SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3803

Animacija pokretana praćenjem lica u programskom paketu Maya

Ela Marušić

Zagreb, lipanj 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 14. ožujka 2014.

ZAVRŠNI ZADATAK br. 3803

Pristupnik: **Ela Marušić (0036469019)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Animacija pokretana praćenjem lica u programskom paketu Maya**

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 animacije, igara, zabave i komunikacija.

Na Zavodu za telekomunikacije dostupan je sustav za praćenje lica u stvarnom vremenu. Potrebno je implementirati dodatak (engl. plugin) za programski paket Maya koji će omogućiti njegovo povezivanje sa sustavom praćenja lica.

Vaša je zadaća proučiti postojeće sustave te predložiti i implementirati izvedbu dodatka za programski paket Maya za animaciju pokretanu praćenjem lica.

Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 13. lipnja 2014.

Mentor:

Prof.dr.sc. Igor Sunday Pandžić

Otpravljeno

Doc.dr.sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Prof.dr.sc. Siniša Srbljić

Sadržaj

Uvod

Računalni vid i animacija su dvije grane tehnologije čija je suradnja imala velik utjecaj na mnoga područja, posebice industriju zabave. Kako je animacija lica složen i za čovjeka često zahtjevan zadatak, mogućnost automatizacije tog procesa je vrlo privlačna. Jedna od mogućnosti, koja je ujedno i tema ovog rada, je praćenje karakterističnih točaka lica te animiranje 3D modela lica pomoću tih podataka.

Prvo je poglavlje uvod u programski paket Maya koji se fokusira na mogućnosti proširenja u obliku skripti i dodataka (engl. plug-ina). Drugo poglavlje bavi se sustavom visage|SDK te se daje pregled nekoliko razreda najbitnijih za problematiku rada. Nakon upoznavanja s navedenim sustavima za praćenje i animaciju lica, prelazi se na prijedlog rješenja njihovog spajanja te na objašnjavanje razreda programske implementacije predloženog rješenja.

1. Autodesk Maya

Maya je Autodeskov program za 3D modeliranje prikladnog imena –značenje sanskrit izraza "maya" može se interpretirati kao "iluzija". Jedan je od poznatijih 3D programa te je korištena u mnogim poznatim i uspješnim filmovima, serijama i igrama. [LITwiki]

Svoju popularnost jednim dijelom duguje lakoj proširivosti i fleksibilnosti. Uz mogućnost automatizacije čestih procesa skriptama pisanim Pythonom ili Mayinim jezikom MEL, moguće je i dodavati joj potpuno novu funkcionalnost u obliku novih komandi i čvorova. Za ovo je potrebno napisati dodatak (engl. plug-in) koristeći Mayino aplikacijsko programsko sučelje (engl. application programming interface, API). Programeru se na izbor nude izvedbe u jeziku C++, Pythonu te .NET jezicima. Prije upuštanja u izradu vlastitog *plug-ina*, poželjno je pobliže se upoznati s Mayinom arhitekturom.

1.1. Ukratko o Dependency Graphu

Mayin graf ovisnosti (engl. Dependency Graph, DG) ujedno je i njena srž. Svi podatci scene zapravo su njegovi čvorovi te svaki čvor ima svoje ulazne i izlazne podatke. Važno je razumjeti tok podataka u grafu – takozvanu "push-pull" metodu. Izlazna vrijednost čvorova u *Dependency Graphu* računa se samo kada je to potrebno. Ako se promijeni vrijednost ulaza čvora, očito je da je potrebno ponovno izračunati izlazne vrijednosti koje ovise o promijenjenom ulazu. No, Mayinim pristupom to se neće dogoditi dok se izlazna vrijednost tog čvora ne zatraži.

Sljedeća situacija ilustrira ovu funkcionalnost. Graf se sastoji od dva čvora. Prvi je transformacijski te je njegov izlaz povezan na ulaz drugog koji je *shape* čvor. Dodavanjem novog čvora u graf njegov izlaz spajamo na ulaz prvog transformacijskog čvora. Kako se vrijednost ulaza spomenutog čvora sad promijenila, označena je njegova zastavica promjene (engl. dirty flag). Sada je potrebno nastaviti prolaziti kroz graf i označiti sve izlaze i ulaze koji ovise o

promijenjenom ulazu jer se i njihova vrijednost promijenila. Treba primjetiti da se nove vrijednosti nisu izračunale – samo su postavljene zastavice. Tek kada Maya zatraži izlaznu vrijednost nekog čvora kojoj je postavljena zastavica promjene, taj će čvor ponovno izračunati njenu vrijednost. No, ako je zastavica postavljena i na ulazu čija se vrijednost koristi pri računanju, čvor će zatražiti ponovno izračunavanje ulaza. Očigledno je da će se ovako "putovati" unazad sve do prvog ulaza ili izlaza čija zastavica promjene nije postavljena. Vrijednosti puta se zatim izračunavaju sve do onog prvog izlaza kojeg je Maya zatražila.

