SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 701

ANIMACIJA POKRETANA PRAĆENJEM LICA NA POKRETNIM UREĐAJIMA

Mihael Grilec

Zagreb, lipanj 2014

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 9. ožujka 2014.

DIPLOMSKI ZADATAK br. 701

Pristupnik: **Mihael Grilec (0036428206)**
Studij: Računarstvo
Profil: Programsko inženjerstvo i informacijski sustavi

Zadatak: **Animacija pokretana praćenjem lica na pokretnim uređajima**

Opis zadatka:

Praćenje lica je postupak kojim se tehnikama računalnog vida slijedi ljudsko lice i njegove ključne točke u video slici. Ova tehnologija može se koristiti za pokretanje animacije lica virtualnog lika, što je poznato i pod engleskim nazivom performance animation. Potencijalne primjene su u području igara, zabave i komunikacija a posebno su zanimljive primjene na pokretnim uređajima.
Na Zavodu za telekomunikacije dostupan je sustav za praćenje lica u stvarnom vremenu. Potrebno je odabrati odgovarajući grafički sustav (preporučen je Unity 3D) te ga na odgovarajući način povezati sa sustavom praćenja lica.
Vaša je zadaća proučiti postojeće sustave, te predložiti i implementirati izvedbu sustava za animaciju pokretanu praćenjem lica na pokretnim uređajima.
Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 30. lipnja 2014.

Mentor:

Prof.dr.sc. Igor Sunday Pandžić

Djelovođa:

Doc.dr.sc. Igor Mekterović

Predsjednik odbora za diplomski rad profila:

Prof.dr.sc. Krešimir Fertalj

Sadržaj

1. Uvod

Moderni sustavi za praćenje lica mogu mjeriti puno više parametara od jednostavne pozicije i rotacije glave počevši od izraza lica, pomaka obrva, usta i očiju do ocjene starosti i rase korisnika. Koriste se kao ulazni uređaji za kontrolu igara, kao pomoć pri korištenju računalom, u industriji oglašavanja, u zabavnim aplikacijama i kao sustavi za hvatanje pokreta lica za animiranje govora. Razvojem mobilnih uređaja i rastom mobilnog tržišta su dobili novu perspektivu za aplikacije na pokretnim uređajima.

U sklopu ovog diplomskog rada izrađen je virtualni lik sa pripadnim animacijama koje kombinirano mogu prikazati razne izraze lica. Korištenjem modernog pokretačkog sustava izrađena je aplikacija u koju je integriran sustav za praćenje lica i koja animira virtualni lik koristeći se dobivenim parametrima pomaka dijelova lica koje sustav prati. Aplikacija je višeplatformska, izvodi se na pokretnim uređajima i lagana je za nadogradnju i konfiguraciju bez dodatnog programiranja.

Uz kratki uvod o mobilnim uređajima, platformama i načinu izrade aplikacija za svaku platformu i uvoda u računalnu animaciju koristeći kosti i mješanje oblika, u četvrtom poglavlju je dan detaljan pogled na različite aspekte izrade aplikacije uključujući izradu virtualnog lika i izrade animacija, integracije sustava za praćenje lica u pokretački sustav, sustava za konfiguriranje i animaciju i aspekte izrade višeplatformske aplikacije. Uz rad je priložena i sama aplikacija kao i njen izvorni kôd.

2. Praćenje lica na pokretnim uređajima

Praćenje lica u stvarnom vremenu je, do pojave pametnih telefona sa snažnijim procesorima, bilo ograničeno na izvođenje na jačim stolnim računalima i laptopima. Razvojem tržišta i pojavom Apple iPhone i Google Android uređaja sa prednjom kamerom i jakim procesorom otvorila se mogućnost praćenja lica na novim platformama – pokretnim uređajima.

Najčešće primjene praćenja lica na pokretnim uređajima su bile aplikacije proširene stvarnosti koje su nad slikom korisnika dobivenoj sa prednje kamere uređaja prikazivale estetske dodatke kao što su naočale, šminku ili različite tipove frizure u stvarnom vremenu. Prve takve aplikacije su mogle prikazivati grafiku samo nad statičnim slikama snimljenim na određeni način. Korisnik se morao slikati postavljajući lice u iscrtano polje na ekranu. Aplikacije nisu izračunavale podatke o pozi lica, već su iscrtavala grafiku računajući na ispravan položaj i rotaciju lica u slici. Naprednije aplikacije su izračunavale poziciju i rotaciju lica iz slike i prikazivale grafiku prema dobivenim podacima. Objekti su se iscrtavali translatirani i rotirani na način da odgovaraju na lice korisnika dobivajući stvarniji prikaz. Taj proces je trajao više sekundi po slici, no to su bili prvi algoritmi za detekciju poze lica na pokretnim uređajima što je bitan dio procesa praćenja lica. Porastom procesorske moći uređaja se smanjilo i vrijeme potrebno za detekciju poze lica te je praćenje lica moglo biti izvedeno u stvarnom vremenu.

Detekcija lica danas dolazi kao integrirani servis mobilnih operacijskih sustava, no postoji i velik broj biblioteka za praćenje lica koje nude različite rezultate od prepoznavanja gesti do pojedinih točaka na licu i praćenja očiju i pogleda.

2.1. Mobilni uređaji i platforme

Danas postoje 3 glavne platforme na tržištu mobilnih uređaja; iOS, Android i Windows Phone koje se razlikuju u minimalnim specifikacijama uređaja, tehnologiji programiranja i načinima distribucije aplikacija. Različitost platformi i načina razvoja aplikacija za njih čini pisanje multiplatformskih korisničkih aplikacija gotovo nemogućim. Često se nameće razvoj zasebnog pisanja aplikacija za pojedinu platformu kao jedino rješenje za dostupnost aplikacije na svim uređajima.

2.1.1. iOS

iOS je operacijski sustav za Apple iPhone, iPod, iPad i Apple TV uređaje razvijen u Apple-u. Predstavljen 2007 u prvoj verziji nije dopuštao izradu aplikacija koje bi se pokretale na uređaju, nego su se kao alternativa prikazale web aplikacije. To se promjenilo od iOS 2.0 kada je preplatnicima razvojnog programa pružen paket za razvoj aplikacija (engl. software development kit). Uz otvaranje platforme za razvoj Apple je ponudio i AppStore kao jedini način distribucije aplikacija.

Najprodavaniji uređaji sa iOS operacijskim sustavom su sigurno iPhone uređaji. Trenutno aktualan uređaj je iPhone 5 sa 1.3 GHz 64-bitnim dvojezgrenim procesorom i 1 GB radne memorije što je dovoljno snage za praćenje lica i iscrtavanje kompleksne 3D grafike u stvarnom vremenu.

