SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 3908

# Iscrtavanje 3D grafike prema položaju glave

Jure Pajić

Zagreb, lipanj 2015.

*Umjesto ove stranice umetnite izvornik Vašeg rada.*
*Da bi ste uklonili ovu stranicu obrišite naredbu* `\izvornik`.

# SADRŽAJ

# 1. Uvod

Kod današnjih programa visoka razina interakcije s korisnikom je vrlo bitan faktor. Da bi smo to postigli potrebno je puno truda i rada na dizajnu, korisničkom sučelju, razumljivosti programa i tako dalje. Ipak ponekad je jednostavno potrebno izaći iz okvira standardne interakcije s računalnim programima i krenuti u drugom smjeru.

Time dolazimo do teme ovoga rada "Iscrtavanje 3D grafike prema položaju glave" (engl. *View dependent rendering*). U ovom radu pogledat ćemo kako se postojeća tehnologija za praćenje ljudskog lica i gesti može ugraditi u računalnu igru radi postizanja veće razine interakcije s korisnikom.

Centar ovog rada je igra "Tunnel Terror". "Tunnel Terror" je igra u kojoj se igrač nađe u tunelu koji je poplavljen krvlju i njegov je zadatak naći izlaz iz tunela. Za izlazak iz tunela igrač mora naći ključ na kojem se nalazi četveroznamenkasti kod potreban za izlazak iz tunela. Taj kod se nalazi samo na jednoj od četiri mogućih strana ključa (ispod, iznad, s lijeve i desne strane) stoga igrač mora pomicati vlastitu glavu da pogleda ključ iz druge perspektive.

Gledanje ključa iz druge perspektive je ostvareno koristeći Visage razvojni alat (više o Visage razvojnom alatu pogledajte poglavlje 2). Razvojni alat nudi knjižnice (dll) koje posjeduju funkcije koje vračaju 3D položaj i orijentaciju glave. Skupa sa razvojnim alatom dolazi *plugin* za Unity *game engine* (za više informacija o *plugin*-u pogledajte poglavlja: 2.3.1 za instalaciju *plugin*-a i 4.3 za primjenu *plugin*-a u igri).

Igra je napravljena uz pomoć programskih alata: Unity, Blender, Photoshop i Visual Studio. Preko Unity *game engine*-a ostvarena je funkcionalnost i logika igre. Unity je ujedno i služio za spajanje funkcionalnosti svih komponenata u jedan zaokruženi program. Izrada 3D modela ostvarena je preko programa Blender, a dorada/izrada tekstura uz pomoć Photoshop-a (o detaljima izrade 3D modela i tekstura pogledajte poglavlje 4.4). Za postizanje nasumičnog generiranja tunela prilikom svakog pokretanja igre korišten je program "MazeGenerator". On izrađen u programu Visual Studio koristeći programski jezik C# (više o programu "MazeGenerator" pogledajte poglavlje 4.2, a više o pojedinostima pojedinih programskih alata pogledajte poglavlje 3.

Na početnom ekranu igre "Tunnel Terror" igrač ima opciju pokretanja igra (tipka "START") i izlaska iz nje (tipka "EXIT"). Kad se pokrene igra na ekranu se pojavi broj razine na kojoj se igrač trenutno nalazi i pritiskom tipke "Enter" pokreče se izgradnja razine i igra počinje. Nakon što igrač nađe traženi četveroznamenkasti kod igrač mora pronaći kraj tunela koji je označen ljestvama oko kojih se vrti natpis "FINISH". Kad igrač dođe do kraja i pritisne tipku "Enter" na ekranu se pojavi tipkovnica pomoću koje igrač upisuje kod i ako je upisani kod točan počinje sljedeća razina.

Igra se sastoji od 4 razine (ili kata tunela) gdje je svaka razina sve veća, a kretanje igrača je sve sporije. Za detalje o elementima igra pogledajte poglavlje 4.

U nastavku ćemo proći kroz detalje svakog elementa pomoću kojeg je ostvarena funkcionalnost ovoga rada.


**Slika 1.1:** "Tunnel Terror" *banner*

# 2. Visage razvojni alat (engl. *Visage SDK*)

Visage razvojni alat ugrađuje sveobuhvatni skup tehnologija računalnog vida i animacije lica u jednostavnom za korištenje, potpuno dokumentiranom razvojnom alatu za široko područje primjene.

Glavne značajke su:

- detekcija i praćenje lica i crta lica uključujući 3D položaj glave, smjer pogleda, zatvorenost očiju potpuni 3D model glave i druge;
- sinkronizacija usana iz mikrofona ili audio datoteke u stvarnom vremenu;
- animacija lica u stvarnom vremenu pokretana sa SAPI-5 sintezom govora
- višestruke interaktivne animacije lica iz različitih izvora (praćenje lica, sinkronizacija usana, sinteza govora, datoteke, proceduralne animacije itd.);
- potpuna podrška za MPEG-4 standard animacije lica i tijela (engl. *Face and Body Animation* (FBA))

Detaljnije objašnjenje svih funkcionalnosti Visage razvojnog alata pogledajte literaturu: [5].

## 2.1. Zahtjevi sustava

Podržana razvojna okruženja su:

- Microsoft Visual Studio 2010
- Microsoft Visual Studio .NET 2008
- Microsoft Visual Studio .NET 2008 with Service Pack 1

## 2.2. API

Visage razvojni alat sastoji se od različitih knjižnica (.dlls), svaka implementira specifični skup funkcionalnosti.

Visage razvojni alat koristi knjižnice treće strane: "OpenCV", Intel JPEG knjižnica i "libXML".

U nastavku će mo ukratko proći kroz bitne funkcionalnosti pojedinih knjižnica.

### 2.2.1. Praćenje crta lica (engl. *Facial Features Tracking*)

Praćenje crta lica ostvaruje se pomoću "vsvision.dll" (*Visage VISION functions*). Knjižnica "vsvision.dll" sadrži vidno bazirane algoritme za detekciju i praćenje crta lica u slikama i video zapisima. Visage tragač prati lice i crte lica iz video sekvencama i vrača 3D poziciju glave, izraz lica, smjer pogleda, bitne točke lica i 3D teksturu modela lica. Tragač je u potpunosti prilagodljiv u smislu performansi, kvalitete, količini praćenih crta lica i pokreta te mnoge druge opcije. Često korištene konfiguracije dolaze u kompletu sa razvojnim alatom.

Glavni razredi su:

- VisageTracker2
- FaceData
- VisageTrackerObserver

---

### 2.2.2. Detektiranje crta lica (engl. *Facial Features Detection*)

Za detektiranje crta lica, kao što je prethodno rečeno, koristi se knjižnica "vsvision.dll". Razred "VisageFeaturesDectector" detektira lice i crte lica iz ulaznih slika. Rezultati (za svako detektirano lice) su 3D pozicija glave, koordinate glavnih točaka lica, (vrh nosa, brade, krajevi usana itd.) i 3D model lica. Rezultati su vraćeni u obliku jednog ili više FaceData objekata (jedan za svako detektirano lice).

Glavni razredi su:

- VisageFeaturesDetector
- FaceData

---

### 2.2.3. Sinkronizacija usana(engl. *Lip Sync*)

Sinkronizacija usana se ostvaruje pomoću knjižnice "vslipsync.dll". Knjižnica proizvodi animaciju usana sinkronizirane s prirodnim govorom analizirajući govorni zvučni signal. Ona može u stvarnom vremenu pokretati animaciju lica. Govor može biti unaprijed snimljen ili može doći direktno iz mikrofona.