Zaključak je jednostavan – ako nema potrebe za izlazima nekog čvora, njegova se compute() funkcija neće pozvati. Razumijevanje ovog koncepta od velike je pomoći pri programiranju dodataka za Mayu.

1.2. MEL

MEL je jednostavan skriptni jezik bogatih mogućnosti poput mijenjanja Mayinog grafičkog korisničkog sučelja (engl. graphical user interface, GUI), integracije Maye u cjevovod pomoću novih eksportera i importera datoteka te automatizacije često korištenih nizova naredbi. Čitav Mayin GUI izveden je pomoću MEL-a te svaka interakcija mišem u pozadini pokreće MEL kod. Uz dostatno znanje jezika, moguće je zaobići korištenje GUI-ja i ubrzati proces modeliranja i animiranja.

Neki od načina izvršavanja MEL naredbi:

1. Preko komandne linije
   Najbrži način za izvođenje jednostavnih naredbi. Komandna linija nalazi se na dnu prozora.
2. Komandna ljuska
   Do komandne ljuske dolazi se izbornicima Windows > General Editors > Command Shell
3. *Script Editor*
   Jednostavan način za unošenje i izvršavanje više linija naredbi. *History Panel* prikazuje izvršene naredbe i njihove rezultate. Zanimljiva je mogučnost uključivanje *Echo all commands* opcije. Tada će se ispisivati sve izvršene

naredbe, ne samo one koje korisnik unese. Ovo je jednostavan način za učenje naredbi ili stvaranje skripti bez znanja jezika. Dovoljno je mišem napraviti željeni niz akcija i kopirati naredbe koje *Script Editor* ispiše.

4. *Shelf*
   Niz naredbi može se spremiti kao *Shelf* gumb te se može pokrenuti jednim klikom.

5. Pokretanje skripte
   Naredbe se mogu spremiti u MEL datoteku za kasnije pokretanje. Naredba za izvršavanje skripte je:

```
source imeSkripte;
```

Format MEL naredbe je sljedeći:

```
imeNaredbe –zastavica vrijednost;
```

Naredba može imati nula ili više zastavica te zastavice mogu imati nula ili više vrijednosti. Pomoću naredbe

```
help imeNaredbe;
```

možemo dobiti više informacija o zastavicama neke naredbe i vrijednostima koje one zahtijevaju.

Naredbe imaju tri načina djelovanja: *create*, *query*, i *edit*. Prvi način je podrazumijevan, a druga dva se aktiviraju korištenjem zastavica **–q** i **–e**. *Query* naredba ne mijenja Mayino stanje već samo vraća neku zadanu vrijednost. Naredba u *edit* stanju mijenja vrijednost nekog atributa. Treba imati na umu da ne podržavaju sve naredbe sve načine djelovanja te je preporučljivo konzultirati se s priručnikom MEL naredbi dostupnom u izborniku Help > MEL Command Reference.

Velika većina Mayine funkcionalnosti dostupna je pomoću MEL naredbi. Iako C++ API i MEL imaju dosta preklapanja oko mogućnosti, neke su ipak ekskluzivne. Stvaranje grafičkog korisničkog sučelja nije moguće pomoću C++ API-ja, no MEL ne može stvarati nove naredbe i čvorove. Također, usporede li se MEL i C++ kodovi koji obavljaju istu funkciju, C++ kod će se uglavnom brže izvoditi.

1.3. C++ API

Funkcionalnost Maye lako je proširiva izgradnjom *plug-ina,* Pomoću API-ja mogu se stvarati nove naredbe i novi čvorovi koji se integriraju u Mayinu strukturu. Naredbe se, kao i ostale MEL naredbe, mogu pozivati u komandnoj liniji i skriptama, a čvorovi se mogu dodavati u *Dependency Graph*.

Postoje četiri osnovna tipa C++ objekata. To su:

- Omotači (engl. wrapper)
  Razredi omotači za strukture podataka i iteratore, npr. MString, MIntArray, Mvector, MItDependencyGraph, etc.
- *Proxy*
  Apstraktni razredi čijim se naslijeđivanjem implementiraju nove naredbe i čvorovi. Imena im uvijek započinju s "MPx". Neki primjeri su: MPxCommand, MPxNode, MpxDeformerNode, etc. Može se primijetiti da se uz generalni razred za stvaranje čvore nude i specifični za određene vrste čvorova, npr. *Deformer* čvor.
- Objekti
  MObject je osnovni razred koji može predstavljati bilo koji objekt u Mayi, od krivulja do izvora svjetla. Ovo omogućuje lagano prenošenje objekata pomoću parametara funkcija. Bitno je napomenuti da je instanca razreda MObject samo pokazivač na stvarni objekt kojem se drukčije ne može pristupiti. Na ovaj način Maya zadržava potpunu kontrolu nad podatcima i važno je da se u *plug-inu* ovi pokazivači ne pamte jer se ne može garantirati da će uvijek pokazivati na isti objekt.
- Skupovi funkcija
  Razredi koji sadrže skup funkcija za određenu vrstu čvora, npr. MFnDagPath, MFnMesh, etc.