Za iOS uređaje aplikacije se razvijaju koristeći *Xcode* – Apple-ov alat za razvoj aplikacija koji podržava iOS, OS X i Safari aplikacije. Aplikacije se razvijaju koristeći Objective-C programski jezik koji je superset jezika C proširen objektno orijentiranim načinom pisanja. Objective-C i C++ se mogu koristiti paralelno što olakšava prilagodbu desktop aplikacija pisanih C++ jezikom.

2.1.2. Android

Jedini operacijski sustav za mobilne uređaje otvorenog kôda je razvila tvrtka Android, Inc. iz Kalifornije na bazi Linux-a koju je kupio Google 2005. Lansirajući 2007., većina proizvođača mobilnih uređaja je preuzela Android kao glavni operacijski sustav sa vlastitim nadogradnjama. Android trenutno ima najveći postotak tržišta od svih mobilnih platformi i najviše dostupnih aplikacija.

Android je prisutan na cijeloj paleti uređaja uključujući mobilne telefone, tablete i računala. Bitno različite specifikacije uređaja otežavaju razvoj aplikacija čak ako je razvoj ograničen samo na mobilne telefone. Uređaji se mogu razlikovati u procesorskoj moći (od 600 MHz procesora do osmojezgrenih 2 GHz procesora baziranih na ARM-u i 32-bitnih procesora), prisutnosti prednje kamere, prisutnosti stražnje kamere, prisutnosti fizičke tipkovnice, ekrana osjetljivog na dodir, količini memorije i veličini ekrana.

Za razvoj korisničkih aplikacija se koristi programski jezik Java uz prilagođeno *Eclipse* razvojno sučelje uključeno u Android paket za razvoj aplikacija. Dostupan je i Android paket za razvoj aplikacija baziranih na C++ jeziku koji bi zaobilazili standardni Java virtualni stroj (engl. Java Virtual Machine) u svrhu postizanja boljih performansi. Zahtjevne aplikacije mogu izdvojiti procesorski zahtjevne dijelove koji bi se onda izvršavali bez virtualnog stroja što je ključno za implementacije praćenja lica.

2.1.3. Windows Phone

Microsoftov operacijski sustav je najmlađi od dostupnih velikih platformi za mobilne uređaje i trenutno je aktualna druga verzija sustava; Windows Phone 8.

Windows Phone 8 ima deklarirane minimalne specifikacije uređaja što olakšava razvoj za ovu platformu. Minimalni specifikacije su Qualcomm-ov dvojezgreni Snapdragon S4 sa najmanje 1.2 GHz procesorske snage, 512 MB radne memorije i ekran na dodir što čini sve uređaje na ovoj platformi dovoljno snažnima za praćenje lica.

Za razvoj aplikacija se koristi Microsoft-ovo razvojno sučelje Visual Studio sa dodacima za Windows Phone 8 i jezik C#, Javascript, kao i C++ što znači da postojeće aplikacije i biblioteke pisane za desktop verziju Windows operacijskog sustava rade i na mobilnoj platformi ako koriste ograničen dio Windows programskog sučelja. Visual Studio je najstarije i najpotpunije razvojno sučelje što čini razvoj korisničkih aplikacija za ovu platformu jednostavnije od razvoja za iOS i Android.

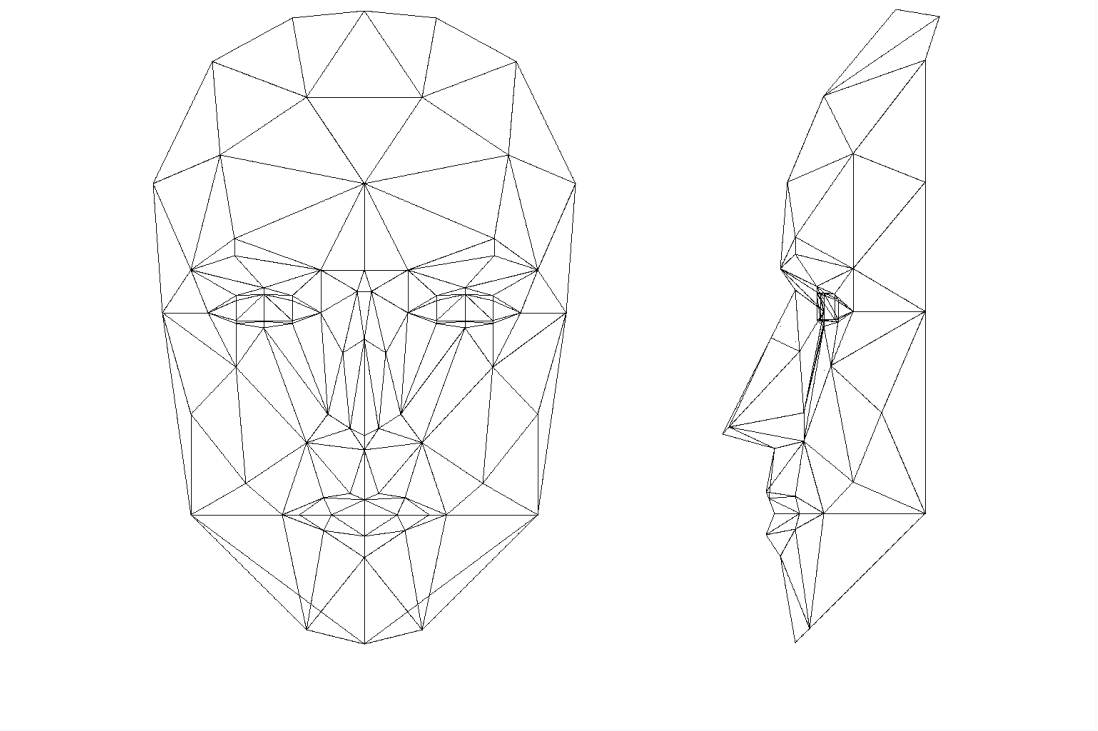
2.2. Praćenje lica

U ovom radu se koristi programska biblioteka za praćenje lica razvijena u *Visage Technologies AB* – visage|SDK. Visage|SDK je dostupan za iOS i Android platforme pa je implementacija aplikacije razvijene u sklopu ovog rada ograničena na iOS i Android uređaje.

Za implementaciju aplikacije se koristi *FaceTracker2* implementacija praćenja lica iz biblioteke koja uz osnovnu translaciju i rotaciju lica pruža i informacije o pomacima dijelova lica kroz akcijske jedinice i informacije o smjeru gledanja. Sustav se konfigurira uz pomoć konfiguracija praćenja u kojima se mogu podesiti parametri bitni za performanse i preciznost praćenja kao i isključiti i uključiti akcijske jedinice koje se prate.

