Sveučilište u Zagrebu

Fakultet Elektrotehnike i Računarstva

ZAVRŠNI RAD br.3950

Sustav proširene stvarnosti s virtualnom šminkom

NIno Nikola Stanusic

Zagreb, lipanj 2015.

Veliku zahvalu dugujem mentoru dr. sc. Igoru Sunday Pandžiću koji me uputio u rad sa 3D grafikom u Unity sustavu, te svima koji su radili na VisageSDK projektu i pisanju dokumentacije koja je detaljno i razumljivo napisana..

SADRŽAJ

1. UVOD

Povijest ‘uljepšavanja’ lica šminkom datira još iz vremena starih Egipćana, pa sve do danas. Šminka se pri tome najčešće koristila za naglašavanje pojedinih dijelova lica, ali i za prikrivanje nepravilnosti kože. Vjeruje se da je prva šminka bila mast za usta koju su ljudi izrađivali još davne 3400. godine prije Krista, a nedugo zatim postalo je popularno i naglašavanje očiju ugljenim štapićima za oči.

U Egiptu je šminka često bila povezana s medicinom – usta su se ukrašavala zemljom crvenicom pomiješanom s masnoćom, obrazi smokvinim sokom, kapci terakotom i crnim kajalom od ugljena za oči i obrve, a koža i kosa kanom. Osim u svrhe uljepšavanja, šminka je služila i za zaštitu od Sunca i insekata. Svo znanje o kozmetici koje su kroz godine usvojili Egipćani, proširilo se i u zemlje Sredozemlja i Azije.

Za razliku od Egipta, gdje su žene i muškarci često koristili šminku, u Grčkoj taj ‘trend’ nije bio toliko zastupljen. Na lice se je stavljao bijeli (olovo – karbonat) i crveni (živa – sulfid) pigment, a posebno su se naglašavale obrve. Prema Hipokratovim zapisima, vjeruje se da su već tada Grci koristili sredstva za uklanjanje bora i spriječavanje nastanka sunčanih pjega.

U Rimu kozmetika doživljava procvat - Rimljani počinju koristiti boje za oči, obrve, pudere i rumenila, a statusni simbol predstavljala je svijetla put zbog koje su žene često koristile razne otrove. U vrijeme Srednjeg vijeka, šminkanje doživljava svoj mračni period, gdje su žene često bile spaljivane zbog nanošenja šminke. Cijenila su se bijela lica s rumenim obrazima, a dopušteno je bilo koristiti jedino cerosit i crljenicu, te brojne sastojke koji su narušavali ljudsko zdravlje. Vrijeme baroka, kako u arhitekturi tako i u ostalim aspektima, karakterizira veliki kič. Lica su se bojala u bijelo rižinim puderom, obrazi su se šminkali u rozo, crtale su se kišobranaste obrve i srcolika usta. Uz sve navedene uspone i padove, šminkanje u konačnici postaje prihvaćeno diljem svijeta. U 19. stoljeću počinje industrijska proizvodnja kozmetičkih proizvoda, dok u 20. stoljeću nastaju prve kozmetičke kuće od kojih neke postoje još i danas.

Međutim, osim nanošenja šminke, u novije vrijeme ‘usavršavanje’ ljudskog lica i tijela dostiže novu razinu. Razvijaju se brojni računalni programi, uz pomoć kojih je moguće dodatno uređivanje ljudskog tijela na fotografijama. Thomas Knoll, student na sveučilištu u Michiganu, 1987. godine piše prvi program za prikaz slike u sivim tonovima na crno – bijelom zaslonu pod imenom „Display”. Program se kasnije razvija u jedan od najpoznatijih programa današnjice korištenih pretežito za uređivanje fotografija – „Photoshop”. Photoshop, kao i brojni drugi programi nudi razne mogućnosti koje doprinose usavršavanju slike ljudskog tijela.

Inspiriran slijedom razvijanja usavršavanja ljudskog lica, napisan je i ovaj završni rad, čija je svrha omogućiti svim korisnicima digitalno uređivanje lica u nekoliko osnovnih poteza.

2. ZADATAK I RJEŠENJE

2.1. ZADATAK

Proširena stvarnost je skup tehnologija kojima se u sliku stvarnog svijeta uključuju virtualni elementi. Pritom je važno da se poravnavanje stvarnih i virtualnih elemenata radi u trodimenzionalnom koordinatnom sustavu te da sustav radi u stvarnom vremenu. Jedan od vidova proširene stvarnosti je iscrtavanje virtualnih elemenata u slici lica, zasnovano na praćenju lica korisnika. Jedna takva primjena je virtualna šminka, što omogućava korisnicima virtualno isprobavanje raznih boja i vrsta šminke. Vaša je zadaća proučiti osnove proširene stvarnosti i sustav praćenja lica dostupan na Zavodu, te na temelju tog sustava implementirati sustav za virtualno isprobavanje šminke.

2.2. RJEŠENJE

Glavna ideja bila je koristiti “VisageTracker”, alat iz “VisageSDK API-a” u svrhu dobivanja Mesh-a obraza i usana, gdje bi se zatim promijenili materijali (boja, transparencija i effect) te time dobila šminka na licu.