Iz opisa zadnja dva tipa objekata može se primijetiti da Maya ne koristi dizajn specifičan za objektno orijentiranu paradigmu, već su podatci i funkcije koje

operiraju nad tim podatcima odvojeni u zasebne klase. Ovim se smanjuje količina memorije potrebna za svaku instancu čvora.

Kako se nove naredbe stvaraju nasljeđivanjem MPxCommand razreda, naredbe koje definira *plug-in* tretiraju se kao i ostale MEL naredbe. Zbog ovog je jako bitno voditi računa o ispravnoj implementaciji *undo* i *redo* funkcionalnosti nove naredbe. Iako je moguće stvoriti naredbu bez ovih mogućnosti, to bi trebalo biti rezervirano isključivo za naredbe koje svojim izvođenjem ne mijenjaju Mayino stanje. Maya svaku izvršenu naredbu stavlja na stog s čijeg će se vrha, pri odabiru opcije *Undo* u izborniku, uzeti naredba i pozvati njezina funkcija undoIt(). Dobro je primijetiti da se instanca naredbe zatim ne uništava dok se ne pozove neka nova naredba koja će ići na stog. Razlog ovakvog ponašanja je što korisnik može odlučiti odabrati *Redo* opciju – u tom će slučaju biti potrebna instanca zadnje naredbe nad kojom je izvršen *Undo* kako bi pozvali njenu funkciju redoIt().

Kako naredba mora biti na vrhu stoga da bi se zvala funkcija undoIt(), pretpostavlja se da je scena u istom stanju u kojem je bila nakon što se funkcija prvi put izvršila. Dakle, *Undo* funkcionalnost obično se implementira tako da se netom prije izvođenja zapamti stanje dijelova scene koje će se mijenjati. Naredbom koja mijenja scenu, a ne nudi *Undo* mogućnost, narušavamo istinitost ove pretpostavke.

Prilikom programiranja *Dependency Graph* čvora, najbitnije je implementirati funkciju compute(). Može se reći da je to "mozak" čvora te je u njoj potrebno izvršavati sva preračunavanja ulaznih u izlazne vrijednosti. Izlaze i ulaze moguće je povezivati s ostalim čvorovima u *Dependency Graphu.* Daljne proučavanje procesa izvedbe *Dependency Graph* čvora prelazi okvire ovog završnog rada.

2. visage|SDK™

Produkt suradnje tvrtke Visage Technologies i istraživačke grupe HOTLab (Human-Oriented Technologies Laboratory), visage|SDK™ spaja metode računalnog vida i animacije ljudskog lica i tijela te nudi velik broj mogućnosti za upotrebu u aplikacijama. Uz detekciju i praćenje lica na slici i videu, postoji i mogućnost animacije usana na licu prema snimci govora te animacija lika i sinteza govora iz zadanog teksta. Za potrebe ovog rada, bit će dovoljno pobliže opisati samo mogućnosti praćenja lica.

2.1. VisageTracker2

VisageTracker2 je jedan od ključnih razreda koji se koriste pri praćenju lica. Za stvaranje instance nudi dva konstruktora – zajednički parametar im je ime konfiguracijske datoteke. Sadržajem spomenute datoteke praćenje lica može se dodatno definirati. Njeno se učitavanje odvija jednom pri svakom započinjanju praćenja, tako da je moguće promijeniti konfiguraciju *trackera* bez ponovnog pokretanja programa. [4] Neke od zanimljivijih mogućnosti su odabir između ručnog i automatskog prepoznavanja lica tijekom inicijalizacije praćenja te odabir *action unita* i 3D modela koje će se koristiti pri praćenju. Visage nudi nekoliko konfiguracijskih datoteka optimiranih za specifične situacije te se trenutno u svima koriste varijacije Candide3 modela razvijenog na Linköping Sveučilištu.

Na raspolaganju su tri funkcije koje pokreću proces praćenja lica. Ukoliko je cilj praćenje lica pomoću videa, potrebno je pozvati trackFromVideo() funkciju te joj proslijediti putanju do videa. Uključi li se i drugi parametar, ime FBA datoteke, animacija će se spremiti u danu datoteku. Pozivom funkcije trackFromCam() pali se kamera, ukoliko ista postoji, te se započinje praćenje lica pomoću kamere. Naposlijetku, funkcija trackFromRawImages() nudi mogućnost praćenja lica pomoću slike. Potonje dvije funkcije također mogu spremiti animaciju u FBA datoteku ukoliko im se ista zada kao parametar.

Nakon započinjanja praćenja, do rezultata se može doći na više načina. Pozivom funkcije getTrackingData() rezultati za obrađeni okvir kopiraju se u FaceData strukturu koja se funkciji šalje kao parametar. Ovo je najjednostavniji način te bi korisnije bilo implementirati sučelje VisageTrackerObserver.