2.2.1. Model lica

Za praćenje se koristi Candide-3 parametrizirana maska napravljena za kodiranje ljudskog lica. Parametrizirana je sa oblikovnim jedinicama (engl. shape unit) i akcijskim jedinicama (engl. action unit). Oblikovne jedinice definiraju statičke promjene modela koja ga čine sličnijim slijeđenom licu kao što su visina glave i dužina nosa. Akcijske jedinice definiraju dinamičke promjene na licu kao što su pomicanje obrva i širenje usana [1].



Slika 1. Linijski prikaz Candide-3 modela[1]

[1] Slika preuzeta sa http://www.icg.isy.liu.se/candide/

2.2.2. Akcijske jedinice

Visage|SDK prati dinamičke promjene na licu po akcijskim jedinicama Candide-3 maske definirane u konfiguraciji korištenoj pri praćenju. Pri animiranju lica akcijske jedinice će se korisiti za upravljanje određenim animacijama uz prethodno filtriranje i mapiranje. Tablica 1. sadrži popis podržanih akcijskih jedinica.

Ime akcijske jedinice	Opis
au_nose_wrinkler	Pomicanje nosnica
au_left_brow_lowerer	Spuštanje lijeve obrve
au_left_outer_brow_raiser	Podizanje vanjskog dijela lijeve obrve
au_left_inner_brow_raiser	Podizanje unutarnjeg dijela lijeve obrve
au_right_brow_lowerer	Spuštanje desne obrve
au_right_outer_brow_raiser	Podizanje vanjskog dijela desne obrve
au_right_inner_brow_raiser	Podizanje unutarnjeg dijela desne obrve
au_lip_stretcher_left	Rastezanje usnice lijevo
au_lip_stretcher_right	Rastezanje usnice desno
au_jaw_z_push	Pomicanje čeljusti prema kameri
au_jaw_x_push	Horizontalno pomicanje čeljusti
au_jaw_drop	Otvaranje čeljusti
au_leye_closed	Zatvaranje lijevog oka
au_reye_closed	Zatvaranje desnog oka
au_rotate_eyes_left	Pomicanje očiju horizontalno
au_rotate_eyes_down	Pomicanje očiju vertikalno

2.2.3. Ograničenja na mobilnim platformama

Za razliku od jačih stolnih i prijenosnih računala, na mobilnim platformama je potrebno optimizirati konfiguraciju praćenja kako bi se dobile bolje performanse. Prva i najbitnija optimizacija je smanjivanje veličine slike koja se koristi za praćenje lica. Visage|SDK sliku sa kamere smanjuje na veličinu zadanu u konfiguraciji i ovo je najbitniji parametar za poboljšanje okvira u sekundi koje sustav obrađuje. Manja veličina radne slike također znači i slabije rezultate praćenja, no sustav dobro reagira na smanjenje slike do neke granice. Veličina radne slike na mobilnim platformama je postavljena na 220 piksela za jače uređaje i 152 piksela za slabije, dok je na jačim platformama postavljena na 380 piksela.

Druga bitna optimizacija je isključivanje akcijskih jedinica koje se ne koriste i koje nemamo namjeru koristiti u animaciji. U implementaciji se ne koriste akcijske jedinice pomicanja čeljusi u x i z osi (horizontalno i prema kameri) i sužavanje i pomicanje samih usana, pa se te akcijske jedinice mogu isključiti u konfiguraciji praćenja.

3. Računalna animacija

Pojam računalna animacija se obično odnosi na animiranje 3D objekata na računalu. Postoji više tehnika animacija pogodnim za različite primjene, no u svrhu ovog rada koristimo tehnike pogodne za animaciju ljudskih likova.

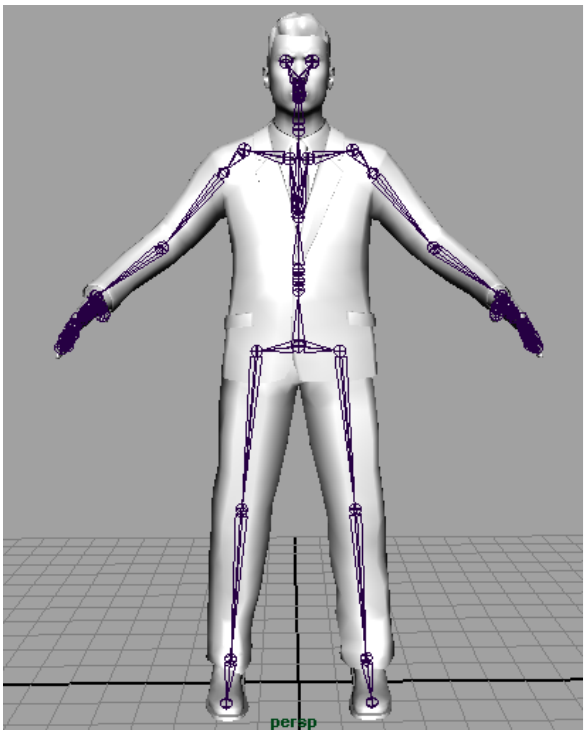
Najjednostavnija vrsta animacije je animiranje pozicije, rotacije i skaliranja modela. Model je definiran pomoću vrhova trokuta i iscrtan na ekranu transformiranjem i projiciranjem trokuta na ekran procesom rasterizacije. Osim što sami vrhovi trokuta imaju vlastitu poziciju definiranu u 3D prostoru u sustavu modela, i sami model može imati vlastitu poziciju, rotaciju i veličinu definiranu 4x4 matricom. Ti parametri se za vrijeme iscrtavanja prenose na vrhove trokuta i mogu se animirati po ključnim okvirima (engl. key frame) u vremenu.

Bitna tehnologija u izradi animacija je praćenje pokreta (engl. motion capture). Tradicionalna metoda praćenja pokreta je uz pomoć sustava kamera i markera pozicioniranih po ključnim dijelovima tijela. Kamere bilježe pozicije markera koje se transformiraju u transformacije kostiju po ključnim okvirima. Dobiveni podaci se u općem slučaju nemogu direktno koristiti zbog trzanja pri praćenju uslijed niske preciznosti i potrebno ih je filtrirati. Druge metode uključuju dubinske kamere i metode računalnog vida. Praćenje lica se može koristiti kao metoda praćenja pokreta za potrebe sinkronizacije usana sa govorom i snimanje izraza lica.

3.1. Animacija kostiju

Animacija kostiju je tehnika animiranja 3D modela napravljena za stvarniju animaciju ljudskih likova kopirajući pokrete kostiju. Danas se koristi kao bitna tehnika u definiranju animacije likova ljudi, životinja ali i mehaničkih sustava.