Glavni razredi su:

- VisageLipSync
- LipSyncEventHandler

### 2.2.4. Vizualni tekst u govor(engl. *Visual Text-To-Speech*)

Knjižnica "vsvtts.dll" generira govornu i pripadajuću animaciju lica koristeći SAPI-5 tekst u govor (engl. *Text-To-Speach*(TTS)) engine, ili za renderiranje u letu (engl. *on-the-fly*) ili za pisanje u datoteku. Da bi se implementirala ova funkcionalnost, nužno je instalirati MS SAPI-5 razvojni alat. Za vrijeme izvođenja, nužno je imati instaliran barem jedan govorni *engine* koji je kompatibilan s SAPI-5. Razvojni alat i govorni *engine*-i su besplatno dostupni u Microsoft-u.

Glavni razredi su:

- Visagesapi5tts
- VisagesapiTtsObserver

### 2.2.5. Temeljne funkcije(engl. *Core functions*)

Izvorne funkcije sadržane unutar knjižnica "vscore.dll" i "vsanim.dll" sadrže podršku za mehaniku animacije igrača, MPEG-4 animaciju lica i tijela (engl. *Face and Body Animation* (FBA)) i podršku za integraciju prikazivanja.

#### Animacija igrača

Razredi "FAPlayer" i "FbaAction" su centar Visage razvojnog alata. Oni se koriste da istovremeno pokreću različite animacije na virtualnom liku, gdje informacije o animaciji mogu doći iz različitih izvora (datoteke, sinkronizacije usana, govorne

sinteze, proceduralne animacije itd.). Informacije o animaciji često dolaze iz prethodno spomenutih knjižnica.

Visage razvojni alat implementira visoku razinu MPEG-4 FAB funkcionalnosti preko razreda "FAPlayer". Razred od glavne aplikacije dobiva model lika nad kojem vrši animaciju. Može čitati jednu ili više animacija iz MPEG-4 FBA toka i prihvaćati druge izvore preko apstraktnoga sučelja "FBAction". Sve animacije su pokrenute i združene zajedno. Moguće je koristiti više instanci razreda "FAPlayer" radi istovremenog animiranja više virtualnih likova.

"FAPlayer" može raditi u stvarno vremenu i van vremena. Sučelje razreda "FbaAction" pruža jednostavni, a ipak moćni mehanizam za pokretanje bilo kakvih izvora animacije. Kao na primjer da bi smo postigli animaciju govora lika potrebno je povezati razred "Visagesapi5tts" s "FAPlayer" i pozvati metodu Visagesapi5tts::speak(). Sinkronizaciju usana i praćenje lica je ostvareno na isti način. Razredi "VisageLipSync" i "VisageTracker2" su implementacije razreda "FAPlayer".

Mnoge druge pomoćne funkcije su dostupne kao: automatsko praćenje pogleda za vrijeme animacije, spajanje različitih tokova, njihovo spremanje u datoteke, kao i pomoćni razred za izvoz animacija.

Glavni razredi su:

- FAPlayer
- FbaAction
- FbaFileAction
- FARenderer
- FAExporter

---

**MPEG-4 FBA podrška**

Visage razvojni alat sadrži razrede koji podržavaju kodiranje i dekodiranje MPEG-4 FAB datoteka i bitstream-ova, razrede za spremanje MPEG-4 FAB animacijskih parametara (razred "FBAPs") i pogodne funkcije za postavljanje raznih parametara. Razred "CFBAEncoder" kodira "FBAPs" u MPEG-4 bitstream koji se može zapisati u datoteku. S druge strane razred "CFBADecoder" dekodira MPEG-4 bitstream da dobije natrag parametre.

Glavni razredi su:

- CFBADecoder
- CFBAEncoder
- FBAPs, BAPs, FAPs, LLFAPs, FAP1, FAP2
- CodingParameters
- FDP

---

**Sučelje prikazivača**

Razred "VisageCharModel" pruža sučelje za pristup i manipulaciju modela lika. U OpenGl verziji Visage razvojnog alata, razred "AFM" je implementacija razreda "VisageCharModel" specifična za OpenGl.

Glavni razredi su:

- AFM

---

## 2.3. *Plugin* za Unity

*Plugin* "VisageTrackerUnityPlugin" objedinjuje pojedine funkcionalnosti razreda "VisageTracker2" za korištenje u jezicima kao što su C# i Java. Više o primjeni *plugin*-a u poglavlju 4.3.

Pojedine bitne metode *plugin*-a:

- _initTracker()
- _trackFromCam()
- _stopTracker()
- _get3DData()

---

### 2.3.1. Instalacija *plugin*-a

Za korištenje *plugin*-a potrebno je imati program Unity (verzija 4.x).

Da bi izradili i pripremili projekt u Unity-u da može koristiti plugin potrebno je obaviti sljedeće korake:

- 1. Pokreni novi projekt u Unity-u (ili koristi postojeći).
- 2. Aplikacija ovisi o knjižnicama razvojnog alata te je potrebno u početni direktorij projekta dodati ove datoteke:
  - opencv_calib3d243.dll
  - opencv_core243.dll
  - opencv_features2d243.dll
  - opencv_flann243.dll
  - opencv_highgui243.dll
  - opencv_imgproc243.dll
  - opencv_legacy243.dll
  - opencv_ml243.dll
  - opencv_video243.dll
  - VisageTrackerUnityPlugin.dll
  - vscore.dll
  - vsvision.dll
  - iconv.dll
  - libxml2.dll
  - zlib1.dll
- 3. Napravi "StreamingAssets" direktorij u Assets direktoriju.
- 4. Kopiraj sljedeće datoteke iz "visageSDK/Samples/OpenGL/data/FaceTracker2" direktorija:
  - bdtsdata direktorij
  - candide3.wfm
  - candide3.fdp
  - konfiguracijska datoteka (npr. Head Tracker.cfg)

Nakon ovoga Unity projekt je spreman koristi plugin. Da bi se pozivale funkcije Visage razvojnog alata potrebno je uvesti metode iz knjižnice "VisageTrackerUnityPlugin.dll", a o tome ćemo detaljnije pričati u poglavlju 4.3.

# 3. Programski alati

Igra "Tunnel Terror" je izrađena kombinacijom različitih programskih alata. Programski alat Unity je korišten za izradu logike i funkcionalnosti igre, Blender za izradu 3D modela, Photoshop za doradu i izradu pojedinih tekstura te Visual Studio 2013 za izradu programa MazeGenerator (više o njemu u 4.2). Sada ćemo ukratko proći kroz funkcionalnosti programskih alata koje su bile bitne za izradu ovoga rada, a o detaljima elemenata igre ćemo pričati u poglavlju 4.

## 3.1. Unity

Unity je više platformni *game engine* razvijen od Unity Technologies i korišten je za izradu video igara za PC, konzole, mobilne uređaje i web stranice. Unity podržava skripte jezike: C#, JavaScript i Boo (literatura: [3]).

Unity ima sučelje koje je vrlo intuitivno i lagano za korištenje. Izgled sučelja je prilagodljiv, a njegove glavne komponente su: "Project", "Hierarchy", "Inspector", "Scene" i "Game".