Spomenuto sučelje sastoji se samo od jedne funkcije čiju je implementaciju potrebno napisati – notify(). Ako se, pomoću attach() funkcije u razredu VisageTracker2, implementacija VisageTrackerObservera poveže s instancom VisageTracker2, funkcija notify() će biti pozvana nakon svakog obrađenog okvira (uz izuzetak prvog koji služi za inicijalizaciju) te će joj biti predan pokazivač na instancu VisageTracker2 razreda. Pomoću dobivenog pokazivača može se pozivom već objašnjene getTrackingData() funkcije dobivati rezultat praćenja za svaki okvir.

Kako VisageTracker2 proširuje FbaAction razred, možemo ga funkcijom addTrack() dodati instanci FAPlayer razreda. Funkcijama play() i update() animacja se pokreće. Ukoliko je cilj iscrtavanje animacije, to je zgodno riješiti implementiranjem sučelja FARenderer. Visage|SDK™ nudi jednu moguću implementaciju u obliku razreda FAExporter. Navedeni razred učita sve okvire animacije te odvaja one ključne (engl. keyframe). Linearnom interpolacijom podataka ključnih okvira moguće je rekonstruirati originalnu animaciju sa svim okvirima.

Posljedni način je već spomenut – animacija se može spremiti u FBA datoteku za kasnije potrebe.

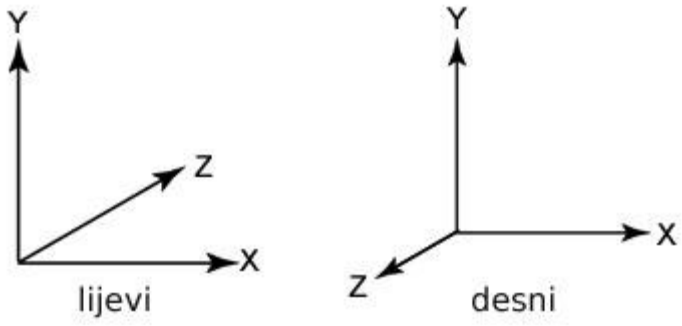
2.2. FaceData

FaceData razred služi kao spremnik podataka animacije za neki okvir. Već je spomenuto da se rezultati u FaceData spremaju prilikom poziva funkcije getTrackingData(). Bitno je napomenuti da neće nužno sve varijable biti inicijalizirane vrijednostima. Ovo ovisi o statusu praćenja te je isti moguće provjeriti na temelju vrijednosti koju vraća getTrackingData(). Moguće vrijednosti su:

- TRACK_STAT_OFF – praćenje je zaustavljeno ili nije započeto
- TRACK_STAT_OK – praćenje se odvija kako je očekivano

- TRACK_STAT_RECOVERING – tracker trenutno ne prepoznaje lice u videu
- TRACK_STAT_INIT – inicijalizacija trackera odvija se na početku praćenja i kada se dovoljno dugo (parametar u konfiguracijskoj datoteci) ne pronalazi lice u videu

Varijable faceTranslation i faceRotation su polja s tri decimalna broja. Vrijednosti u faceTranslation predstavljaju pomak glave od kamere u metrima, a vrijednosti u faceRotation rotaciju glave u radijanima. Potrebno je napomenuti da visage|SDK tracker koristi lijevi koordinatni sustav – potrebno je promijeniti predznak vrijednosti za koordinatu na z-osi ukoliko animaciju prenosimo u desni koordinatni sustav. Slika 2.1 prikazuje razliku između dva sustava.



Slika 2.1 Lijevi i desni koordinatni sustav

Ovime je riješena animacija pokreta glave. Ekspresije lica mogu se animirati pomoću vrijednosti točaka lica te pomoću vrijednosti akcijskih jedinica (engl. action unit). Tracker rijekom rada koristi akcijske jedinice te se iz njih izvode spomenute vrijednosti točaka lica. Zbog ovog je razloga preciznije koristiti akcijske jedinice.

Varijabla actionUnits sadrži pokazivač na blok decimalnih brojeva koji predstavljaju vrijednosti svih akcijskih jedinica. Broj se akcijskih jedinica može saznati varijablom actionUnitsCount, a varijablama actionUnitsNames i actionUnitsUsed njihova imena i informaciju iz konfiguracijske datoteke koja definira preskače li se tu akcijsku jedinicu.

Razred FaceData sadrži još mnogo podataka, no za potrebe ovog rada dostatno je razumjeti objašnjene varijable.

3. Visage Facial Animation plug-in

3.1. Specifikacija zahtjeva

Potrebno je implementirati *plug-in* za programski paket Maya koji će omogućavati generiranje animacije lica prema zadanom videu. Korisniku se treba prikazati grafičko sučelje u kojem će moći odabrati putanju do videa te mapirati interpolacijske jedinice koje su definirane u trenutnoj sceni s onima koje će se koristiti prilikom animacije. Odabiranjem 3D modela lica i klikom na gumb pokreće se generiranje animacije. Rezultat je animacija ključnim okvirima (engl. keyframe) 3D modela.