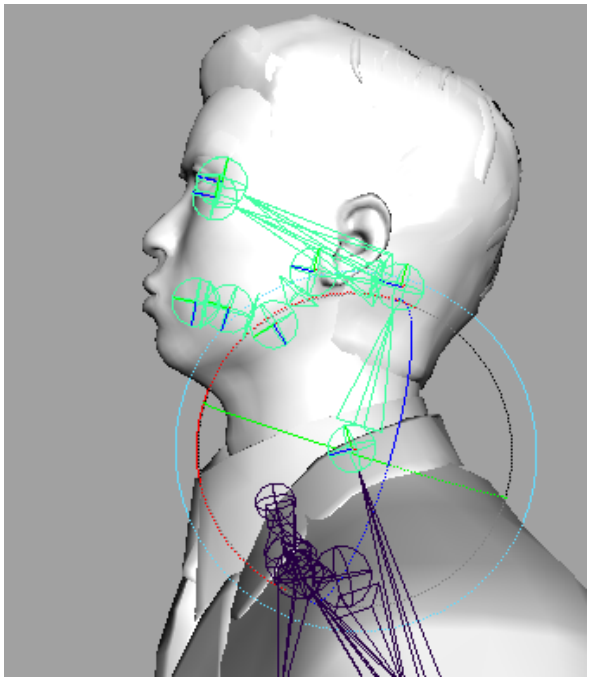
Kosti su dodatni objekti definirani u modelu koji deformiraju vrhove određenih trokuta prema masci tako da prenose vlastitu translaciju, rotaciju i skaliranje na skup vrhova. Kosti također mogu biti poredane u hijerarhiju u kojoj nasljeđuju transformaciju kosti roditelja. Tako možemo preslikati stvarne kosture likova ljudi i životinja koje animiramo. U ovom kontekstu trokuti u modelu su nazvaju koža modela, pa tako imamo definirane kostur i kožu.



Slika 2. Kostur virtualnog lika

Svaka kost u kosturu ima pridružene podatke o roditelju, listi vrhova sa pridruženim faktorom mješenja (engl. blend weight) za svaki vrh i transformacijom same kosti. Tako na jedan vrh može djelovati više kostiju uz različite faktore mješanja. Uz ove podatke

mora se definirati i početna poza modela (engl. bind pose). U najjednostavnijem obliku koristi se set ključnih okvira koje sadrže nove transformacije kostiju za animaciju. Ovo se naziva direktna kinematika (engl. forward kinematics) zbog izravne veze transformacije u ključnom okviru i transformacije kosti. Animacije se prikazuje ovisno o odabranom broju ključnih okvira po sekundi što određuje brzinu animacije. Za svaki okvir i za svaku kost se određuje transformacija same kosti iz hijerarhije kostura. Referenciranjem na listu vrhova i faktora mješanja za svaki vrh, transformiraju se i sami vrhovi množenjem sa matricom transformacije kosti počevši od početne poze kože.



Slika 3. Direktna transformacija kosti vrata

Naprednija tehnika animacije kostiju je pomoću inverzne kinematike (engl. inverse kinematics). Određivanjem željene transformacije kostiju riješavanjem kinematičkih jednadžbi se određuju transformacije prijašnjih kostiju u lancu kako bi se dobila željena transformacija ciljne kosti. Tako se olakšava postupak animacije jer je potrebno samo odrediti završnu transformaciju ciljne kosti bez micanja svih kostiju u lancu. Ako je definiran ljudski kostur u modelu, micanjem jednog prsta na ruci se miču i

kosti nadlaktice, podlaktice i šake kako bi se zadovoljila ciljna transformacija. Ova tehnika zahtjeva postavljanje ograničenja na domet i međusobni odnos kostiju kako bi se spriječile nemoguće pozicije kostura. Ovisno o implementaciji animacije, inverzna kinematike može biti podržana u stvarnom vremenu ili može biti izračunata prije i animirana kao direktna animacija. Ova tehnika je bitni za proceduralnu animaciju kada nije moguće znati u kojem će položaju biti kostur. Primjene ove tehnike uključuju pozicioniranje nogu modela tako da stoje na zemlji u pravoj poziciji i pod pravim kutem i pozicioniranje ruku modela pri animiranju penjanja.

Nekada je potrebno izvoditi dvije odvojene animacije nad istim modelom. To se implementira na način da se za svaku animaciju definira kojim kostima upravlja kako bi se spriječilo mješanje. Tako je moguće izvoditi animaciju trčanja definiranu samo na nogama kostura i zasebnu animaciju koja animira gornji dio kostura modela.

3.2. Animacija lica

Animacija pomoću kostiju je efikasna za upravljanje tijelom ljudskog modela, no lice pokreće velik broj mišića i mali broj kostiju pa ova tehnika animacije nije pogodna za animiranje izraza lica i govora. Za ovakav tip animacije se koristi mješanje oblika (engl. blend shapes).

Ako imamo dva modela – osnovni model (engl. base) i ciljni (engl. target), sa istim brojem vrhova i ako definiramo parove vrhova iz jednog modela u drugi, možemo animirati model interpoliranjem pozicije svih vrhova iz baznog modela u ciljni. Tipično se parovi vrhova definiraju njihovim poretkom u modelima – ciljni model se obično izrađuje deformacijom osnovnog modela pa se poredak vrhova ne mijenja. Može se interpolirati između više ciljnih modela tako da se rezultatne pozicije vrhova izvedu iz interpoliranih nekom funkcijom, obično zbrajanjem ili usrednjavanjem.

Slika 4. Osnovni i ciljni model animacije otvaranja usta

Ova tehnika se koristi za animiranje lica koristeći neutralno lice kao osnovni model i željen izraz lica kao ciljni. Tako je moguće iz neutralnog lica deformacijom modela i pomicanjem vrhova izraditi ciljne modele za svaki od izraza lica koji bi željeli prikazati. Izraze je moguće mješati kako bi se dobio kombinirani izraz – ako imamo ciljne modele smijeha i iznenađenja možemo oblikovati lice da prikazuje izraz između dva definirana.

Ciljni modeli također mogu biti jednostavna pomicanja dijelova lica čijim bi kombiniranjem dobili detaljan izraz cijelog lica. Ako imamo modele sa dignutom lijevom obrvom, dignutom desnom obrvom i otvorenim ustima, njihovim kombiniranjem možemo dobiti izraz lica koji prikazuje iznenađenje. Kod ovakvog načina animiranja sa kombiniranjem, interpolirane pozicije vrhova trokuta je potrebno zbrojiti kako bi se dobile rezultatne pozicije pošto su animacije dijelova lica isključive i usrednjavanje bi rezultiralo smanjivanjem definiranog pomaka u modelu.

