SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3192

Animirani objekti u okruženju Unity

Nikolina Očić

Zagreb, lipanj 2013.

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

SADRŽAJ

1. Uvod

U doba u kojem vlada tehnologija, s animacijom se susrećemo na dnevnoj bazi. Prisutna je u filmovima, televizijskim programima, računalnim igrama, internetskim stranicama, čak i kućanskim aparatima. Iako tako ne izgleda, koncept animacije datira sve do kamenog doba. Čovjek je uvijek tražio način kako animirati crteže i skice, no pravi procvat animacije krenuo je tek s razvojem računalne znanosti. Usko povezana s animacijom, računalna znanost uvodi nove načine i tehnike animacije. Razvitkom znanosti, sve se više razvija i sama animacija koja postaje toliko realna da više ni sami ne znamo što je stvarno snimljeno a što animirano.

Iako je animacija sveprisutna, osim osnovnih stavki, o njoj malo znamo. Ovaj rad dat će kratak uvid u pojam animacije te prikazati primjere u pogonu Unity.

U drugom poglavlju upoznat ćemo se s počecima animacije te vidjeti kako se stvarala kroz povijest. Također, bit će prikazano kako se animacija danas stvara te koje se tehnike pritom koriste.

Treće poglavlje donosi kratak uvid u povijest razvoja grafičkog pogona Unity te kratke upute o korištenju.

Četvrto poglavlje bavit će se osnovnim geometrijskim transformacijama potrebnim za razvoj animacija.

Zadnje poglavlje, ujedno i najvažnije, govorit će o animacijama u pogonu Unity. Biti će objašnjena uporaba skripti (eng. *scripts*) za kreiranje animacija te prikazani osnovni primjeri istih. U ovome poglavlju bit će prikazan programski zadatak na kojem se temelji ovaj rad.

2. Animacija

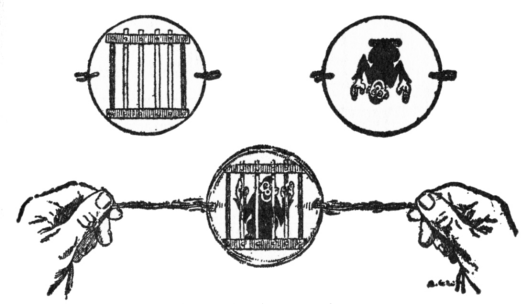
2.1. Povijest animacije

Riječ **animacija** dolazi od latinske riječi *animatio* koja u prijevodu označava čin oživljavanja. Možemo reći da je animacija oživljavanje nepokretnih objekata. U svakodnevici, pod pojmom animacije podrazumijevamo brzi prikaz niza slika kako bi se dobio dojam kretanja, dok u računalnom svijetu, pojam animacije označava proces generiranja animiranih slika uz pomoć računalne grafike.

Prvi pokušaji animacije sežu sve do doba paleolitika, gdje su životinje prikazivane s više nogu kako bi se dobio dojam kretanja. Pravi zamah, animacija je dobila u 19. stoljeću, kada se pojavljuju brojne jednostavne naprave kojima su se uspješno dobivale animirane slike. Ovakve naprave služile su za razonodu i zabavu. Kako nisu projicirale slike i u nekom trenutku mogle su biti korištene od strane samo jedne osobe, smatrane su igračkama. Mnoge se i danas koriste pri učenju temeljnih principa animacije.

Thaumatrop (1824)

Riječ je o kartonskom disku ili ploči koja je sa svake strane imala naslikane različite slike. Ploča je na dva nasuprotna dijela bila pričvršćena na vrpcu koja se vrtjela među prstima. Prilikom vrtnje, dolazilo je do brze izmjene slika, te zahvaljujući tromosti oka dobio se dojam da se slike stapaju u jednu.



Slika 2.1.1: Thaumatrop

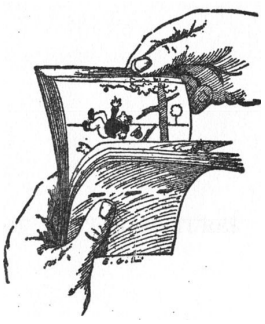
Fenakistoskop (1831)

Sastojao se od diska okomito pričvršćenog na ručku. Oko centra diska, bio je radijalno raspoređen niz slika koje su prikazivale faze u animaciji. Kada bi se disk zavrtio, slike bi stvorile dojam pokreta.