3.2. Prijedlog rješenja

Za izvedbu željenog ponašanja *plug-ina* predlaže se kombinacija skripte pisane jezikom MEL i naredbe stvorene Mayinim C++ API-jem. Skriptom će se pokretati grafičko korisničko sučelje, a skripta će, ukoliko su unesene potrebne informacije i stisnut gumb "Animate", pokrenuti naredbu. Putanju do korištenog videa i imena interpolacijskih jedinica skripta će naredbi proslijediti u obliku argumenata. Naredba će generirati animaciju pomoću visage|SDK™ *trackera*. Pomoću vlastite implementacije sučelja VisageTrackerObserver, za svaki okvir videa zatražit će se podatci animacije i prema njima stvoriti ključni okvir u Mayi.

3.3. Programska izvedba

3.3.1. Razred pluginMain

Ovo je generični razred koji sadrži funkcije za inicijalizaciju plug-ina. U funkciji initializePlugin() registriramo plug-in te naredbu koju sadrži.

```
MFnPlugin plugin( obj, "Ela Marusic", "2013", "Any");
```

```
plugin.registerCommand( "vsgFacialAnim", vsgFacialAnim::creator,
                vsgFacialAnim::createSyntax );
```

Analogno, u funkciji uninitializePlugin() odjavljuju se registrirani plug-in i naredba.

3.3.2. Razred vsgFacialAnimCmd

U funkciji createSyntax() definiraju se zastavice, njihova kratka i duga imena te tip argumenta kojeg primaju.

```
syntax.addFlag( "-v",  "-video", MSyntax::kString );
syntax.addFlag( "-bs",  "-blendshapes", MSyntax::kString );
```

Naredba treba moći preko zastavica primiti putanju do videa te polje imena interpolacijskih jedinica. Kako se preko zastavica ne mogu slati polja stringova, razred očekuje jedan string u kojem nakon svakog imena jedne jedinice slijedi točno jedan razmak. Ako korisnik nije zadao neku interpolacijsku jedinicu, umjesto njenog imena dobit će se prazan string. Razmak je izabran kao delimiter jer ga imena BlendShapeova u Mayi ne mogu sadržavati. Primanje argumenata i parsiranje stringa u vector strukturu obavlja se u funkciji doIt(). Zatim se stvaraju instance razreda VisageTracker2 i SimpleObserver te započinje praćenje lica uz putanju videa kao parametar.

```
VisageTracker2 *tracker =  new VisageTracker2( "Facial Features
                Tracker - Asymmetric.cfg" );
SimpleObserver *obs = new SimpleObserver( &dgMod, &blendShapes );

                tracker->attach( obs );
tracker->trackFromVideo( avi.asChar(), NULL );
```

Kako ova naredba mijenja stanje Maye generiranjem animacije, potrebno je implementirati funkcije redoIt() i undoIt(). Najjednostavniji način za ovo je korištenje MDGModifier razreda. Instanci tog razreda mogu se zadavati MEL naredbe te pomoću njenih funkcija doIt() i undoIt() brigu o *undo* i *redo*

funkcionalnosti prepustiti njoj. U ovom razredu, pozive ovih funkcija potrebno je smjestiti u funkcije sličnih imena, redoIt() i undoIt().

MEL naredbe se razredu MDGModifier zadaju unutar SimpleObserver razreda pa je prije poziva redoIt() funkcije potrebno pričekati da se praćenje lica zaustavi.

3.3.3. Razred SimpleObserver

Razred je implementacija sučelja VisageTrackerObserver. U konstruktoru prima pokazivače na MDGModifier razred kojemu treba zadati MEL naredbe te na vector strukturu koja sadrži imena BlendShapeova.

U implementaciji funkcije notify() pozivom getTrackingData puni se FaceData struktura s podatcima animacije. MDGModifier objektu zadaje se MEL naredba currentTime s vrijednosti timeStamp koju dobije kao parametar. Kako se vrijednost vremena u Mayi može spremati na vise načina, a dobiveni timeStamp predstavlja broj proteklih milisekundi u videu, treba biti oprezan i u naredbi dodati "millisec" nakon vrijednosti. Ukoliko status praćenja nije jednak TRACK_STAT_OK, za taj okvir animacija se ne generira. Inače, čvorove s transformacijskim matricama za odabrane modele možemo obići iteratorom kojeg dobivamo sljedećim kodom:

```
MGlobal::getActiveSelectionList( list );

MItSelectionList iter( list, MFn::kTransform );
```

Svakoj matrici promijenimo vrijednosti s obzirom na vrijednosti u faceTranslation i faceRotation poljima. Vrijednosti rotacija treba pomnožiti sa 180/π da bi dobili vrijednosti u stupnjevima. Vrijednosti translacije po z osi treba promijeniti predznak jer Maya koristi desni, a *tracker* lijevi koordinatni sustav.