Kada kombiniramo animaciju kostima i mješanje oblika bitan je poredak transformacija. Potrebno je prvo pomaknuti vrhove mješanjem oblika, a zatim te vrhove pomaknuti kostima. Kako je standardno da kosti ne miču i vrhove ciljnih modela pri mješanju oblika, obrnutim poretkom i dobili izobličeni model.

3.3. Proceduralna animacija

U aplikacijama interaktivne grafike često nije poznato na koji način je potrebno animirati modele prije pokretanja aplikacije za razliku od učitavanja statičnih podataka kojima se pokreće animacija. Koristeći animaciju kostima i mješanje modela možemo proceduralno postavljati parametre animacije prema ulaznim podacima u stvarnom vremenu u ovisnosti o stanju modela i okoline. Primjeri ovakve animacije su postavljanje nogu ljudskih likova da odgovaraju neravninama na kojima model stoji i hvatanje raznih predmeta u okolini modela.

Animacija kostiju se koristi u mnogo primjena u interaktivnim aplikacijama, u kojima je posebno bitna tehnika inverzne kinematike. Ovdje ćemo nabrojati samo neke česte primjene koje koriste direktnu i inverznu kinematiku.

- Okretanje glave prema važnim objektima u okolini
- Praćenje ostalih likova u okolini pogledom (animacija očiju)
- Stajanje na neravnom terenu
- Hvatanje za predmete u okolini
- Proceduralni hod

U implementaciji koristimo animaciju kostiju kako bi animirali translaciju i rotaciju glave i nadopunili je sa realističnim kretnjama gornjeg dijela torza animiranog lika.

Kosti možemo proceduralno animirati koristeći simulaciju fizike krutih tijela. Dijelovi modela postaju modeli u simulaciji i iz njihovih međusobnih pozicija i rotacija dobivenih iz simulacije se određuju pozicije kostiju. Ova metoda se obično koristi za simulaciju pada tijela kada se u trenutku kada tijelo treba izgubiti kontrolu i pasti uključi simulacija fizike koja nadalje upravlja tijelom.

Pri mješanju oblika možemo kontrolirati parametre interpolacije u ciljne modele koristeći ulazne podatke u stvarnom vremenu i okolinu modela u aplikaciji. Tako u implementaciji koristimo ulazne podatke dobivene praćenjem lica (prvenstveno akcijske jedinice) za mijenjenje parametara animacije mješanjem oblika.

4. Implementacija

U sklopu ovog rada je napravljena aplikacija koja demonstrira animiranje modela ljudskog lika koristeći ulazne podatke iz sustava za praćenje lica. Aplikacija prikazuje ljudski lik nazvan *Kenchi* koji svojim izrazima lica i pokretima glave prati korisnikove izraze i pokrete.

Aplikacija je izrađena na pokretačkom sustavu Unity i može se pokrenuti na iOS, Android i Windows operacijskim sustavima, no za potrebe ovog rada opisana će biti samo iOS i Android verzija. Unity je višeplatformski sustav koji omogućuje gradnju aplikacija za Windows, Mac, Linux, iOS, Android, Windows Phone, Windows Store i Blackberry uređaje.

Za praćenje lica je korišten *visage|SDK* programski paket koji uz pomoć web kamere spojene na računalo može detektirati ljudsko lice na slici i dobiti podatke vezane uz poziciju i rotaciju cijele glave, pomicanju dijelova lica i smjeru gledanja. Primarno su korišteni podaci o pomicanju dijelova lica prevedeni u akcijske jedinice.

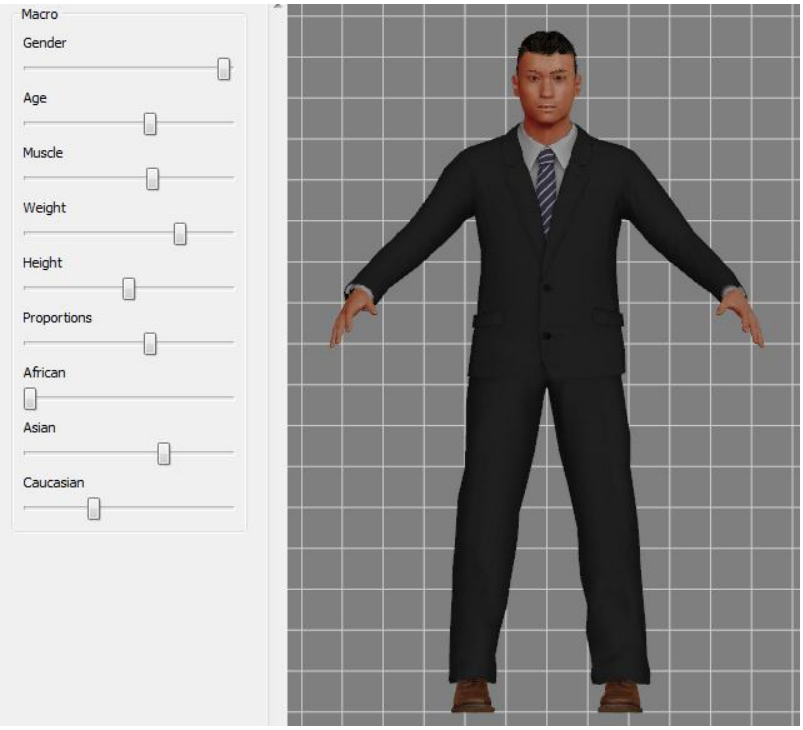
Model, kostur i animacija *Kenchi-a* su izrađeni pomoću besplatnog alata otvorenog kôda *MakeHuman*. *MakeHuman* omogućuje izradu ljudskih modela iz parametriziranog generičkog modela kojem je moguće promjeniti svaki aspekt lica i tijela. Uz sami model se isporučuje i generički kostur i ciljni modeli izraza lica i pokretanja dijelova lica.

4.1. Izrada virtualnog lika

Model Kenchi-a se sastoji od osnovnog modela koji se sastoji od cijelog tijela, kostura i odjeće, niza ciljnih modela koji odgovaraju osnovnom uz promjenjene pozicije vrhova za definiranje animacije i niza tekstura za dijelove tijela, lica i odjeće. Teksture i modeli su izgrađeni u alatu *MakeHuman*, zapisani u FBX format, dorađeni u alatu za modeliranje *Maya* gdje su definirane i animacije. Krajnji rezultat je FBX datoteka sa osnovnim modelom i definiranim animacijama na ciljne modele i niz tekstura materijala i normala definiranih u PNG formatu.