3. UNITY SUSTAV

3.1. FUNKCIONALNOSTI SUSTAVA

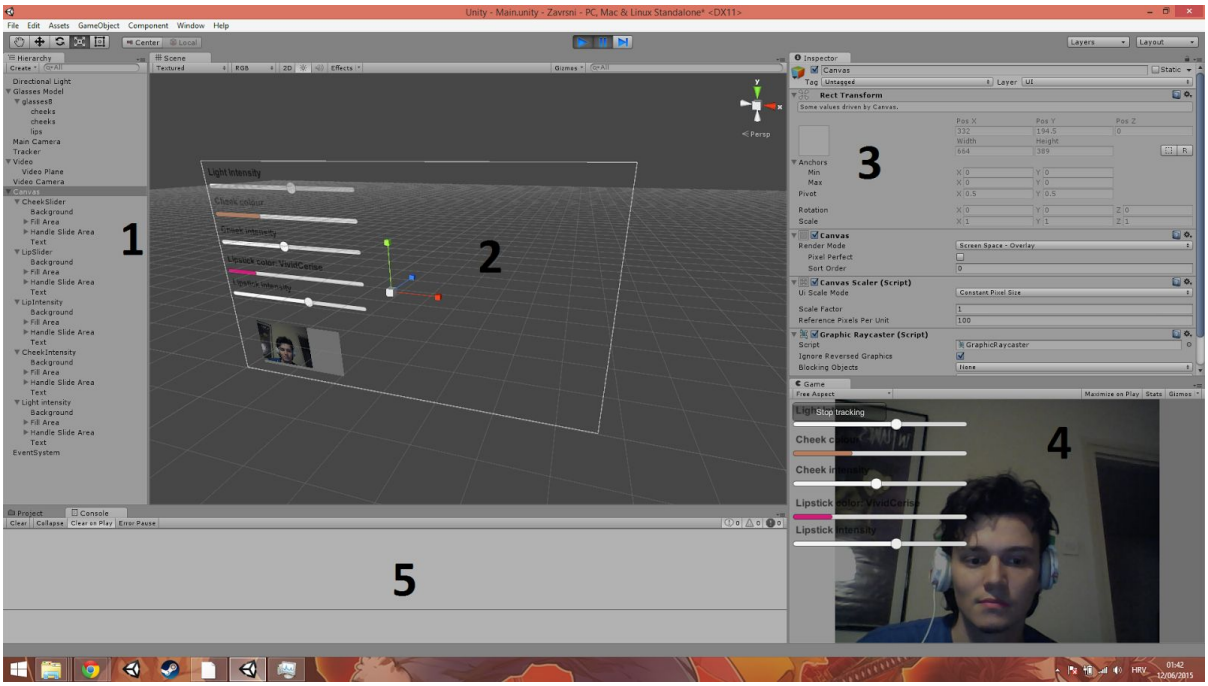
Unity je više-platformski sustav u kojem se primarno izrađuju računalne igre, a pogodan je i za simulacije. Najveća snaga Unity-a leži u njegovoj mogućnosti pokretanja projekta na raznim platformama. Tako se najveći problem grafike riješava korištenjem različih API-a za različite operacijske sustave:

- Direct3D za Windows operacijski sustav i Xbox360,
- OpenGL za Linuxe, Mac i Windowse,
- OpenGL ES za Android i IOS,
- te vlastite komercijalne API određene konzole za igrice: Nintendo Wii, Nintendo Wii-U, PlayStation 2, PlayStation 3, PlayStation 4, PlayStation Portable, PlayStation Vita, PlazStation Mobile,Windows Phone, Windows Store, Xbox One, Blackberry te sve web preglednike.

Za ovaj projekt korištena je verzija 4 Unity-a. U ovome trenutku zadnja verzija je Unity 5, ali VisageSDK nije jos migriran na novu verziju.

3.2. PREGLED

Osnovni prazni projekt sastoji se od jedne scene u kojoj je samo glavna kamera. U nju importamo custom package iz Visage SDK API-a na lokaciji
Samples/OpenGL/data/VisageTrackerUnity
Te otvorimo main scenu i dodemo ususret sa osnovnim projektom VisageTracker-a u Unity-u.



Slika 3.1.

Na slici je prikazan Unity urednik, na kojem su brojevima označeni glavni segmenti. Pod brojem ‘1’ su svi objekti hijerarhijski postavljeni u sceni, a pod brojem ‘2’ su prikazani svi objekti scene. Treći panel na popisu zove se ‘Inspector’ te su u njemu opisani najvažniji atributi trenutno označenog objekta.

U doljnjem desnom kutu slike, pod brojem ‘4’, nalazi se sam prikaz igre/simulacije kroz glavnu kameru. Posljednji broj, broj ‘5’, predstavlja konzolu u kojoj Unity sustav javlja sve pogreške i mi možemo ispisivati povratne informacije. Ovo je samo jedna od mogućnosti distribuiranja panela; oni se mogu distribuirati po preferenci u uredniku.