**"Project"**

Unutar "Project" komponente smješteni su svi resursi koje koristi igra (zvukovi, teksture, modeli itd.). Resurse možemo organizirati unutar direktorija radi bolje preglednosti i same organizacije. Vrlo je korisno što Unity automatski obavješćuje sve objekte u slučaju pomicanja resursa iz jedne lokacije u drugu. Time se ne moramo brinuti o samoj fizičkoj lokaciji resursa.

**"Hierarchy"**

Unutar prozora "Hierarchy" smješteni su svi objekti koji se trenutno nalaze u sceni igre.

**"Inspector"**

"Inspector" nam nudi detalje informacije o objektima koje odaberemo. Također unutar "Inspector"-a možemo uređivati podatke o objektu te ujedno i dodavati nove komponente kao što su teksture, fizičke osobine (npr. gravitacija), osvjetljenja, skripte koje pridonose dinamici objekata i tako dalje.

**"Scene"**

Komponenta "Scene" prikazuje izgled igre i unutar nje možemo vršiti interakciju sa objektima igre (pomicanje, postavljanje tekstura itd.).

**"Game"**

Prozor "Game" nam prikazuje kako će točno igra izgledati iz pozicije igrača.

Dinamiku objekata igre postižemo pisanjem skripti koje definiraju ponašanje objekta. Da bi skripta imala utjecaja na objekt ona se mora dodati objektu kao komponenta unutar "Inspector"-a. Za pisanje skripti Unity na nudi IDE Mono Developer.

## 3.2. Blender

Blender je profesionalni besplatni 3D računalno grafički program otvorenog koda koji se koristi za izradu animiranih filmova, vizualnih efekata, umjetničkih radova, 3D modela za ispis, interaktivne 3D aplikacije i video igre. Blender-ove mogućnosti uključuju 3D modeliranje, UV odmatanje, restersku grafiku, fluidni i dimni simulator, simulacija čestica itd. Uz mogućnosti modeliranja uključuje i ugrađeni *game engine*. (Literatura: [1])

Alati Blender-a koji su bili posebice bitni pri izradi ovog rada su: "Object Mode", "Edit Mode", modifikatori ("Boolean", "Array" i "WireFrame") te "UV unwrapping".

**"Object Mode"**

Preko "Object Mode" upravljamo s objektom kao cjelinom te ga možemo: pomicati, rotirati, skalirati, umnožavati ili ga spajati s drugim objektima te mnoge druge opcije.

#### "Edit Mode"

"Edit Mode" nam služi da možemo vršiti interakciju s pojedinim vrhovima, bridovima ili stranicama koje pripadaju jednom te istom objektu. Bitno je napomenuti da unutar "Edit Mode" nije moguće vršiti interakciju s elementima koji nisu dio jednog te istog objekta.

#### "Boolean", "Array" i "WireFrame" modifikatori

"Boolean", "Array" i "WireFrame" modifikatori su skripte napisane u Pythonu i služe za vrlo lagano modificiranje objekata. Više o njihovoj funkcionalnosti u poglavlju 4.4.

#### "UV unwrapping"

"UV unwrapping" je vrlo bitna mogućnost Blender-a i služi za preslikavanje teksture na površinu objekta. Detaljnije o ovoj funkcionalnosti u poglavlju 4.4.

## 3.3. Photoshop

Adobe Photoshop je rasterski grafički uređivač dostupan za Windows i OS X operacijske sustave. (Literatura: [2])

Photoshop nudi golemu količinu funkcionalnosti za obradu i izradu slika. Pomoću alata kao što su "History Brush", "Magic Laso", "Magic Eraser" te samo slojevi ("Layers") veoma olakšavaju obradu slika.

## 3.4. Visual Studio 2013

Visual Studio je Microsoft-ov ugrađena razvojna okolina (engl. *Integrated Development Environment* IDE). Visual Studio podržava jezike: C++, C#, F#. Podržava i druge jezike kao što su Python i Ruby. (Literatura: [4])

U ovom radu Visual Studio je korišten za izradu programa "MazeGenerator" u programskom jeziku C#. Više o programu "MazeGenerator" pogledajte poglavlje 4.2.

# 4. Elementi igre "Tunnel Terror"

Funkcionalnost igre Tunnel Terror je ostvarena u kombinaciji različitih komponenti. Komponente su: Unity *game engine* ( logika, funkcionalnost igre i spajanje funkcionalnosti ostalih alata), MazeGenerator koji se poziva iz Unity skripte, Visage razvojni alat za praćenje pokreta glave te grafički modeli i teksture.

## 4.1. Unity komponente

Funkcionalnost igara u Unity-u se ostvaruje preko izrade scena i pisanjem skripti koje određuju logiku i dinamiku igre. Svaka scena je zasebni zaokruženi dio igre, kao na primjer: početni ekran, menu, prva mapa igre i tako dalje.

Ovdje se koriste 5 scena: Menu, Builder, LevelMessage, Level i EndScreen. U nastavku ćemo detaljno razmotriti svaku scenu te njezine pod komponente.

### 4.1.1. Logička struktura igre

Prije nego li krenemo na opis scena pogledajmo kakva je općenita logička struktura igre.

Kako je u prethodnom poglavlju rečeno logika igre se ostvaruje putem pisanja skripti. Kao središte cijele igre koristi se razred "GlobalClass". On u sebi sadrži informacije o kojima ovise svi ostali objekti u igri (npr. koji je kat, da li je igra zaustavljena, trenutni položaj igrača itd.). Također svaka komunikacija između dva objekta je ostvarena preko statičkih varijabli i metoda razreda "GlobalClass".

Kad se pokrene igra (scena "Menu") i pritisne tipka "START" elementi razreda "GlobalClass" se postave na početne vrijednosti i pokreče se scena "Builder". U toj sceni provjerava se trenutno stanje razreda "GlobalClass" i na temelji toga određuju se parametri za sljedeću razinu i pokreće se scena "LevelMessage". U ovoj sceni prikazuju se podaci o sljedećoj razini. Nakon što se pritisne tipka "Enter" pokreće se scena "Level". U ovoj sceni započinje igranje igre. Nakon što se završi trenutna razina

ponovno se pokreće scena "Builder". Ovaj proces se ponavlja sve dok igrač ne dođe do zadnje razine. Tada se iz scene "Builder" poziva scena "EndScreen" u kojoj igrač može zatvoriti igru ili se vratiti na početnu scenu "Menu".

U nastavku ćemo detaljnije pogledati kako je ostvarena prethodno opisan tijek događaja.

### 4.1.2. Scena: Menu

Scena "Menu" je prva scena koja se pokreče prilikom pokretanja igre. Ovdje igrač ima opciju započinjanja igre ili da izađe iz nje. Ova scena se sastoji od pozadinske slike, naslova, dvije tipke, komponente osvjetljenja te pozadinski zvuk.


**Slika 4.1:** Menu igre

#### Pozadina

Pozadina je ostvarena korištenjem slika zida, koja se ujedno koristi kao i tekstura tunela u igri. Slika je postavljena na tri mjesta: u sredini, s lijeve i desne strane, radi postizanja kompatibilnosti s uskim i veoma širokim ekranima.

#### Naslov

Izgled naslova je postignut korištenjem teksture krvi preslikane na tekst. Taj proces je ostvaren pomoću photoshop-a. Radi postizanja "horror" efekta, na pažljivo odabranim

područjima ispred naslova su dodane komponente osvjetljenja (Unity komponenta: "Point Light").