4.1.1. MakeHuman

MakeHuman je besplatan alat otvorenog kôda za izradu realističnih 3D ljudskih modela za ilustracije, animaciju i igre[2]. Modeli izrađeni ovim alatom su licencirani CC0 licencom koja omogućava potpunu slobodu nadogradnje, poboljšanja i korištenja modela.



Slika 5. MakeHuman makro parametri

[2] *MakeHuman* se može preuzeti sa http://www.makehuman.org/

Za izradu ljudskog modela *MakeHuman* koristi generički model koji se oblikuje prema želji koristeći makro i mikro parametre. U osnovni set parametara su uključeni spol, starost, mišićavost, težina, visina, proporcije i rasa. Daljnja podjela parametara je organizirana u parametre spola, lica, torza, ruku, nogu i konkretnih mjera čovjeka kao što je širina vrata, struka, ruku i slično. Svaka od ovih podjela ima podkategorije; samo za lice postoji 17 podkategorija koje uključuju oblik glave, vrata, očiju, nosa i usta. Ukupno postoji 125 parametara samo u kategoriji lica. Parametri ne mijenjaju samo model, nego i teksturu modela.

Slika 6. Makehuman sučelje za modeliranje lica

Osnovni model uključuje geometrije tijela i lica. Uz ovo postoji i niz dodatnih modela koje je moguće uključiti u modeliranje i animaciju. U aplikaciji su dostupni modeli odjeće (hlača, majica, obuće i šešira), kose, zubiju, obrva, trepavica i jezika gdje svaki model ima više tekstura na odabir. Uz teksture materijala modela uključene su i teksture normala gdje je to potrebno. Za potrebe ovog rada uključene su dodatne geometrije odjeće, kose, zubiju, obrva, trepavica i jezika.

Za animacije je potrebno izraditi ciljne modele koji su istovjetni osnovnom modelu uz promjene u vrhovima trokuta na dijelu modela kojeg želimo animirati. Ovo je moguće sa parametrima podešavanja izraza lica koji su podijeljeni u kategorije obrva, očiju, usta, nosa i vrata. Ciljni modeli moraju odgovarati akcijskoj jedinici koju želimo koristiti za kontroliranje animacije, pa tako u animaciji koju kontrolira akcijska jedinica dizanja lijeve obrve moramo podignuti lijevu obrvu putem parametara. Ovo često uključuje kombiniranje parametara kako bi se dobila željena animacija. Svaki ciljni model se zapisuje zasebno i kombinira u alatu za 3D modeliranje sa osnovnim modelom kako bi se definirala animacija pomoću mješanja oblika.

Generirani osnovni model (i svaki ciljni model) je sastavljen od 35 000 vrhova i u kombinaciji sa teksturom normala i materijala predstavlja dovoljnu količinu detalja kako bi sve animacije bile jasno uočljive.

4.1.2. Izrada animacije

Za izradu samih animacija korišten je alat za 3D modeliranje, animiranje i iscrtavanje *Maya* koja je besplatna u ograničenoj verziji za studente. Svi ciljni modeli su uključeni u scenu sa osnovnim modelom i definirano je 12 animacija kojima se upravlja pomoću akcijskih jedinica. Animacije su definirane linearnom interpolacijom i postavljene prije animacije kostiju u hijerarhiji deformacija. Jedna animacija može djelovati na jedan model, a pošto lice i tijelo tvore više modela, nekada je potrebno definirati više animacija za jednu akcijsku jedinicu. Primjer takve animacije je otvaranje usta što uključuje animaciju za lice, zube i jezik.

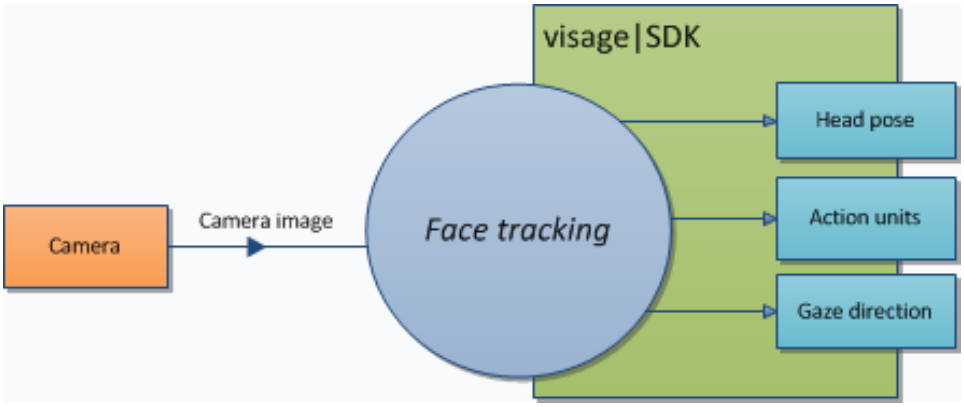
4.2. Izrada aplikacije

Aplikacija je izrađena koristeći pokretački sustav Unity za koji je napravljena biblioteka za integraciju visage|SDK sustava. U Unity-u se postiglo iscrtavanje modela ljudskog lika, dohvaćanje podataka iz sustava za praćenje lica, animacija modela koristeći mješanje oblika i sustav za konfiguraciju pokretanja animacija koristeći akcijske jedinice.

Za programiranje ponašanja aplikacije koristio se programski jezik C# kojeg Unity izvršava koristeći Mono pokretačku platformu na svim podržanim platformama.

4.2.1. Unity

Unity je pokretački sustav čija glavna odlika je podrška za izgradnju za više platformi. Ovo čini izgradnju višeplatformskih aplikacija efikasnijom od izrade zasebne aplikacije za više platformi pošto se jedino dio aplikacija koji ovisi o specifičnostima jedne platforme mora raditi specifično za tu platformu. U ovoj aplikaciji to je integracija sa visage|SDK bibliotekom.