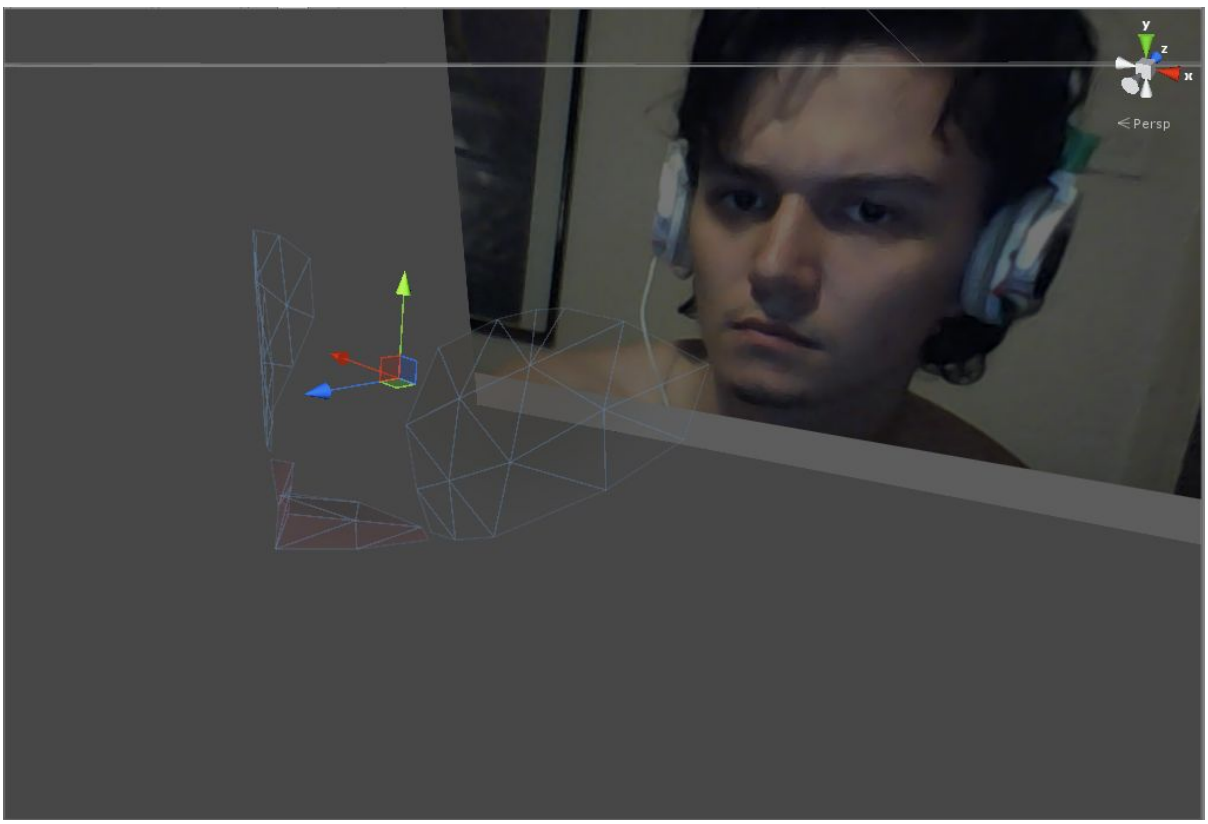
4. VISAGE TRACKER UNITY PLUGIN

4.1. OPIS

Visage Tracker Unity Plugin je C# omotač kojim možemo koristiti pokoje funkcionalnosti.

4.2. ULOGA

Preko omotača dohvaćamo podatke sa web camere koje omotač zatim obradi, sazna lokaciju lica, te zadani Mesh skalira, rotira i translatira, Mesh se postavi između glavne kamere scene i live feeda web camere. Time se dobije efekt da kamera prvo uočava šminku, a zatim sadržaj kamere.



Slika 4.1.

4.3. FUNKCIJE VISAGE TRACKER UNITZ PLUGINA

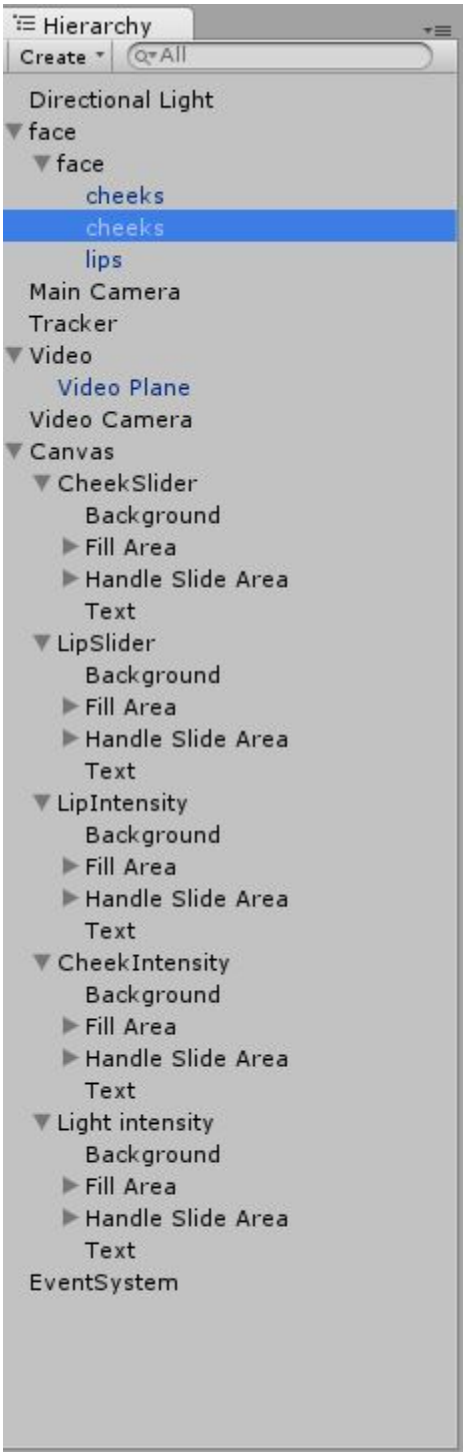
"Visage Tracker Unity Plugin" implementira određen broj funkcija, te ih prilagodi za korištenje u jeziku C# i Unity okruženju.

Neke od važnijih funkcija za ovaj rad koje su se implementirale su:

EXPORT_API void	_initTracker (char *config) Navedena funkcija inicijalizira "Tracker".
EXPORT_API void	_startTracker () Navedena funkcija pokreće "Tracker".
EXPORT_API void	_stopTracker () Navedena funkcija zaustavlja "Tracker".
EXPORT_API void	_get3DData (float *tx, float *ty, float *tz, float *rx, float *ry, float *rz, int *status) Navedena funkcija vraća trenutnu translataciju glave, rotaciju i status praćenja.
EXPORT_API bool	_getFaceModel (int *vertexNumber, float *vertices, int *triangleNumber, int *triangles, float *texCoord) Navedena funkcija vraća podatke potrebne za crtanje 3D modela lica.
EXPORT_API bool	_getGazeDirection (float *direction) Vraća smjer pogleda.
EXPORT_API bool	_getFeaturePoints2D (int number, int *groups, int *indices, float *positions) Navedena funkcija vraća buduće pozicije svojstvenih točaka u normaliziranim 2D koordinatama