#### Tipke

Tipke "START" i "EXIT" su ostvareni preko skripte koja koristi dvije komponente "UI skin" i zvuk pritiska tipki. "UI skin" je skinut sa Unity "Asset Store"-a i kako "UI skin" definira drugačiji izgled UI komponenata za različite vrste interakcija (*mouseover*, *hover*, *click* itd.) postignuta je veća razina interakcije. Ujedno pritisak na svaki botun generira zvuk pritiska tipke radi povratne informacije. Tipka "START" započinje scenu "Builder" a "EXIT" izlazi iz igre.

#### Pozadinski zvuk

Pozadinski zvuk je ostvaren korištenjem Unity komponente "Audio Source". Ta komponenta svira jedan te isti zvuk ispočetka dok se god igrač nalazi u sceni "Menu".

### 4.1.3. Scena: Builder

Scena "Builder" se prilikom pritiska tipke "START" i nakon svakog završetka scene "Level". Scena "Builder" se sastoji od pozadine, natpisa "Loading...", osvjetljenja i objekta koji vrši logiku postavljanja parametara.

#### Pozadina

Slika pozadine je u potpunosti jednako ostvarena kao i kod scene "Menu". U sredini ispred pozadine nalazi ne natpis "Loading..." pokraj kojeg je postavljeno osvjetljenje ("Point Light"). Izgled cijele ove scene će ostati vidljiv dok se ne pokrene sljedeća scena ("LevelMessage"), čime je ostvaren efekt ekrana učitavanja (engl. loading screen).

#### Objekt "LevelBuilder"

Objekt "LevelBuilder" na temelju stanja razreda "GlobalClass" postavlja parametre za sljedeću razinu. Ta funkcionalnost je ostvarena na način da "LevelBuilder" u svom konstruktoru povećava varijablu "floor" za 1 i ako je ona manja ili jednaka 4 objekt postavlja parametre za sljedeću razinu i poziva scenu "level". Inače pokreće se scena "EndScreen".

### 4.1.4. Scena: LevelMessage

Scenu "LevelMessage" se pokreće nakon scene "Builder". Scena se sastoji od pozadine, dva teksta, osvjetljenja i objekta postavlja natpis. Jedan tekst prikazuje informacije o stranici a drugi poruku "Press Enter to continue.".


**Slika 4.2:** Scena LevelMessage.

#### Pozadina

Slika pozadine je u potpunosti jednako ostvarena kao i kod scene "Builder". U sredini ispred pozadine nalazi ne natpis (koji je drugačiji za svaku razinu) pokraj kojeg je postavljeno osvjetljenje ("Point Light").

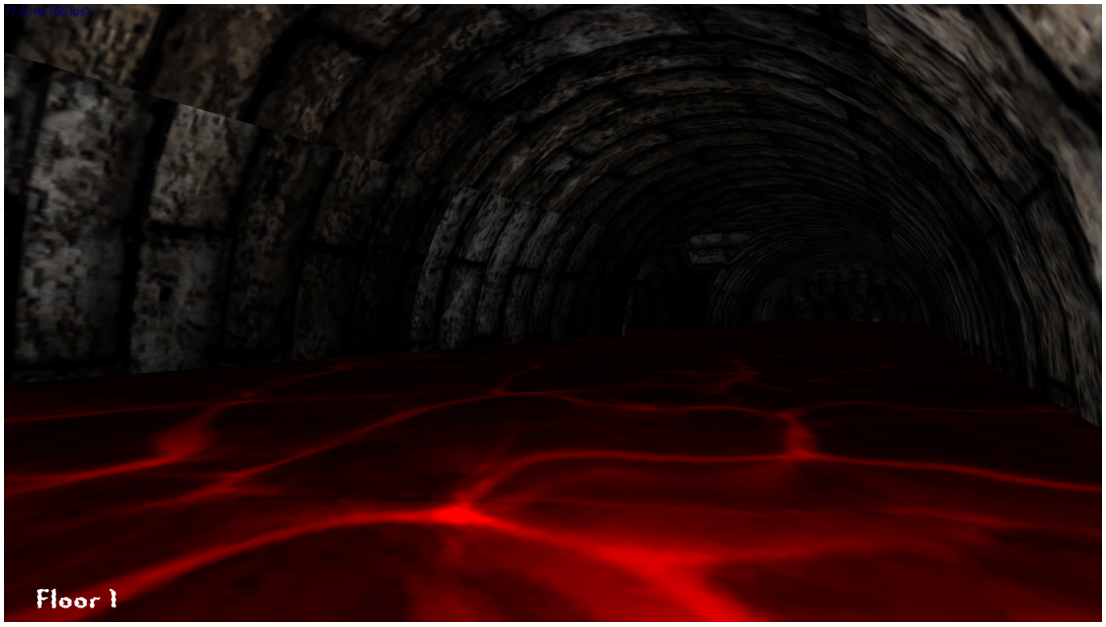
#### Objekt "Message"

Objekt "Message" ovisno o tome koja je razina postavlja drugačiji natpis na ekran. Natpisi su "Level 1", "Level 2" itd. Kada korisnik pritisne "Enter" natpis se promjeni u "Loading..." (čime je ponovno postignut ekran učitavanja) te se pokrene scena "Level".

### 4.1.5. Scena: Level

Scena "Level" je središnja scena u cijeloj igri. Ova se pokreče iz scene "Level-Message". Na samom početku objekt "Generator" generira: nasumično postavljene dijelove tunela, poziciju ključa i igrača. Ostale komponente koje su sadržane u ovoj

sceni su "InspectingTunnelCube", "Finish", "Keypad", "Pause Canvas", "VisageTracker" i pozadinski zvuk.




**Slika 4.3:** Izgled scene "Level".

### Generator

Komponenta generator služi za generiranje nasumičnih tunela te postavljanje ključa, kraja i igrača na njihove pozicije. Generator to ostvaruje pozivanjem vanjskog programa "MazeGenerator" (više o njemu u poglavlju 4.2) te čitanjem konfiguracijske datoteke labirinta koju generira "MazeGenerator".

Generator na početku iz razreda "GlobalClass" uzima vrijednost za veličinu labirinta. Nakon toga pozove program "MazeGenerator" s argumentom o veličini labirinta. Kada se labirint izgenerira (program "MazeGenerator" je završio s radom) generator podatke konfiguracijske datoteke spremi u svoju memoriju te je izbiše s računala. Na temelju pročitane konfiguracije se izgenerira labirint. Više o konfiguracijskoj datoteci u poglavlju 4.2.

Generator za generiranje ključa i tunela (labirinta) koristi unaprijed napravljene 3D modele ključa i tunela o kojima će biti više govora u poglavlju 4.4.1.