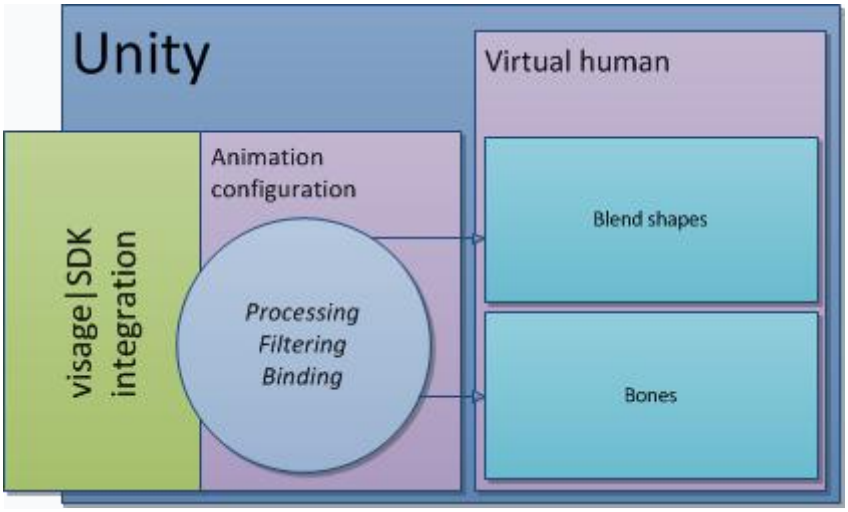


Slika 7. Vanjsko sučelje visage|SDK bilioteke

Unity radi na principu komponenti i objekata u sceni. Scena sadrži više objekata koji mogu sadržavati komponente. Uz postojeće komponente za iscrtavanje modela i slično, korisnik može izraditi vlastite i razmjenjivati poruke između njih. Objekti mogu biti posloženi u hijearhije. Svaki objekt nužno sadrži transformaciju koja određuje poziciju, rotaciju i skaliranje objekta, a od ostalih su dostupne komponente za sustave čestica, fiziku, umjetnu inteligenciju i razne vizualne efekte. Komponente se mogu programirati u jednom od tri programska jezika – C#, UnityScript (sličan Javascript-u) i Boo (dijalekt Python-a) [3].

Za uređivanje scena Unity pruža interaktivni alat pomoću kojeg je moguće stvarati objekte, slagati komponente i pozicionirati objekte u 3D prostoru. Sučelje se može prilagoditi potrebama aplikacije također putem kôda pisanog u jednom od tri dostupna jezika što je iskorišteno za izradu sučelja prilagođenog postavljanju veza između akcijskih jedinica i animacija.

Izrađeni model *Kenchi-ja* je učitan u Unity koristeći FBX format u kojem su bile i pripadne animacije i za svaki dio modela je napravljen zasebni materijal sa teksturama materijala i normala koje su isto tako učitane koristeći PNG format koji podržava prozirne dijelove.



Slika 8. Arhitektura Unity aplikacije

U aplikaciji je korištena jedna scena u kojoj je postavljena kamera, pozadina i model Kenchi-ja. Od ostalih objekata tu je komponenta koja služi za integraciju sa visage|SDK sustavom i komponenta koja prikazuje sliku sa web kamere. Model se sastoji od hijerarhije objekata koje služe za prikaz model – svaki sačinjen od vlastite geometrije i sustava za iscrtavanje koji koristi jedan materijal, i objekata čijom manipulacijom se mogu proceduralno animirati kosti. Svaki objekt za prikaz modela ima parametre za manipulaciju animacijom mješanja oblika koji također služi za proceduralnu animaciju. Izgrađeni sustav za konfiguraciju koristi ove parametre za pokretanje animacije pomoću akcijskih jedinica. Izgrađene su i komponente koje zasebno animiraju određene dijelove modela. Komponenta za animaciju smjera gledanja uzima vrijednosti smjera u kojem gleda korisnik i rotira model očiju kako bi oni odgovarali stvarnom smjeru gledanja. Za preslikavanje translacije i rotacije glave se brine zasebna komponenta koja uzima te vrijednosti iz sustava praćenja lica i pomiče kost koja kontrolira glavu modela.

4.2.2. Integracija visage|SDK u Unity

Kako Visage Technologies nudi visage|SDK u verzijama za nativne aplikacije posebno za iOS i za Android potrebno je integrirati obje verzije sustava u Unity, tj. omogućiti dohvaćanje vrijednosti i upravljanja sustavom. Obje verzije se integriraju na isti način – potrebno je napraviti biblioteku koja koristi visage|SDK i otkriva funkcije za upravljanje i dohvat podataka, te funkcije iste biblioteke koristiti pomoću poziva vanjskim funkcijama iz C# skripte pokretane u Unity-u [2]. Odabir ispravne datoteke se radi prilikom same izgradnje aplikacije za određenu platformu.

Biblioteka koja otkriva funkcije je napisana u C++-u i pruža iste funkcije za obje platforme. Za Android platformu biblioteka se izgrađuje Android NDK sustav, a za iOS se izgrađuje u sklopu Unity projekta u Xcode razvojnom okruženju.

Lista i kratak opis bitnih funkcija i njihovih parametara se nalazi u tablici.

Prototip	Opis
`initTracker(config)`	Inicijalizacija sustava koristeći konfiguraciju na zadanoj putanji
`trackFromCam()`	Počinje praćenje lica sa kamere
`stopTracker()`	Zaustavlja praćenje lica
`get3Ddata(...)`	Dohvaća translaciju i rotaciju glave i stanje praćenja putem parametara
`getCameraInfo(...)`	Dohvaća širinu i visinu slike sa kamere i fokus kamere
`getActionUnitCount()`	Dohvaća ukupan broj akcijskih jedinica u sustavu
`getActionUnitName(index)`	Dohvaća ime akcijske jedinice sa danim indeksom
`getActionUnitValues(...)`	Dohvaća vrijednosti svih akcijskih jedinica putem polja
`getGazeDirection(...)`	Dohvaća smjer gledanja

4.2.3. Sustav za konfiguriranje animacije

Za lakšu izmjenu animiranih virtualnih likova i konfiguraciju animacije implementiran je sustav za konfiguriranje. Proces vezanja akcijskih jedinica za pojedine modele i njihove animacije uz filtriranje i normalizaciju akcijskih jedinica se kontrolira putem tekstualne datoteke koja sadrži definicije tih veza.

Animaciju kontroliramo jednim realnim brojem u granicama [0, 1], dok vrijednosti akcijskih jedinica mogu izaći izvan tih granica. Svaka akcijska jedinica ima svoje granice koje je potrebno eksperimentalno utvrditi. Sustav za konfiguriranje mapira vrijednosti korištene akcijske jedinice za pojedinu animaciju na vrijednosti [0,1], te vrijednosti filtrira koristeći vremenski filter sa težinskom, pomičnom srednjom vrijednošću kako bi se moglo utjecati na glatkoću i odziv animacije.

Parametri veza akcijskih jedinica i animacija su slijedeći:

- au_name – ime korištene akcijske jedinice kako je zapisana u konfiguraciji sustava za praćenje lica
- blendshape_identifier – identifikator imena animacije u obliku ime_modela:indeks_animacije. Objekt se mora nalaziti u trenutnoj sceni.
- min_limit – najmanja vrijednost akcijske jedinice koju želimo mapirati na animaciju
- max_limit – najveća vrijednost akcijske jedinice koju želimo mapirati na animaciju
- inverted – ako je ovaj parametar postavljen, vrijednosti akcijskih jedinica se obrnuto mapiraju
- weight – kontrolira skaliranje vrijednost akcijske jedinice
- filter_window – veličina prozora koji se koristi u filteru, veći prozor će dati glađu animaciju, no i sporiji odziv
- filter_amount – količina filtriranja

U jednoj sceni je moguće koristiti više ovakvih konfiguracija za mapiranje i više od jednog virtualnog lika.