5. IMPLEMENTACIJA

5.1. OBJEKTI U UNITY-U



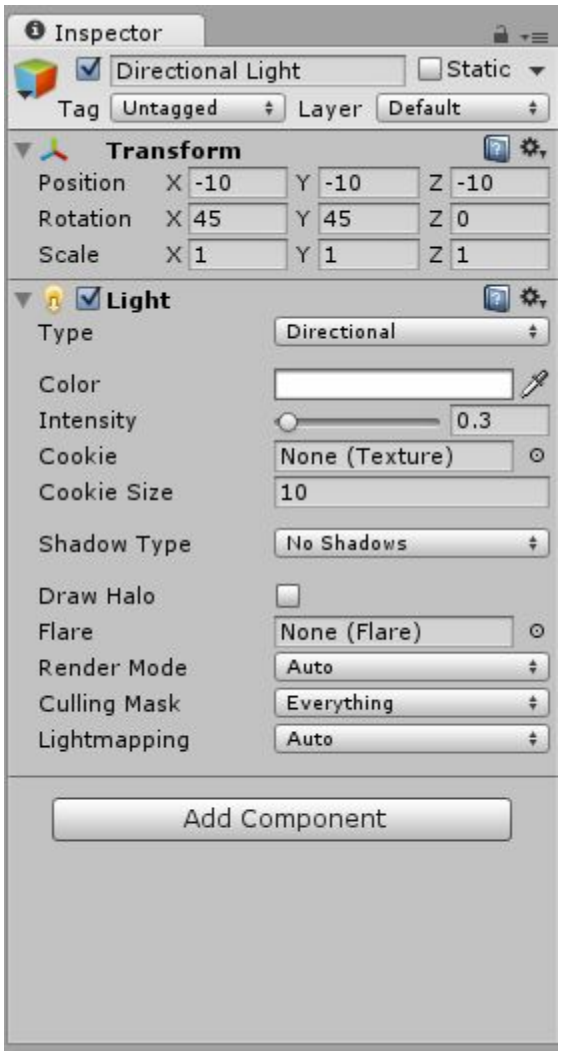
Slika 5.1.

- "Directional Light" je svijetlo koje konstanto udara sve objekte pod istim kutem. Služi za eventualnu korekciju svijetlosti na ekranu.
- "Face" objekt je 'container' u kojem su spremljeni svi Meshevi koje Visage SDK translatira, rotira i skalira.
- "Tracker" je objekt u kojem je sve u vezi "Visage SDK Unity wrapper-a" definirano, te obavlja sve transformacije nad objektom "Face".
- U "Video" objektu je spremljen zaslon koji pokazuje live feed web camere.
- "Canvas" objektom je definirano cijelo korisničko sučelje kojim korisnik komunicira sa igrom/simulacijom.
- "CheekSlider" je objekti u kojem je definiran 'slider' preko kojeg se mijenja materijal za obraze.
- "LipSlider" analogno obavlja istu funkciju kao i "CheekSlider".
- "LipIntensity" i "CheekIntensity" su također 'slider' objekti koji mjenjaju intenzitet transparencije.
- "LightIntensity" je 'slider' objekt koji omogućuje mijenjanje jačine svijetla

od “Directional Light” objekta.

- “EventSystem” je dio “Canvasa” koji nadgleda komunikaciju između korisnika i “Canvasa” tokom izvođenja igre/simulacije.

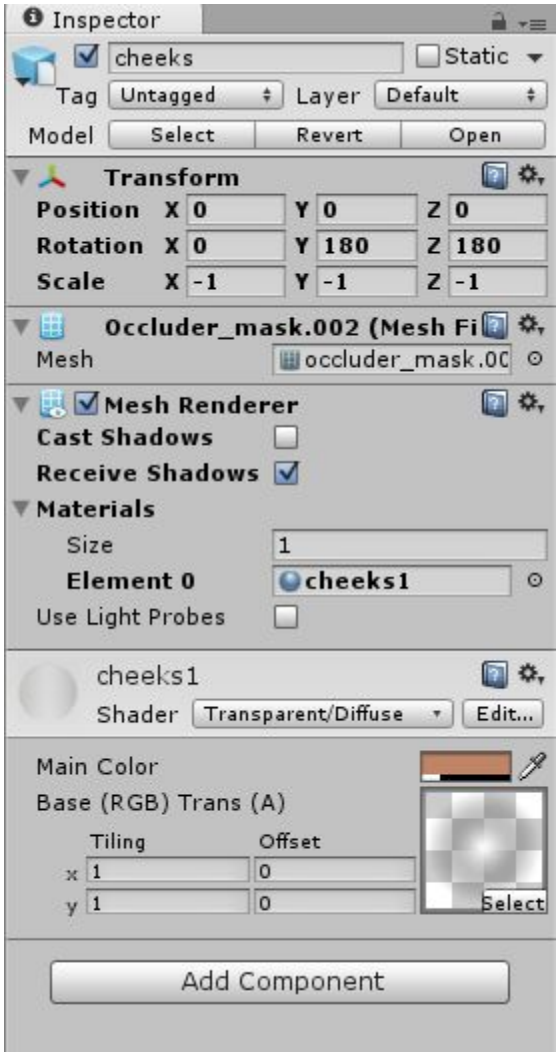
5.1.1. DIRECTIONAL LIGHT



Običan izvor svijetlosti koji konstanto sjaji u zadanom smjeru, u bijeloj boji. Jačina svijetla je konstantna u svakoj točki i služi isključivo za korekciju zaslona, kao ambijentalno svijetlo.

Slika 5.2.