Animacija ekspresija lica odvija se prolaskom kroz sve akcijske jedinice trackera i kopiranjem njihovih vrijednosti na odgovarajuće BlendShapeove. Ako je ime nekog BlendShapea prazan string, pripadajuća akcijska jedinica se preskače jer korisnik nije definirao BlendShape u sceni za nju.

Naposlijetku, za sve promijenjene vrijednosti postavlja se ključan okvir MEL naredbom setKeyframe.

3.3.4. Skripta vsgFacialAnim

MEL skripta napisana je s namjerom da se njenim pokretanjem i interakcijom kroz grafičko sučelje pokreće funkcionalnost *plug-ina*. Njom se definira sljedeća globalna procedura kojom se pokreće naredba *plug-ina* ili ispisuje error, ovisno o tome je li korisnik zadao putanju do videa.

```
global proc vsgAnimate(string $avi, string $bs[]) {
	if (size($avi) == 0) {
		confirmDialog -t "Error" -m "Please choose an AVI video." -ma "center" -b "OK" -icn "critical";
	} else {
		string $bsString;
		for ($i = 0; $i < size($bs); $i++) {
			$bsString += $bs[$i] + " ";
		}
		waitCursor -state on;
		vsgFacialAnim -v $avi -bs $bsString;
		waitCursor -state off;
	}
}
```

Imena BlendShapeova je potrebno formatirati u jedan string da bi se mogla poslati kao argument naredbi.

Definira se još jedna globalna procedura za otvaranje pretraživača datoteka.

```
global proc vsgBrowse() {
	string $file[] = `fileDialog2 -ds 2 -cap "Choose AVI video" -
ff "AVI Video (*.avi)" -fm 1 -okc "Choose"`;
	textField -edit -tx $file[0] vsgBrowseTextField;
}
```

Definiranje izgleda prozora glavni je dio skripte. U prozoru se prikazuje lista svih akcijskih jedinica *trackera* te se pokraj svake nalazi padajući izbornik s listom svih BlendShapeova u sceni zajedno s praznim stringom, ukoliko korisnik ne želi koristiti sve akcijske jedinice. Klikom na gumb “Browse” otvara se pretraživač AVI datoteka, a klikom na gumn “Animate” poziva se globalna procedura koja poziva naredbu za generaciju animacije.

Zaključak

Rezultat završnog rada je *plug-in* za programski paket Maya koji pomoću tehnika praćenja lica animira 3D model. Iako se uvijek može naći mjesta za proširenja, ovime je zadani zadatak uspješno obavljen.

Najveći problem pri izradi je bio nedostatak literature. Ipak, nakon dužeg proučavanja dokumentacije Maye i sustava visage|SDK, zbog fleksibilnosti navedenih programa pronađeno je efikasno rješenje pomoću interpolacijskih jedinica.

Problem je predstavio i nedostatak modela lica s definiranim svim interpolacijskim jedinicama koje se koriste pri animaciji. Iako ovo nije utjecalo na izradu samog *plug-ina*, otežalo je validaciju istoga.

Literatura

[1] Visage Technologies, visage|SDK 7.1 documentation

[2] Autodesk, Maya 2013 API documentation

[3] Autodesk, Maya 2013 User's Guide

[4] Visage Technologies, Tracker Configuration Manual

[5] David Gould, Complete Maya Programming

Animacija pokretana praćenjem lica u programskom paketu Maya

Sažetak

Spojem računalnog vida i animacije izronila je mogućnost automatiziranja procesa animacije - računalo generira okvire animacije prema videu željenih gesta i pokreta. Zadatak je ovog završnog rada implementirati ovakvu mogućnost u obliku plug-ina za programski paket Maya. Animatoru se treba, unutar Maye, omogućiti mapiranje vlastitih interpolacijskih jedinica s onima koje se koriste pri praćenju lica. Željeni rezultat je animacija odabranog 3D modela lica pomoću translacije, rotacije te vrijednosti interpolacijskih jedinica.

Ključne riječi: Maya, animacija lica, praćenje lica, interpolacijski ciljevi

Performance-driven animation in Maya software package

Summary

Computer animation has long since played an important role in the entertainment industry. Yet, animation has proven to be quite a tedious task for a human animator. Through a blend of computer vision and animation, a possibility of digital puppetry has emerged. Each frame of animation can be generated by a computer, using a motion capture video as input. The objective of this thesis is an implementation of this feature in the shape of a plug-in for the Autodesk product Maya. The animator can, using the aforementioned 3D modelling software, map previously created morph targets to those used by the face tracker. The result is a keyframe animation of a selected 3D model using translation, rotation and BlendShape weight values.

