Brnčić, Jadranka (ur) Karikatura: zagrebački pojmovnik kulture 20. stoljeća: zbornik znanstvenih radova (str. 11- 33). Zagreb: Filozofski fakultet.

[8] Brnčić, Jadranka (2005) Od ikone do karikature. U: Brnčić, Jadranka (ur) Karikatura: zagrebački pojmovnik kulture 20. stoljeća: zbornik znanstvenih radova (str. 263- 84). Zagreb: Filozofski fakultet.

[9] Liang, Lin & Chen, Hong & Xu, Ying-Qing & Shum, Heung-Yeung. (2002). Example-based caricature generation with exaggeration. 386 - 393. 10.1109/PCCGA.2002.1167882.

[10] Brennan, S.E. (1985) Caricature generator: The dynamic exaggeration of faces by computer. Leonardo, 18, 170-178. doi:10.2307/1578048

Sažetak

SUSTAV ZA AUTOMATSKO STVARANJE KARIKATURA

Cilj rada je izrada automatskog sustava za stvaranje karikatura. Sustav je implementiran kao Unity aplikacija koja prikazuje lice korisnika preko kojeg se iscrtava karikatura u realnom vremenu. Karikatura je stvorena pomicanjem vrhova mreže koja definira model lica i prekrivena je slikovnom teksturom ili teksturom slike dobivenom iz video kamere. U sklopu rada su prikazane metode za stvaranje karikature i korištene funkcije koje služe za praćenje položaja lica.

Ključne riječi: karikatura, automatsko stvaranje karikature, analiza lica, Unity, Visage|SDK

Summary

AUTOMATIC CARICATURE GENERATION

The aim of this paper is to create an automatic system for creating caricatures. The system is implemented as a Unity application that displays the user's face over which a caricature is drawn in real time. The cartoon is created by moving the vertices of the grid that defines the face model and is covered with an image texture or image texture obtained from a video camera. The paper presents methods for creating a caricature and the functions used to monitor the position of the face.

Keywords: caricature, automatic caricature creation, face analysis, Unity, Visage|SDK