5.1.2. "FACE" OBJEKT



Slika 5.3. Componente objekta "cheeks"

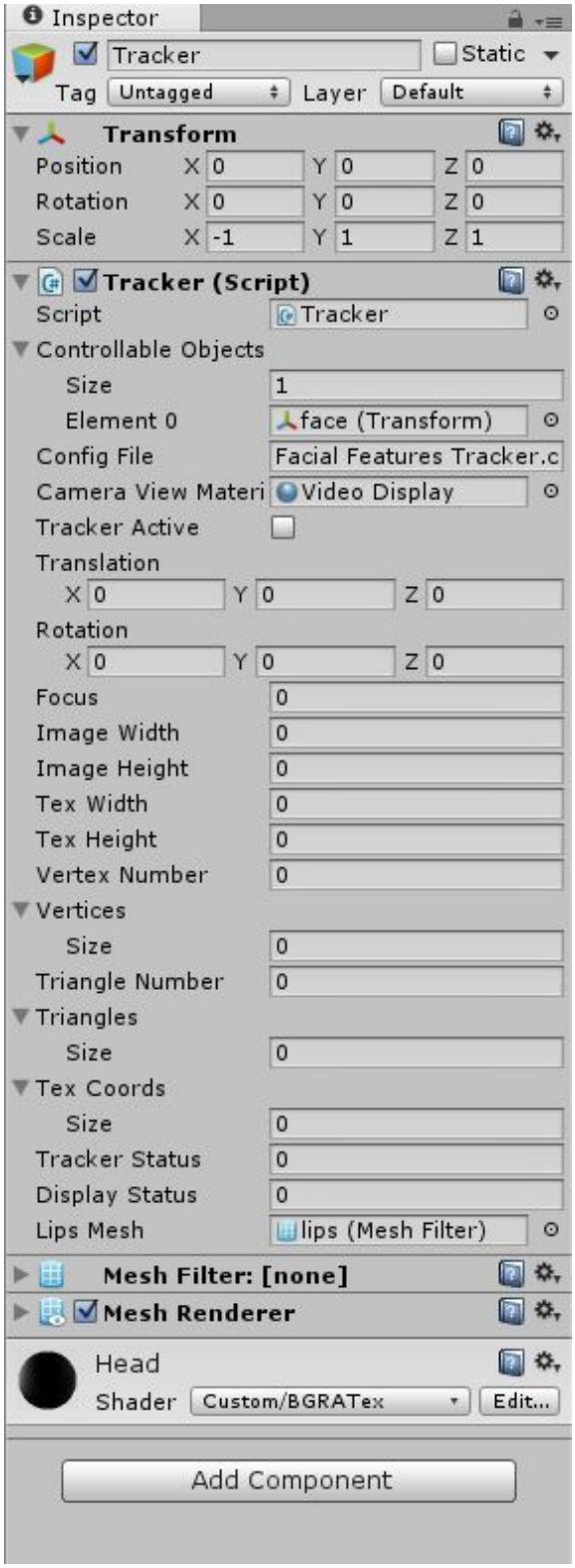
"Face" objekt je prazan objekt koji služi jedino za čuvanje Mesh objekata "lips" i "cheeks".

Na slici 5.3. prikazani su atributi objekta "cheeks".

Objekt ima za komponente:

- Transform: svoju poziciju, orijentaciju i veličinu
- Mesh objekt.
- Mesh render - u kojem je opisan njegov materijal i interakcije sa svijetlom. Budući da se radi sa šminkom, evidentno je da objekt ne baca nikakvu sjenu.
- Opis Shadera koji se koristi uz odgovarajuću teksturu za transparenciju.

5.1.3. TRACKER



Slika 5.4. Componente objekta Tracker

U "Tracker" objektu najveću važnost ima "Tracker (script)" zbog toga što se u njoj odvija cijela analiza slike. Te kao izlaz se dobiju rezultantne transformacije.

U 'Inspectoru' je za vrijeme izvođenja moguće vidjeti sve podatke koje "Tracker" svaki frame izračuna.

Skripta unosi c++ funkcije preko "VisageTrackerUnity Plugina". Primjer koda je naveden na sljedecoj stranici.

"Vertices" i "Triangles" ispisuje ukupni broj svih koordinata Mesha, ukupni broj svih trokuta te koordinate svakog vrha corespodentnog trokuta.

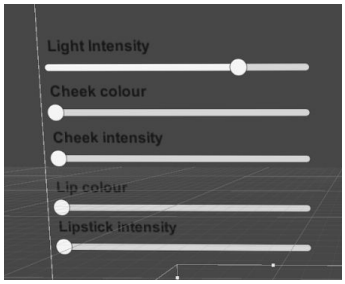
```
[DllImport("VisageTrackerUnityPlugin")]
private static extern void _initTracker(string config);

/** This functions starts the tracker.
 *
 * Implemented in VisageTrackerUnityPlugin library.
 */
[DllImport("VisageTrackerUnityPlugin")]
private static extern bool _trackFromCam();

/** This functions stops the tracker.
 *
 * Implemented in VisageTrackerUnityPlugin library.
 */
[DllImport("VisageTrackerUnityPlugin")]
private static extern void _stopTracker();

/** This functions returns tracking reults as a string.
 *
 * Implemented in VisageTrackerUnityPlugin library.
 */
[DllImport("VisageTrackerUnityPlugin")]
private static extern string _getTrackingResults();

/** This functions gets image pixel data in data byte array.
 *
 * Implemented in VisageTrackerUnityPlugin library.
*/
[DllImport("VisageTrackerUnityPlugin")]
```

```
    private static extern bool _GetFrame(
                        [MarshalAs(UnmanagedType.LPArray, SizeConst = 640 * 480 * 3)]
                        byte[] data);


    [DllImport("VisageTrackerUnityPlugin")]
    private static extern void _setFrameData(IntPtr imageData);
     /** This functions binds a texture with the given native hardware texture id through OpenGL.
     *
     * Implemented in VisageTrackerUnityPlugin library.
     */
    [DllImport("VisageTrackerUnityPlugin")]
    private static extern void _bindTexture(int texID);

    /** This functions returns the current head translation, rotation and tracking status.
     *
     * Implemented in VisageTrackerUnityPlugin library.
     */
    [DllImport("VisageTrackerUnityPlugin")]
     private static extern void _get3DData(out float tx, out float ty, out float tz, out float rx, out float ry, out float rz, out int status);

    /** This functions returns camera info.
     *
     * Implemented in VisageTrackerUnityPlugin library.
     */
    [DllImport("VisageTrackerUnityPlugin")]
    private static extern bool _getCameraInfo(out float focus, out int width, out int height);

    /** This functions returns data needed to draw 3D face model.
```

```
    *
    * Implemented in VisageTrackerUnityPlugin library.
    */
   [DllImport("VisageTrackerUnityPlugin")]
   private static extern bool _getFaceModel(
                           out int vertexNumber,
                             [MarshalAs(UnmanagedType.LPArray, SizeConst = MaxVertices *
3)]
                                 float[] vertices,
                           out int triangleNumber,
                             [MarshalAs(UnmanagedType.LPArray, SizeConst = MaxVertices *
3)]
                                 int[] triangles,
                             [MarshalAs(UnmanagedType.LPArray, SizeConst = MaxVertices *
2)]
                                 float[] texCoords);

   [DllImport("VisageTrackerUnityPlugin")]
     public static extern bool _getFeaturePoints2D(int number, int[] groups, int[] indices,
float[] positions);
```