Keywords: Maya, facial animation, facial tracking, morph targets

Privitak

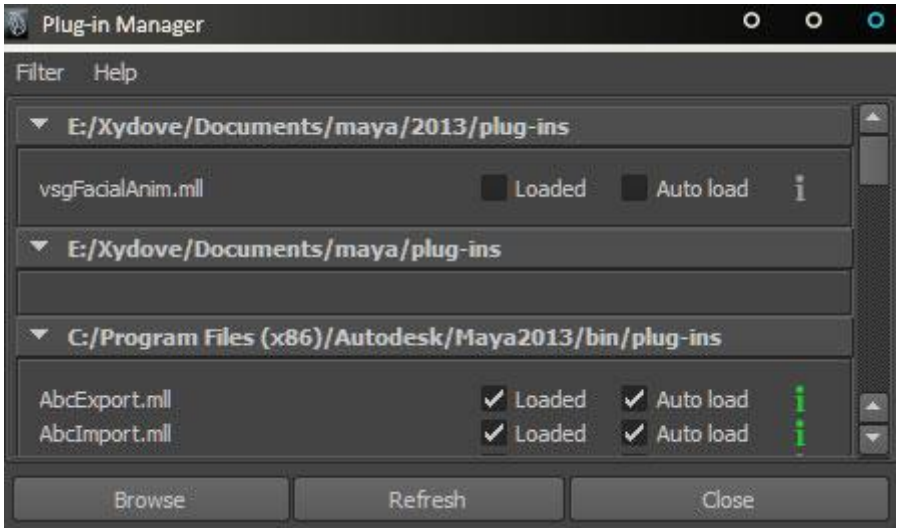
Upute za korištenje programske podrške

Za uspješno izvođenje *plug-ina* potrebno je na računalu imati instalaciju programa Maya 2013 (32-bit). Za *plug-in* je potrebno sljedeće:

- vsgFacialAnim.mll
- vsgFacialAnim.mel
- Visage|SDK licenca (.vlc)
- bdtsdata folder
- candide3-exp.wrl i candide3.fdp
- dlls folder
- Facial Features Tracker - Asymmetric

Da bi Maya mogla pronaći *plug-in* i sve potrebne podatke, datoteke je potrebno razmjestiti u odgovarajuće foldere. MEL skriptu treba premjestiti u bilo koji od foldera na koje pokazuje varijabla okruženja MAYA_SCRIPT_PATH, npr. …\Documents\maya\2013\scripts. Slično treba učiniti i s MLL datotekom, s tim da odredišni folder mora biti sadržan u MAYA_PLUG_IN_PATH varijabli. Naravno, korisnik može ove dvije datoteke staviti u bilo koji folder ukoliko će putanje dodati u spomenute varijable okruženja. Jedan od načina je editiranje Maya.env datoteke koja se obično nalazi u …\Documents\maya\2013. Sve DLL datoteke treba smjestiti u neki od foldera u sistemskoj %PATH% varijabli. Sve ostale datoteke i foldere treba premjestiti u …\Documents\maya\projects.

Sljedeći je korak pokrenuti Mayu i učitati *plug-in*. U izborniku Window treba naći izbornik Settings/Preferences te odabrati Plug-in Manager. Na slici 0.1 prikazan je prozor u kojem treba pronaći vsgFacialAnim plug-in i označiti kućicu uz Loaded/Auto load.