Slika 2.1.2: Fenakistoskop

Slikovne knjižice (1868)

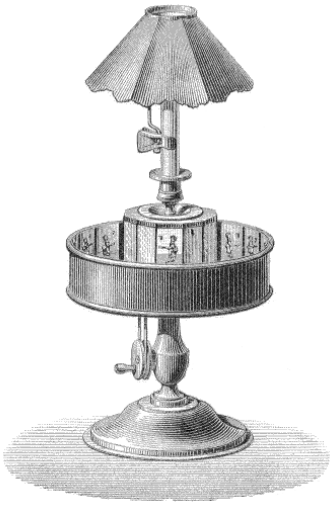
Kao što i naziv kaže, slikovne knjige su bile knjige sa nizom sličica koje su tvorile neku animaciju, a bile su raspoređe postepeno po stranicama. Korisnik bi, brzo listajući stranice, dobio dojam kretanja animacije.



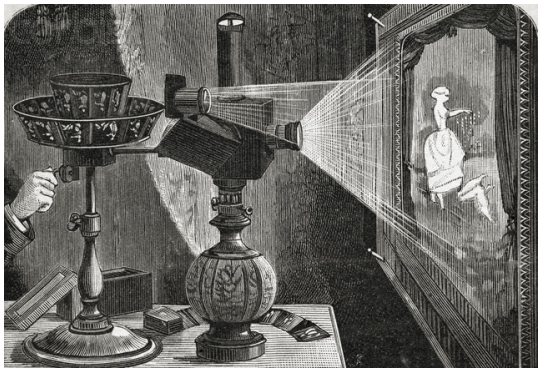
Slika 2.1.3: Slikovna knjižica

Praksinoskop (1877)

Uređaj se sastojao od rotirajućeg cilindra kojemu se sa unutrašnje strane nalazila vrpca sa slikama u nizu. Također, u centru cilindra nalazio se statični krug ogledala u kojemu se gledala animacija kada bi se cilindar zavrtio. Veća verzija praksinoskopa, *Theatre Optique*, imala je mogućnost projiciranja slika na platno.

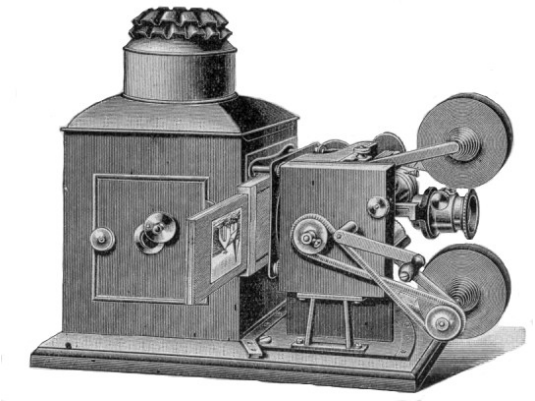


Slika 2.1.4: Praksinoskop



Slika 2.1.5: Theatre Optique

Značajan napredak animacije dogodio se tek pojavom kinematografije. **Kinematograf** je uređaj koji su kreirali ***Auguste i Louise Lumiere*** a sadržavao je projektor, printer i kameru. Omogućavao je prikazivanje pokretnih slika na platnu.