### Player

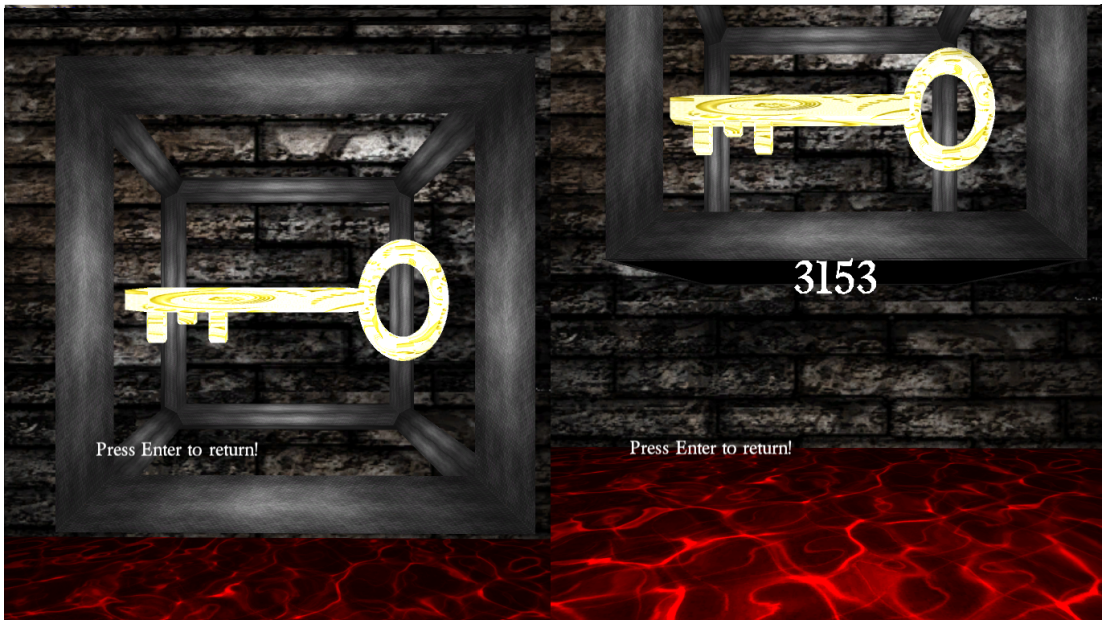
Model "Player" (igrač) je komponenta s kojom igrač upravlja. S komponentom "Player" se može upravljati na dva različita načina: tipkovnicom i mišem ili pokretom

glave. Kada se igrač nalazi u tuneli tada se s modelom "Player" upravlja s tipkovnicom i mišem, a kada se vrši pregledavanje ključa (o čemu će više biti govora kasnije u ovom poglavlju) tada se s modelom "Player" upravlja pomicanjem glave.

Pregledavanje ključa se radi tako da igrač nađe ključ u labirintu i pritisne tipki "Enter" da pregleda ključ. Nakon što je pritisnute tipka "Enter" igrač je "teleportiran" na lokaciju gdje se nalazi drugi 3D model ključa i kontrole se prebace s tipkovnice i miša na pokrete glave. Nakon što se igrač vrati iz pregledavanja, on je vraćen na staru poziciju.

### InspectingTunnelCube

"InspectingTunnelCube" je prazna kocka s teksturom zida u čijoj sredini se nalazi 3D model ključa. Kada se igrač "teleportira" na ovu lokaciju on je smješten ispred ključa okrenut prema njemu. Tada igrač pokretima glave mijenja poziciju modela "Player" čime se postiže to da iz različite pozicije glave se dobiva različita slika na ekranu. Time je postignut efekt "gledanja kroz prozor" (više o mehanici praćenja glavu u poglavlju 4.3).




**Slika 4.4:** Pogled na ključ iz različitih pozicija.

Unutar "InspectingTunnelCube" nalaze se 4 natpisa na kojima se može nalaziti četveroznamenkasti kod koji je potreban za prijelaz na sljedeću razinu. Natpisi se nalazu ispod, iznad, s lijeve i desne strane ključa. Četveroznamenkasti Kod se nalazi u razredu "GlobalClass" i on se generira u sceni "Builder". Odabir na kojem će se natpisu

pokazati ključ se određuje jednostavnim traženjem ostatka pri dijeljenju s 4. Za rezultat 0 kod se prikazuje s gornje strane, za 1 s donje, za 2 s lijeve i za 3 s desne strane.

Dok je igrač u "InspectingTunnelCube" na ekranu se nalazi tekst "Press Enter to return!" što znači da pritiskom tipke "Enter" igrač se vrača na poziciju gdje je bio prije.

### Key

Model "Key" (ključ) je 3D model napravljen u "Blender"-u (više o detaljima izrade u poglavlju 4.4.1). Njega na nasumičnu poziciju postavlja generator. Kada igrač dođe u dodir s modelom ključa tada se u razred "GlobalClass" postavi vrijednost "bool key" na "true". Time se obavješćuje ostale objekte da je igrač došao do ključa. Tada igrač može pregledavati ključ zbog potrebnog koda za prijelaz igre.

### Finish

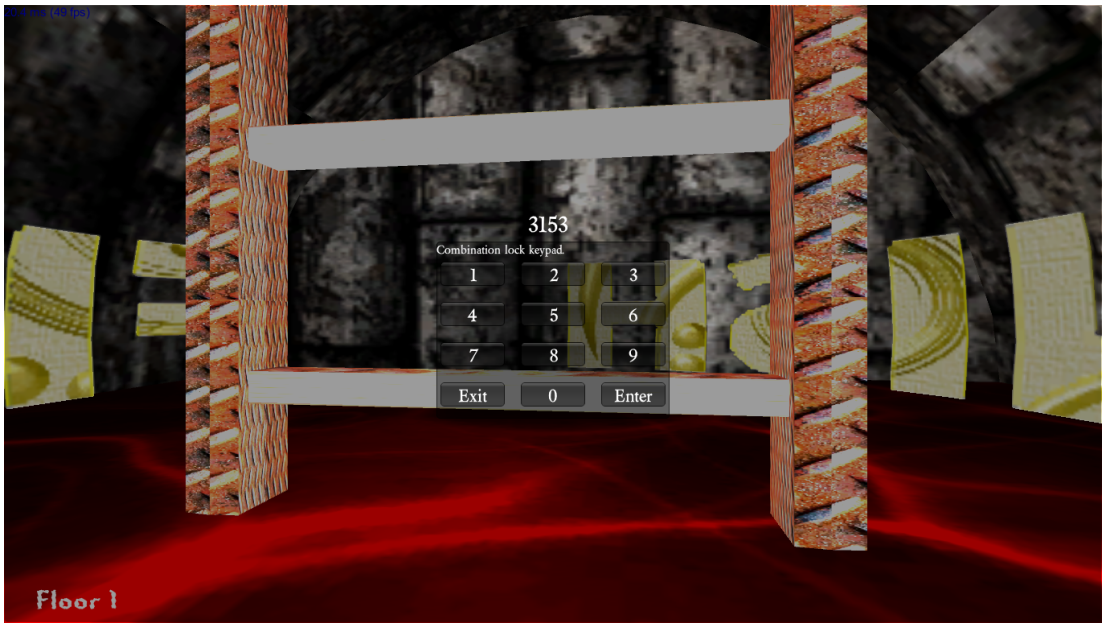
Model "Finish" je sfera na kojoj se nalazi transparenta tekstura na kojoj piše "FINISH". Njega postavlja generator na poziciju kraja. Kada igrač dođe u dodir s modelom "Finish" tada se u razred "GlobalClass" postavi vrijednost "bool atFinish" na "true". Nakon toga pojavi se tekst na ekranu "There is combination lock on the ceiling. Press Enter to enter a code!". Kada igrač pritisne "Enter" na ekranu se pojavi model "Keypad" pomoću kojeg igrač može upisati kod za prijelaz razine.


**Slika 4.5:** Izgled kraja tunela.

### Keypad i KeypadListener

Komponenta "KeypadListener" prati da li se igrač nalazi kod kraja i da li je pritisnuta tipka "Enter". Ako su ispunjena oba uvjeta tada "KeypadListener" stvara "Keypad". Prilikom stvaranja u razred "GlobalClass" postavlja se vrijednost varijable "bool playerMovement" na "false" i "bool atKeypad" na "true".