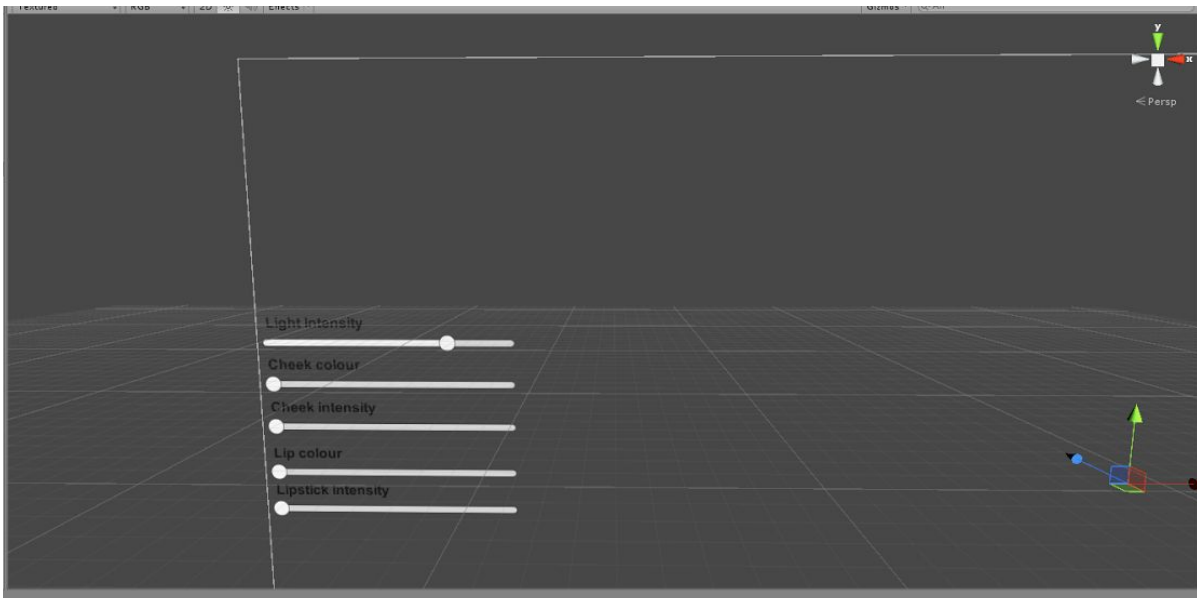
5.1.4. CANVAS

Točna veličina “Canvasa” je određena bijelim pravokutnikom u sceni (slika 5.6.). Pravokutnik predstavlja zaslon i njegova relativna pozicija u sceni je neovisna. U većini slučajeva će “Canvas” uvijek biti iznad svega.