Slika 2.1.6: Kinematograf

2.2. Tehnike animacije

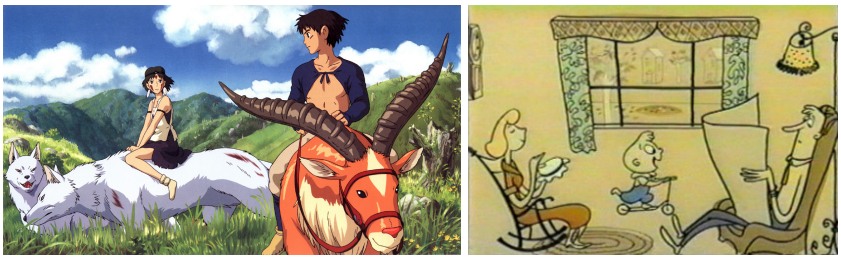
2.2.1. Tradicionalna animacija

Tradicionalna animacija je najčešće korišteni postupak animacije 20. stoljeća. Pojedini kadrovi tradicionalno animiranih filmova su zapravo ručno crtane slike, pri čemu se svaka slika neznatno razlikuje od prethodne. Njihovim brzim izmjenama i prikazivanjem dobiva se iluzija pokreta. Same slike nastaju kao skice koje se prenose na tzv. *celuloidne trake* i snimaju specijalnim kamerama.
Proces tradicionalne animacije je do početka 21. stoljeća zastario. Danas se skice skeniraju ili najčešće crtaju izravno na računalnim sustavima te se animiraju uz pomoć raznih programskih paketa.

Potpuna animacija (eng. *full animation*) se odnosi na proces produkcije visokokvalitetnih tradicionalno animiranih filmova koji koriste detaljne crteže i pokrete. Primjeri tradicionalne animacije uključuju filmove: *Aladdin* (SAD, 1992.), *Kralj lavova* (SAD, 1994.) te *Željezni div* (SAD, 1999.).

Slika 2.2.7: Potpuna animacija: *Aladdin*, *Kralj lavova* i *Željezni div*

Djelomična animacija (eng. *limited animation*) uključuje korištenje manje detaljnih a više stiliziranih crteža i metoda pokreta. Djelomična animacija je često korištena kao metoda stiliziranog umjetničkog izričaja, pa tako velik broj *anime* filmova japanske produkcije pripada ovom stilu. Kao primjer djelomične animacije možemo također navesti film *Gerald McBoing-Boing* (SAD, 1951.).



Slika 2.2.8: Djelomična animacija: *Princeza Monoonoke* i *Gerald McBoing-Boing*

Rotoskopija (eng. *rotoscoping*) je tehnika u kojoj animatori prate prethodno snimljene pokrete. Izvorni film može biti izravno kopiran kao animirani crtež iz silueta glumaca ili može biti stiliziran. Neki primjeri rotoskopije su *Gospodar prstenova* (SAD, 1978.) i *Vatra i led* (SAD, 1983.).

Slika 2.2.9: Rotoskopija: *Gospodar prstenova* i *Vatra i led*

Igrana animacija (eng. *live-action/animation*) je tehnika kod koje se crteži crtaju preko igranih scena. Kao primjer igrane animacije možemo navesti filmove *Tko je smjestio zeki Rogeru?* (SAD, 1988.) i *Space Jam* (SAD, 1996.).

Slika 2.2.10: Igrana animacija: *Tko je smjestio zeki Rogeru?* i *Space Jam*

2.2.2. Stop-motion animacija

Stop-motion animacija je tehnika kojom se iluzija kretanja dobiva manipuliranjem stvarnih objekata te njihovim fotografiranjem kadar po kadar filma. Razlikujemo nekoliko različitih tipova stop-motion animacije, obično nazvanih prema materijalu koji se koristi za izradu.

Lutkarska animacija (eng. *puppet animation*) koristi lutke koje se stavljaju u interakciju s konstruiranom okolinom. Lutke u pravilu imaju žičani kostur koji služi lakšem manipuliranju lutkama i njihovim pozama. Primjeri lutkarske animacije su *Predbožićna noćna mora* (SAD, 1993.) i *Mrtva nevjesta* (SAD, 2005.).

Slika 2.2.11: Lutkarska animacija: *Predbožićna noćna mora* i *Mrtva nevjesta*

Glinena animacija (eng. *clay animation*) koristi glinene lutke koje mogu biti u potpunosti napravljene od gline ili mogu sadržavati žičani kostur. Primjer glinene animacije je film *Wallace i Gromit* (UK, 1989.).

Slika 2.2.12: Glinena animacija: *Wallace i Gromit*

Animacija izrezivanjem (eng. *cutout animation*) je tehnika koja koristi dvodimenzionalni objekt napravljen od papira ili tkanine. Varijanta animacije izrezivanjem je **siluetalna animacija** (eng. *silhouette animation*) kod koje se izrezani likovi osvjetljuju i vidljivi su samo kao siluete. Primjer animacije izrezivanjem je kratak film *Čudo leta* (UK, 1974.) dok kao primjer siluetalne animacije možemo navesti *Avanture princa Achmeda* (NJEM, 1926.).