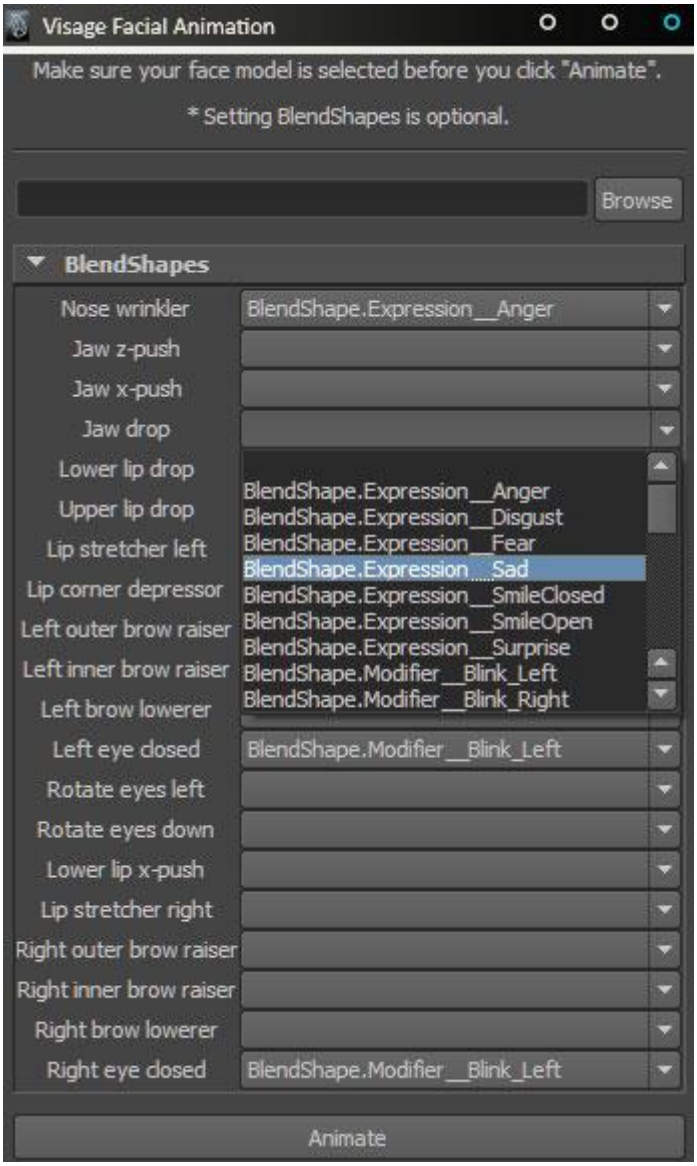


Slika 0.1 Učitavanje plug-ina

Ukoliko je odabrana opcija Auto load, ovaj će se korak automatski izvesti prilikom svakog pokretanja Maye. Prije pokretanja *plug-ina* potrebno je u *Script Editoru* ili *Command Lineu* upisati sljedeću naredbu:

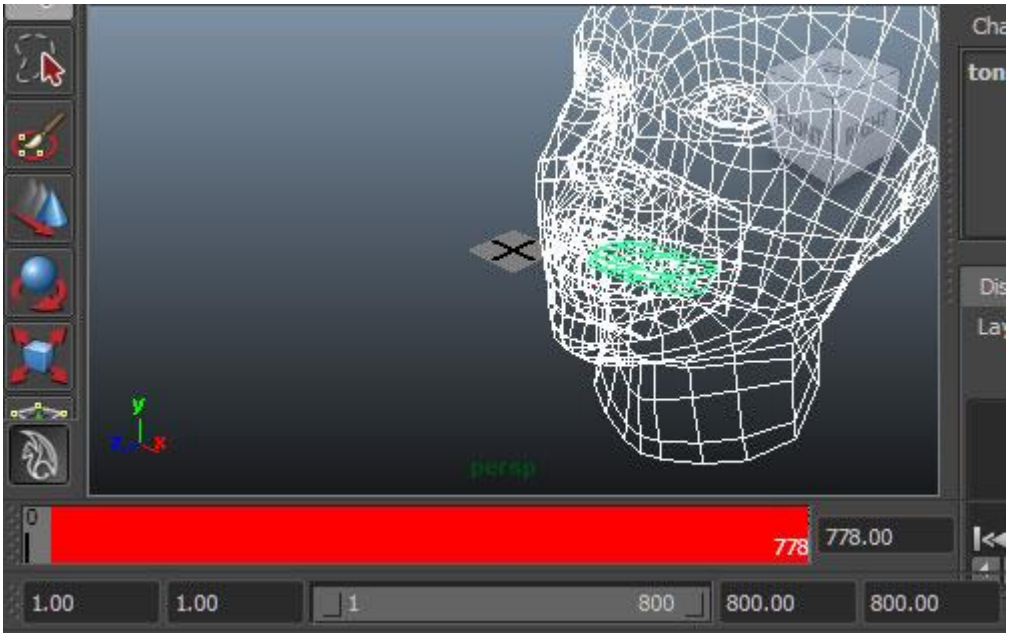
```
source vsgFacialAnim
```

Izvođenjem naredbe pokreće se MEL skripta koja stvara prozor za unošenje podataka potrebnih za animaciju lica vsgFacialAnim *plug-inom*. Klikom na *Browse* gumb pokreće se izbornik u kojem je potrebno odabrati putanju AVI videa prema kojem će se animirati. Unošenje interpolacijskih jedinica (takozvani BlendShapes u Mayi) je opcionalno, no kvaliteta animacije ovisi o ovom koraku. Kao što je prikazano na slici 0.2, svakoj interpolacijskoj jedinici koju *plug-in* koristi pri animaciji može se pridružiti odgovarajući BlendShape koji se trenutno nalazi u sceni. Korisniku se preporučuje da pažljivo obradi 3D model i poveže BlendShapeove kako bi ishod animacije bio što bliži izvornom videu.



Slika 0.2 GUI i postavljanje interpolacijskih jedinica

Prije klika na gumb *Animate*, potrebno je u sceni selektirati model lica. Zatim se korisnik treba naoružati strpljenjem – obrada obično traje koliko i sam video. Jednom kada obrada završi, rezultat će biti vidljiv u obliku keyframea u *Timelineu* na dnu prozora, kao što je prikazano na slici 0.3. Prije pokretanja ili iscrtavanja animacije, preporučljivo je provjeriti treba li mijenjati vrijednost završnog okvira.



Slika 0.3 Rezultat pokretanja plug-ina