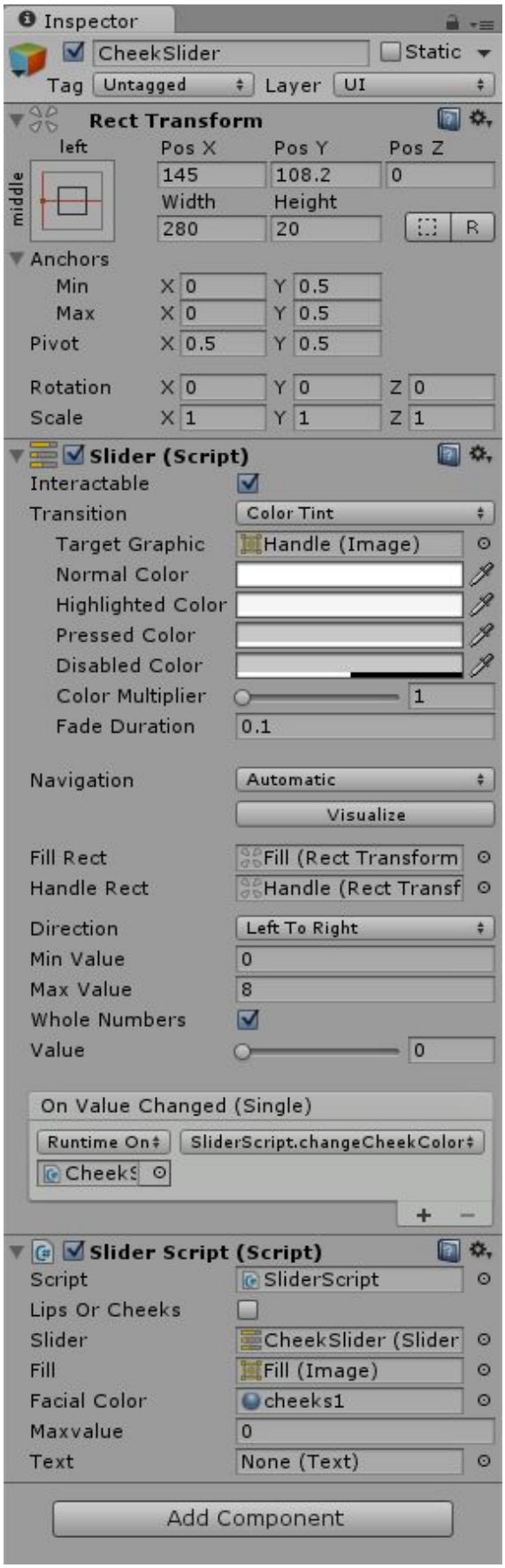


Slika 5.5.

‘Slideri’ su definirani u “Canvasu”i glavni su izvor komunikacije između korisnika i programa(slika 5.5.).



Slika 5.6. Canvas



Slika 5.7.

Pozicija 'Slidera' je apsolutna na "Canvasu" i definirana je "Rect Transform" komponentom.

U komponenti "Slider (Script)" su definirane sve boje 'Slidera', minimalna i maximalna vrijednost, te jesu li inkrementi u cijelim ili realnim brojevima.

Također, komponenta ima funkciju koju svaki puta poziva kada se dogodi 'event' pri kojem se mijenja vrijednost 'Slidera' ('On Value Changed'). Točnije poziva funkcija "changeCheekColor" iz "Slider Script (script)" komponente.

Podaci definirani u "Slider (script)" su 'defaultno' postavljeni, oni se mijenjaju na stvaranju objekta u "Slider Script (Script)" komponenti.

Kako točno izgleda funkcija "changeCheekColor()" prikazano na sljedecoj stranici

```
public void changeCheekColor()
{

   col = new Dictionary<int, string>();
   colorList = new ColorList(lipsOrCheeks);
   int i = 0;
   foreach (KeyValuePair<string, Vector3> x in colorList.colors)
   {
      col.Add(i++, x.Key);
   }

   slider.maxValue = i - 1;

   Vector3 color = colorList.colors[col[(int)slider.value]].normalized;
   fill.color = new Color(color.x, color.y, color.z);

   ColorBlock colb = new ColorBlock();
   colb.highlightedColor = fill.color;
   colb.pressedColor = fill.color;
   colb.disabledColor = new Color(200, 200, 200, 128);
   colb.colorMultiplier = 1;
   colb.fadeDuration = 0.1f;
   facialColor.color = fill.color - new Color(0, 0, 0, 0.8f);

   slider.colors = colb;
}
```

5.2. OBJEKTI U C#

U C# je dodatno definiran samo jedan razred - razred “ColorList”.

Razred isključivo ima samo konstruktor i jedan rječnik.

U konstruktoru se rječnik napuni parom ‘ključ i vrijednost’, pri čemu je ključ *String*, a vrijednost “Vector3” koja reprezentira RGB boje. Primjer jedne konfiguracije je napisan na sljedecoj stranici.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UnityEngine;
public class ColorList
{
    public Dictionary<string, Vector3> colors;
    /// <summary>
    ///  Primjer Liste Boja
    /// </summary>
    /// <param name="lipsOrCheeks">true for lips, false for cheecks</param>
    public ColorList(bool lipsOrCheeks)
        {
            colors = new Dictionary<string, Vector3>();
            if (lipsOrCheeks)
            {
                colors.Add("Barbie", new Vector3(224, 33, 138));
                colors.Add("Razzmatazz", new Vector3(227, 37, 107));
                colors.Add("FandangoPink", new Vector3(222, 82, 133));
                colors.Add("RoseBonbon", new Vector3(249, 66, 158));
                colors.Add("VividCerise", new Vector3(218, 29, 129));
                colors.Add("RasberryPink", new Vector3(226, 80, 152));
                colors.Add("BrightPink", new Vector3(255, 0, 127));
                colors.Add("BrinkPink", new Vector3(251, 96, 127));
                colors.Add("Strawberry", new Vector3(252, 90, 141));
                colors.Add("Folly", new Vector3(255, 0, 79));
                colors.Add("RedPurple", new Vector3(228, 0, 120));
                colors.Add("MexicanPink", new Vector3(228, 0, 124));
                colors.Add("DeepCerise", new Vector3(218, 50, 135));
```