"Keypad" se sastoji od 12 tipaka: tipke od 0 do 9, "EXIT" i "ENTER". Za svaku tipki je korišten jedan te isti "UI skin" kao i kod scene "Menu" i svaka tipka proizvodi isti zvuk kada se pritisne. Razlika je tipka "ENTER" koja proizvodi drugačiji zvuk ako je uneseni kod pogrešan.




**Slika 4.6:** Izgled tipkovnice.

### Pause i PauseListener

Komponenta "PauseListener" čeka dok igrač ne pritisne tipku "Escape". Kada je tipka "Escape" pritisnuta "PauseListener" u razredu "GlobalClass" postavlja vrijednost varijable "bool pause" na "true". Uz to "Pause" kanvas se postavi da bude vidljiv i stvori se tipka na ekranu "EXIT TO TITLE" pomoću koje korisnik može odustati od igranja igre.

### VisageTracker

Objekt "Visagetracker" pokreče praćenje glave. Sve informacije o položaju glave objekt sprema u razred "GlobalClass" tako da budu vidljive ostalim objektima. Više o

implementaciji ove funkcionalnosti pogledajte u poglavlju 4.3.

#### Pozadinski zvuk

Pozadinski zvuk je ostvaren korištenjem Unity komponente "Audio Source". Ta komponenta svira jedan de isti zvuk ispočetka dok se god igrač nalazi u sceni "Level".

### 4.1.6. Scena: EndScreen

Scena "EndScreen" se poziva iz scene "Builder" ako je igrač prešao sve razine igre. Scena se sastoji od pozadine, dviju tipki "MENU" i "EXIT", osvjetljenja i natpisa "THE END".


**Slika 4.7:** Kraj igre.

#### Pozadina

Pozadinska slika natpis i osvjetljenje je ostvareno jednako kao i kod scene "Builder".

#### Tipke

Scena sadrži dvije tipke "MENU" koji vraća igrača natrag na scenu "Menu" i "EXIT" koja poziva metodu za gašenje igre. Tipke koriste jednak dizajn i jednake efekte kao i ostale tipke u igri.

## 4.2. MazeGenerator

Komponenta MazeGenerator je program nepisan u programskom jeziku C#. On se ponaša kao crna kutija koja na svoj ulaz prima cijeli broj veći od nule i na temelju tog parametra izgenerira datoteku "maze.stf" koja u sebi sadrži definiciju labirinta. Igra pričeka da generator završi sa svojim poslom te nakon toga na temelju definicije labirinta izgenerira tunele. U konačnici nakon što su tuneli postavljeni datoteka "maze.stf" se briše s računala.

Svaki redak datoteke maze.stf ima oblik:

<koordinataX>,<koordinataY>,<indeksGrafičkogElementa>,<rotacija>

a samo jedan redak je oblika:

<koordinataX>,<koordinataY>,<indeksGrafičkogElementa>,<rotacija>,1

koji označava gdje se nalazi ključ u tunelu(labirintu).

MazeGenerator se sastoji od razreda: Maze, MazePart, MazeTemplate, Coordinate te početni razred Program.

### 4.2.1. Općenito o algoritmu

Pseudo kod generiranja labirinta je sljedeći:

- učitaj zadanu veličinu
- postavi nasumično odabrani start objekt na početnu koordinatu
- provjeri moguće koordinate grananja te ih postavi u red
- dok je trenutna veličina manja od zadane ponavljaj
- uzmi koordinatu s vrha reda
- provjeri da li već postoji objekt na tom mjestu
- ako je istina vrati se početak petlje
- nađi informacije o susjedima
- na temelju dobivenih informacija postavi novi nasumični objekt
- provjeri nove koordinate grananje te ih stavi u red
- vrati se na početak petlje
- dok postoje koordinate u redu ponavljaj
- uzmi koordinatu s vrha reda
- provjeri da li već postoji objekt na tom mjestu
- ako je istina vrati se početak petlje
- nađi informacije o susjedima

- doradi dobivene informacije tako da novi objekt ne stvara nove koordinate
- provjeri da li je postavljen krajnji objekt (*finish*)
- ako istina postavi kraj
- vrati se na početak petlje
- na temelju dobivenih informacija postavi novi objekt
- vrati se na početak petlje
- nasumično odaberi područje ključa
- kraj

---

Na temelju ovog algoritma vidljivo je da postoje 3 skupa različitih objekata: početni objekti (*startTemplates*), obični(*regularTemplates*) te krajnji(*finishTemplates*). Ovi objekti se nasumično odabiru te kao slagalice (na temelju danih ograničenja i uvjeta) se postave na dane pozicije.

U sljedećim poglavljima če mo detaljno promotriti kako je ovaj algoritam ostvaren.

### 4.2.2. Razred: Program

Razred Program je početna točka u programu "MazeGenerator". Funkcija "main" čita prvi argument te poziva metodu "buildMaze" objekta tipa Maze s tim argumentom. Nakon toga poziva se metoda "outputFile" nad tim objektom i program završava s radom.

### 4.2.3. Razred: Maze

Ovaj razred je središte cijelog algoritma. Njegovo sučelje je:

---

- buildMaze(int size):void
- showMaze():void
- outputFile(string path):void
- pictureOutput(string pictureName):void
- clear():void

---

#### Metoda buildMaze

Metoda "buildMaze" implementira prethodno objašnjeni algoritam. U gornjem algoritmu nisu navedeni rubni uvjet kao što je prijevremeno zatvaranje algoritma (zbog

nasumičnosti). Metoda se brine da u tom slučaju se algoritam pokrene iz početka te time garantira da će se labirint uvijek izgraditi.

#### Metoda showMaze

Metoda "showMaze" prikazuje strukturu labirinta unutar konzole. Ova funkcionalnost služi radi debugiranja.

#### Metoda outputFile

Datoteka koju koristi igra se generira pomoću "outputFile".

#### Metoda pictureOutput

Korištenjem "pictureOutput" dobiva se strukturu labirinta u obliku slike (format png).

#### Metoda clear

Konačno ako se želi izgraditi novi labirint bitno je prethodno pozvati metodu "clear" jer "buildMaze" neće automatski izbrisati prethodno stvoreni labirint.

### 4.2.4. Razred: MazeTemplates

MazeTemplates nudi informacije o svim objektima koji se koriste pri generiranju labirinta.

Njegovo sučelje je:

- getTemplate(HashSet<int> mustHave, HashSet<int> musntHave, int type):MazePart
- getTemplate(int index, int type):MazePart
- showAllTemplates():void
- showTemplate(int index, int type):void
- showDirections(int index, int type):void

#### Metoda getTemplate

Za rad ovog algoritma bitna je metoda getTemplate. U HashSet-ovima "mustHave" i "musntHave" zapisane su informacije koje objekt mora zadovoljavati. Na temelju tih informacija nasumično se odabire jedan od kandidata koji zadovoljava te uvjete. Kada

se odabere index objekta on se vrati preko "getTemplate(int index, int type)" gdje se objekt tipa MazePart inicijalizira.

**Metode showAllTempltes, showTemplate, showDirections**

Metode "showAllTemplates", "showTemplate" i "showDirections" služe samo radi debugiranja i korisne su za olakšanu daljnju nadogradnju.

### 4.2.5. Razred: MazePart

MazePart smo već spomenuli u prethodnom poglavlju. Ovo je vrlo jednostavan razred koji u sebi sadrže sve potrebne informacije o objektu: tip, indeks, koordinata, rotacija i da li se na njemu nalazi ključ.