4.2.4. Filtriranje akcijskih jedinica

Praćenjem lica se često dobivaju nestabilne vrijednosti akcijskih jedinica koje, da direktno kontroliraju animaciju, bi rezultirale trzanjem u animaciji i zbog toga je potrebno filtrirati vrijedosti. Filtriranje se vrši nakon mapiranja vrijednosti na granice [0,1] kako bi ponašanje filtera bilo konzistentno.

Filtrirana vrijednost akcijske jedinice $\boldsymbol{t}_i$ gdje je *i* indeks okvira u veličini prozora *k*, se dobiva iz slijedeće jednadžbe:

$$\boldsymbol{t}_i = \frac{\sum_{j=0}^{k} w_j \boldsymbol{t}_{i-j}}{\sum_{j=0}^{k} w_j},$$

gdje $\boldsymbol{t}_{i-j}$ označava vrijednost akcijske jedinice u okviru indeksa $i - j$. Težine w_j su definirane na slijedeći način:

$$w_j = e^{-j \cdot H \cdot max_{l \in [1,k]} \|\boldsymbol{t}_i - \boldsymbol{t}_{i-l}\|}.$$

Parametri veličine prozora i konstante H se kontroliraju iz konfiguracije animacije parametrima filter_window i filter_amount za svaku pojedinu animaciju [4].

4.2.5. Višeplatformska implementacija

Ispravna verzije biblioteke za pojedinu platformu se odabire pri izgradnji projekta kroz pretprocesorske naredbe što znači da će aplikacija raditi na Windows, Android i iOS platformi.

Razlike u platformama su u načinu dohvata slike sa kamere – visage|SDK upravlja pristupom kameri i tekstura slike kamere se dohvaća iz biblioteke na dva različita načina; slanjem pokazivača na polje slike i dimenzija slike iz biblioteke u Unity za Windows platformu i pisanjem direktno u teksturu koju koristi Unity za iOS i Android platforme. Kako sustav za praćenje prati lice u dretvi nezavisnoj od one u kojoj se pokreće Unity, implementiran je sustav dvostrukog spremnika koji garantira da tekstura koju čita Unity neće biti prepisana novim podacima u trenutku čitanja. Ovo je riješeno u biblioteci za integraciju u Unity i sakriveno od korisnika.

5. Rezultati

Izrađeni virtualni lik prati lice korisnika i pokriva jedan dio pokreta dijelova lica za koje je napravljena animacija i koje sustav za praćenje lica prepoznaje i prati. Kako bi se poboljšali rezultati potrebno je povećati preciznost praćenja, uvesti finije praćenje podjelom trenutnih akcijskih jedinica na manje i izražajnije animirati lika. U nastavku su neki rezultati dobiveni korištenjem aplikacije.

Slika 9. Neutralna poza lica

Slika 10. Izraz lica sa otvorenim ustima

Slika 11. Pomicanje obrva

Slika 12. Detekcija zatvorenih očiju

Slika 13. Kombinacija zatvorenog oka i podignute obrve

6. Zaključak

Mobilna aplikacija u kojoj je implementirana animacija pokretana praćenjem lica prikazuje animirani virtualni lik koji prati pokrete glave i pokrete dijelova lica korisnika. Za potrebe aplikacije izrađen je virtualni lik sa sustavom kostiju i animacijama pokreta dijelova lica. Proceduralnom animacijom kostiju je animirano pomicanje i rotacija glave, a proceduralnom animacijom mješanjem oblika su animirani pokreti dijelova lica. Proceduralne animacije se kontroliraju očitanim podacima koje sustav za praćenje lica dobiva preko ugrađene kamere. Podaci se filtriraju i dobivena animacija je glatka i sa brzim odazivom na korisnikove kretnje. Aplikacija je višeplatformska i dodavanje nove platforme je moguće integracijom sustava za praćenje lice na toj platformi u grafički pokretački sustav.

Virtualni lik prikazuje pokrete dijelova lica koje sustav za praćenje može očitati – to su pokreti obrva, usta, nosa i očiju. Za nove pokrete potrebno je implementirati njihovo praćenje u sustavu, izraditi animaciju pokreta i u sustavu za konfiguraciju dodati mapiranje na odgovarajuću animaciju. Za dodavanje novog virtualnog lika je potrebno izraditi kostur i ciljne modele za animacije pokreta dijelova lica i mapirati akcijske jedinice na te animacije. Razvijeni sustav za konfiguraciju omogućava mapiranje bez dodatnog programiranja.

Sustav za konfiguraciju je moguće i proširiti na način da se mogu kontrolirati sva svojstva objekta – ne samo animacija, te ovime iskoristiti sustav za praćenje lica kao općeniti ulazni uređaj sa jedinstvenim načinom upravljanja.

7. Literatura

[1] J. AHLBERG: „CANDIDE-3 -- an updated parameterized face“, *Report No. LiTH-ISY-R-2326*, Dept. of Electrical Engineering, Linköping University, Sweden, 2001.

[2] visage|SDK documentation

[3] Unity documentation
http://unity3d.com/learn/documentation

[4] T. WEISE, S. BOUAZIZ, H. LI, M. PAULY: „Realtime performance-based facial animation“

8. Naslov, sažetak i ključne riječi

Naslov

Animacija pokretana praćenjem lica na pokretnim uređajima

Sažetak

U radu je objašnjena izrada virtualnog lika sa animacijama pokreta dijelova lica i aspekti izrade mobilne, višeplatformske aplikacije koja koristi sustav za praćenje lica za proceduralno pokretanje animacije izrađenog virtualnog lika. Implementacija dokazuje da je moguće koristiti praćenje lica kao uređaj kontroliranja animacija na pokretnim uređajima.

Ključne riječi

Praćenje lica, proceduralna animacija, mješanje oblika, virtualni lik, mobilna aplikacija, Visage, Unity, MakeHuman

Title

Performance-driven animation on mobile devices

Abstract

This thesis explains how to make a virtual character with defined facial animations and the aspects of building a mobile, crossplatform application which uses face tracking software for driving facial animation. This implementation solves the problem of using face tracking software as a device for controlling animation on mobile devices.

Keywords

Face tracking, procedural animation, blend shapes, virtual characters, mobile aplication, Visage, Unity, MakeHuman