```
            colors.Add("DeepPink", new Vector3(255, 20, 147));
            colors.Add("FrenchFuchsia", new Vector3(253, 63, 146));
            colors.Add("Desire", new Vector3(234, 60, 83));
            colors.Add("CerisePink", new Vector3(236, 59, 131));
            colors.Add("NeonFuchsia", new Vector3(254, 65, 100));
            colors.Add("Blush", new Vector3(222, 93, 131));
            colors.Add("UaRed", new Vector3(217, 0, 76));
        }
        else
        {
            colors.Add("Cheek1", new Vector3(246, 171, 142));
            colors.Add("Cheek2", new Vector3(178, 85, 44));
            colors.Add("Cheek3", new Vector3(208, 111, 56));
            colors.Add("Cheek4", new Vector3(239, 165, 128));
            colors.Add("Cheek5", new Vector3(213, 114, 75));
            colors.Add("Cheek6", new Vector3(244, 159, 104));
            colors.Add("Cheek7", new Vector3(236, 134, 86));
            colors.Add("Cheek8", new Vector3(219, 116, 75));
            colors.Add("Cheek9", new Vector3(208, 138, 86));
            colors.Add("Cheek10", new Vector3(183, 90, 57));
        }
    }
}
```

6. REZULTATI

Slika 6.1. Primjer bez šminke

Slika 6.2. Primjer sa šminkom

7. PRIMJEDBE

Iako pod nekim kombinacijama boja i transparencije ispadaju podosta realistični rezultati, često se zbog izvora svijetla na web camu događaju interferencije i pojavljuju sjene zbog oblika nosa, podočnjaka ili kose. Nadalje u "API Visage Tracker Unity Plugin" postoji naredba "**_getFaceModel()" s** kojom je moguće dohvatiti sve vertexe Mesha lica. U API-u bi bilo poželjno dodati opterećenje ("overload") ili novu funkciju sličnu "**_getFeaturePoints3D",** koja vraća vertexe po grupama u 3D prostoru za stvaranje novih Mesheva. Korištenjem funkcije "**_getFeaturePoints2D"** moguće je ostvariti prikaz Mesheva, ali je nemoguće dodati interferenciju sa svijetlom. Time bi bilo moguće animirane Mesheve koristiti i gledati sminku - primjerice, sa osmijehom koji pokazuje zube.

8. SAŽETAK

Proširena stvarnost je skup tehnologija kojima se u sliku stvarnog svijeta uključuju virtualni elementi. Tehnologije koje smo pritom koristili su Unity te VisageTrackerUniyPlugin. Postupak rješavanja je vrlo jednostavan: ispred slike postavimo Mesh objekt koji izgleda kao usne i obrazi te Meshu promijenimo materijal te stvorimo virtualno sminku na slici. Za računjanje pozicije Mesh objekta smo koristili funkcije iz VisageTrackerUnityPlugina te tracker objekt iz testnog primjera kojeg smo dobili uz API. Usput smo dodali koristili i korisničko sučelje Unitya za naknadno modificiranje parametara tokom simulacije.

9. ZAKLJUČAK

Kako se u Unity okruženju dosta može riješiti grafički, tako i Scene ispadaju dosta pregledne bez suvišne količine koda (pod uvjetom da se uredno postavljaju objekti).

Implementacija koristničkog sučelja bila je poprilično jednostavna i velika većina koda se je mogla ponovno upotrijebiti. Mogućnost gledanja što se događa u pozadini scene tokom izvođenja igre/simulacije je pomagala tokom analize pogrešaka.

Iako se čini da bi se trebalo koristiti komplicirane ‘shadere’, moguće je dobiti odlične rezultate šminke sa jednostavnim difuznim ‘shaderima’ sa nisko postavljenim transparencijama.

“VisageSDK” je efikasan te su testni primjeri odlični. Budući da nije bio dostupan “Unity Pro”, nije se mogo isprobati “FacialAnimationUnityPlugin” testni primjer. Daljnim razvojem plugina, te samog API, mogu se očekivati izvanredne aplikacije na bilo kojoj platformi.

10. POJMOVNIK

- C++ - Programski jezik dizajniran sa fokusom na preformanse, efikasnost i fleksibilnost. Ima osobine i mogućnosti više programskih paradigmi (prodeduralna, funkcijska, objektno orijentirana i generična)
- C# - Programski jezik koji je dizajniran za izradu raznih aplikacija koje se mogu izvršavati na Microsoft-ovoj .NET platformi. C# je jednostavan, snažan, strogo tipiziran i objektno orijentiran jezik.
    - Rječnik - Predstavlja kolekciju key Value parova.
        - key Value - Par vrijednosti untar rječnika. Prva vrijednost je unikatni identifikator, a druga su podatci.
    - DLL - Skraćenica za Dynamic-Link Library. Izvršna datoteka koja omogućava programima djeljenje koda i drugih resorsa za izvršavanje specifičnih zadataka.
    - funkcija - Niz naredbi koje obavljaju neku logičku operaciju ili set operacija.
    - konstruktor - Funkcija koja stvara objekt neke klase u memoriji i inicijalizira početno stanje tog objekta.
- string - Tip podatka koji predstavlja niz character znakova.
- int - Tip podataka koji predstavlja neki cijeli broj
- API - Skraćenica za Application Programming Interface. API je set rutina, protokola i alata za izradu softwerskih aplikacija. Izražava softversku komponentu u obliku njenih operacija, ulaza, izlaza i podslojnih tipova. Definira funkcionalnosti koje su neovisne o njihovoj implementaciji.
- Unity - Razvojna platforma za izradu multi-platform 2D i 3D igara i interaktivnih aplikacija.
    - Wrapper/ Omotač - Program ili skripta koja postavlja stadij i omogućava izvršavanje nekog drugog programa.
    - Mesh - Skup točaka, bridova i lica koji definiraju oblik 3D objekta .
    - Slider - Komponenta koja se može selektirati i koristiti za pomicanje između vrijednosti.
    - Canvas - Komponenta koja se koristi za iscrtavanje drugih elemenata na ekran nakon što se scena iscrtala.

- Event - Događaj koji uzrokuje reakciju u sustavu i započinje izvršavanje odgovarajućih funkcija.
- Vector3 - Podatkovna struktura koja predstavlja trodimenzionalni vektor.
- Directional Light - Izvor svjetla koji osvjetljava scenu pod nekim kutem i bez obzira na udaljenost.
- Materijal - Sadrži podatke koji opisuju kako se iscrtava neki mesh.
    - Transparencija - Što je transparencija veća, nakon iscrtavanja objekt će biti prozirniji i manje prekrivati dru
    - Shader - Računalni program koji kao ulaze uzima mesh, teksture i po potrebi još neke podatke i koristi ih za iscrtavanje na ekran i opisuje način iscrtavanja na ekranu..

11. LITERATURA

Sluzbeni VisageTracker Api dobiven uz API

http://docs.unity3d.com/ScriptReference/

http://docs.unity3d.com/Manual/index.html