### 4.2.6. Razred: Coordinate

Ovaj razred implementira sučelje IEquatable<Coordinate> koji omogućuje lagano uspoređivanje koordinate. Uz to posjeduje dvije varijable x i y koje spremaju informacije o koordinati.

## 4.3. Visage Tracker

"Visage Tracker" je centar ovog završnog rada. Pomoću ove komponente igra dobiva informacije o položaju glave korisnika. Ova funkcionalnost se ostvaruje korištenjem vanjskih knjižnica kako je navedeno u poglavlju 2.3.1.

Ovdje ćemo detaljno razmotriti što je potrebno napraviti s programske strane da se ostvari komunikacija s vanjskim knjižnicama i kako su informacije primijenjene u igri.

### 4.3.1. Izrada potrebnih razreda

U Unity projektu potrebno je izraditi dva parcijalna razreda:

- VisageTrackerNative
- VisageTrackerNative.Windows

Razred "VisageTrackerNative" ostavimo praznim, a u "VisageTrackerNative.Windows" iz knjižnice "VisageTrackerUnityPlugin" uzmimo sljedeće metode:

- _initTracker(string config)
- _trackFromCam()
- _stopTracker()
- _get3DData(out float tx, out float ty, out float tz,
  out float rx, out float ry, out float rz, out int status)

Sada kad imamo potrebne podatke napišimo skriptu "VisageTracker" koja će inicijalizirati i pokrenuti praćenje lica.

Na početku potrebno je učitati konfiguracijsku datoteku i pozvati metodu _initTracker(string config) s tom datotekom. Nakon što smo inicijalizirali "Tracker" možemo pokrenuti praćenje lica pomoću _trackFromCam(). Moguće je pratiti i iz drugih izvora, a kako to nama u ovom slučaju nije potrebno pratimo samo iz kamere.

Sada nam samo preostaje da svaki *frame* pozivamo metodu _get3DDate(...) i na temelju zadnjeg argumenta "status" odrediti da li moramo pokrenuti praćenje iz početka ili ne. Ako je status u redu tada informacije o položaju glave skripta zapisuje u razred "GlobalClass" da ta informacija bude vidljiva ostalim objektima u igri.

"Tracker" može biti u različitim stanjima što očitavamo iz varijable "status":

- $0 \rightarrow$ Off
- $1 \rightarrow$ Ok
- $2 \rightarrow$ Oporavljanje
- $3 \rightarrow$ Inicijalizacija

### 4.3.2. Primjena plugin-a u igri

Nakon što imamo "grube" koordinate glave potrebno ih je primijeniti u igri. Kako raspon koordinata koji nudi "Tracker" nije prilagođen potrebama igre potrebno je koordinate pomnožiti s određenim faktorima.

Za primjenu u igri koristi sljedeće transformacije:

- X * (-8) → X
- Y * 10 - 10 → Y
- Z * 5 → Z

Y koordinata se množi s duplo većim brojem od Z jer kamere snimaju u "landscape" načinu rade te je potrebno kompenzirati taj efekt. Time je omogućeno lakše gledanje gornje i donje strane ključa.

Sada kad imamo transformirane koordinate potrebno je obaviti još jedan korak koji će nam osigurati da naša kamera ne "pobjegne" iz scene. Ovaj korak je vrlo bitan jer kada "Tracker" izgubi položaj glave korisnika tada kada pozovemo metodu _get3DData() sigurno ćemo dobiti koordinatu koja se nalazi van cijele scene. Ovaj učinak se suzbija praćenjem da li se koordinate nalazi izvan zadanih ograničenja.

Logika za prethodni mehanizam je:

- Ako je koordinata izvan ograničenja
  → koordinata glave je jednaka prethodnoj koordinati
- Inače
  → prethodna koordinata je jednaka trenutnoj

Tek sada nakon svih ovih provjera možemo koordinatu modela "Player" postaviti na novi koordinatu. Sada kada se "Player" nalazi na novoj poziciji na ekranu se vidi model ključa iz druge perspektive i time je postignut efekt "gledanja kroz prozor".

## 4.4. Grafički modeli

Grafički modeli su izrađeni pomoću photoshop-a i blender-a. Photoshop je služio za izradi i doradu tekstura, a blender za izradu 3D modela.

### 4.4.1. Blender modeli

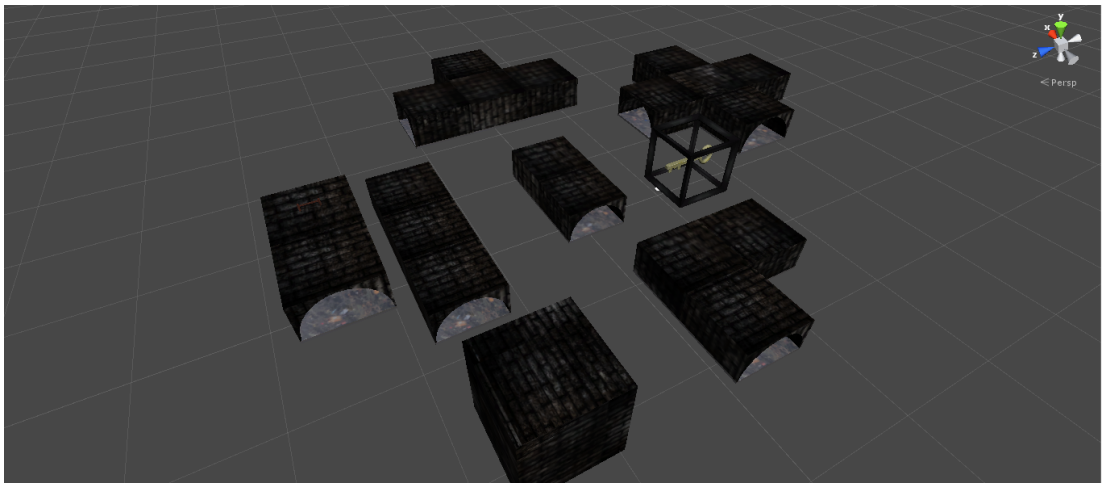
Pomoću programa blender izrađeni su modeli tunela, ključa i ljestva koje se nalazu na kraju labirinta.

Tuneli su izrađeni koristeći alat "Boolean". Pomoću njega moguće je od jednog objekta izrezati proporciju drugog objekta. Tako su modeli izrađeni od kocke od koje je odrezan valjak, kugla ili kombinacija oboje.

Ključ je izrađen kao kompozit dvaju objekta: okvira i samog ključa. Za okvir je korišten "WireFrame" modifikator koji je primijenjen na modelu kocke, a ključ je izrađen kombinacijom već gotovih objekata koje nudi "Blender".

Ljestve su izrađene koristeći alat "Array" pomoću kojeg se može jedan objekt duplicirati više puta.

Za postavljanje tekstura na objekte korišten je alat "UV wrapping". Nad objektom na kojem želimo postaviti teksturu odaberemo opciju "UV unwrap". Tada možemo detaljno odrediti koji dio slike se treba preslikati na koji dio objekta.

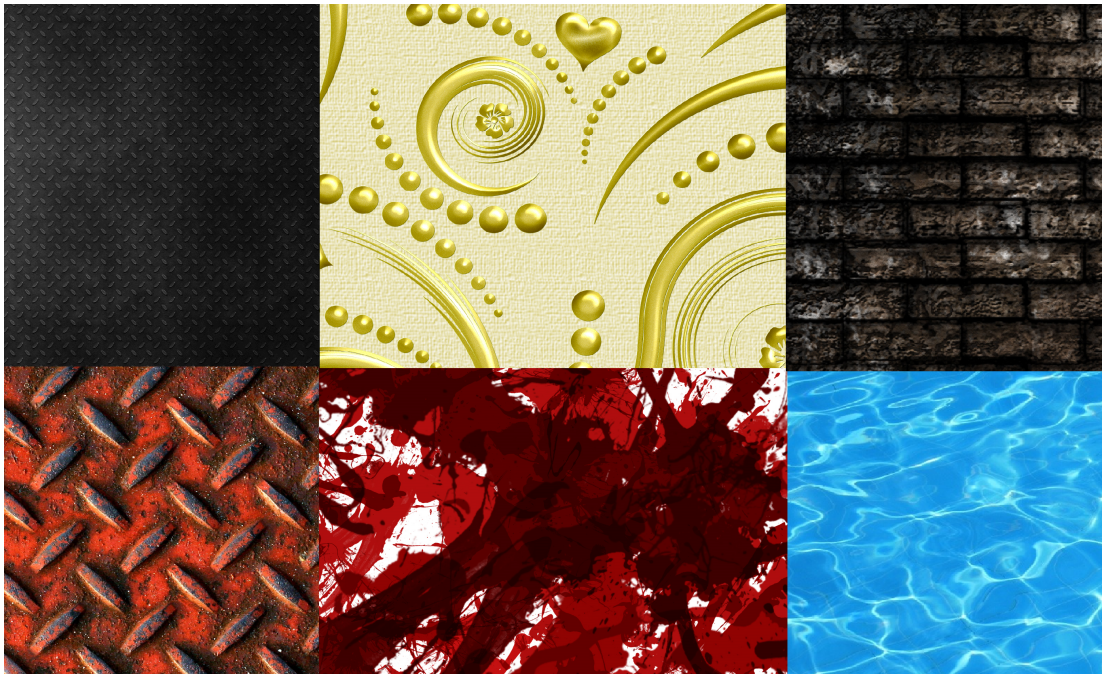



**Slika 4.8:** Modeli igre.

### 4.4.2. Teksture

Teksture koje su dorađene/izrađene u photoshop-u su: tekstura naslova igre i tekstura za "Finish". Obje teksture su izrađene istom metodom preslikavanja slike na tekst. Prvo se tekst napiše preko slike tada s bojom koja odskače od slike. Izbriše se cijeli ostatak. Nakon toga se izbriše tekst i preko "History Brush" se vrati pozadina iza teksta.

Teksture krvi, zlata (za ključ i "Finish"), zida i vode su skinute s interneta.



**Slika 4.9:** Teksture igre.

# 5. Resursi programa

## 5.1. Teksture

Tekstura zlata koja se koristi za natpis "Finish" i za teksturu ključa uzeta je sa stranice:
http://www.forwallpaper.com/wallpaper/turkuaz-gold-textures-1-516287.html

Tekstura koja se koristi za pozadinu scena i teksture zidova je skinuta sa stranice:
http://www.psdgraphics.com/backgrounds/grunge-brick-wall-background/

Tekstura vode čija je boja promijenjena u crvenu radi efekta krvi, skinuta je sa stranice:
http://cs.gamebanana.com/textures/3711

Tekstura korištena za naslov igre skinuta je sa stranice:
http://fc01.deviantart.net/fs71/f/2010/308/b/0/free_texture_by_smileys_4_eva-d3252et.jpg

Tekstura metala korištena za okvir ključa skinuta je sa stranice:
https://abstracthdwallpapers.wordpress.com/2012/09/12/metal-texture/

Tekstura zarđalog metala korištena za ljestve skinuta je sa stranice:
http://pixgood.com/rusty-metal-texture.html

## 5.2. Font

Korišteni font u igri je skinut sa stranice:
https://www.assetstore.unity3d.com/en/#!/content/4233

## 5.3. Zvukovi

Zvukovi za tipke su skinuti sa stranice:
https://www.assetstore.unity3d.com/en/#!/content/22259

# 6. Zaključak

Računalni programi a posebice računalne igre su kroz svoj razvoj težile ka većoj razini interakcije s korisnikom. Iz dosadašnjih trendova vidljivo je kako su tehnologije poput ekrana na dodir i "naočala" za virtualnu realnost, su potpuni uspjeh. Sasvim je jasno da je tehnologija praćenja položaja glave korak u pravom smjeru.

U izradi ovoga rada veoma dolazi do izražaja rad s vanjskim alatima i resursima. Visage razvojni alat u pozadini krije veoma kompleksnu funkcionalnost praćenja ljudskog lica i gesta. Korištenjem već gotovih razvojnih alata možemo postići složene funkcionalnosti u veoma kratkom roku.

U konačnici s jednostavnim preslikavanjem relativnih koordinata glave na relativne koordinate virtualne kamere koja se nalazi u programu postignut je efekt (osjećaj) "gledanja kroz prozor". Time je ujedno i postignut cilj ovog rada.

# LITERATURA


[1] Blender (software). 2015. URL http://en.wikipedia.org/wiki/Blender_%28software%299.

[2] Adobe photoshop. 2015. URL http://en.wikipedia.org/wiki/Adobe_Photoshop.

[3] Unity (game engine). 2015. URL http://en.wikipedia.org/wiki/Unity_%28game_engine%29.

[4] Microsoft visual studio. 2015. URL http://en.wikipedia.org/wiki/Microsoft_Visual_Studio9.

[5] Visage Technologies. *visage|SDK$^{TM}$ v7.2.1268*. Visage Technologies AB, 2002.

## Iscrtavanje 3D grafike prema položaju glave


### Sažetak

Cilj ovog rada je postići drugačiju sliku na ekranu ovisno o položaju glave, to jest kutu i udaljenosti iz koje se gleda na ekran.

Efekt je postignut koristeći Visage razvojni alat koji posjeduje knjižnice čije funkcije vračaju 3D koordinate i rotaciju glave. Ta funkcionalnost je implementirana u igri "Tunnel Terror" u kojoj igrač mora naći ključ na kojem se nalazi četveroznamenkasti kod da bih mogao prijeći razinu. Četveroznamenkasti kod se nalazi samo s jednoj od četiri mogućih strana ključa i igrač mora pomicati vlastitu glavu da pogleda ključ s druge perspektive. Svaka razina igre se u potpunosti nasumično generira.

Igra je napravljena pomoću programa: Unity, Blender, Photoshop i Visual Studio 2013.



**Ključne riječi:** Visage, Tunnel Terror, igra, pračenje, iscrtavanje, Unity, Blender, Photoshop, Visual Studio


## View dependent rendering


### Abstract

The goal of this project to achieve different picture on the screen depending of head position i.e. angle and distance from which screen is looked at.

Effect is achieved using Visage SDK which has libraries whose functions return 3D position and rotation of the head. That functionality is implemented in game the "Tunnel Terror" where player has to find a key on which is four digit code located in order to pass a level. Four digit code is located on only one out of four possible sides of the key so player must move his/her head to look at the key from different perspective. Every level is completely randomly generated.

Game is build with following software: Unity, Blender, Photoshop and Visual Studio 2013.



**Keywords:** Visage, Tunnel Terror, game, tracking, rendering, Unity, Blender, Photoshop, Visual Studio