Slika 2.2.13: Animacija izrezivanjem: *Čudo leta*

Slika 2.2.14: Siluetalna animacija: *Avanture princa Achmeda*

Animacija modela (eng. *model animation*) je tehnika animacije kod koje se stop-motion animacija ugrađuje u igrani svijet, stapajući se sa glumcima i igranom okolinom. Kao primjere animacije modela možemo navesti filmove *Jason i Argonauti* (SAD, 1963.) te *King Kong* (SAD, 1933.).

Slika 2.2.15: Animacija modela: *Jason i Argonauti* i *King Kong*

Animacija objekata (eng. *object animation*) koristi svakodnevne nežive objekte nasuprot specijalno stvorenih objekata koji se koriste u prethodnim tehnikama. Podvrste animacije objekata su ***grafička animacija*** (eng. *graphic animation*) koja koristi nenacrtane ravne vizualne objekte (fotografije, novine, časopise) za stvaranje animacije te ***brickfilm animacija*** (eng. *brickfilm*) koja koristi Lego figure ili slične igračke za izradu animacija.

Slika 2.2.16: Brickfilm animacija

2.2.3. Računalna animacija

Računalna animacija je u osnovi digitalni nasljednik stop-motion animacije, koja može koristiti 3D modele ili digitalne 2D ilustracije. Obuhvaća brojne metode izrade animacija na računalnim sustavima. Glavna prednost računalne animacije je brža produkcija i izrada animacija. Tehnike 2D animacije fokusiraju se na manipulaciju slika dok se 3D animacija fokusira na izradu virtualnih svijetova u kojima se likovi i objekti kreću i međusobno su povezani.

2D računalna animacija

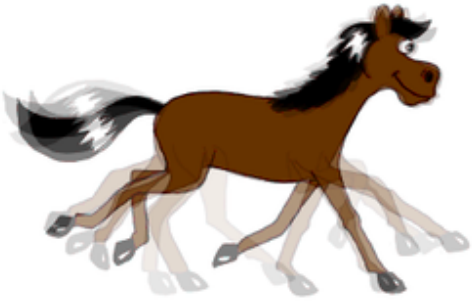
Slikovni objekti 2D animacije nastaju na računalima uz pomoć 2D rasterske grafike[1] ili 2D vektorske grafike[2]. Najčešće tehnike 2D računalne grafike uključuju automatizirane računalne postupke tradicionalne animacije : ***morfizam***, ***interpolacijska rotoskopija*** i ***onion skinning***.

Morfizam (eng. *morphing*) je postupak korišten za izradu specijanih efekata kojim se jedna slika "pretvara" u drugu korištenjem niza diskretnih prijelaza i izobličenja slika.

Slika 2.2.17: Morfizam

Interpolacijska rotoskopija (eng. *interpolated rotoscoping*) je tehnika slična postupku rotoskopije u tradicionalnoj animaciji kod koje se animacija crta preko prije snimljenog igranog filma.

Onion skinning je postupak izrade animiranih flmova kod kojeg se više bliskih slika stapaju u jednu.



Slika 2.2.18: Onion skinning

[1] *Rasterska grafika* ili *bitmap* predstavlja mrežu piksela ili obojenih točaka na nekom izlaznom uređaju kao što je monitor.[2]

[2] *Vektorska grafika* ili *geometrijsko oblikovanje* je način prikazivanja slike pomoću geometrijskih oblika koji su temeljeni na matematičkim jednadžbama.[2]

3D računalna animacija

3D animacija je novije područje računalne grafike. Tehnike 3D animacije zahtijevaju dobro poznavanje matematičkih osnova u grafici, modeliranje objekata te poznavanje fizikalnih svojstava i struktura objekta koji se modelira. 3D animacija se modelira i obrađuje uz pomoć posebnih programskih paketa. Početni korak je kreiranje 3D objekta kojim se manipulira na željeni način. Prednost 3D animacije je njena realističnost zbog koje se može dogoditi da je teško razlikovati stvorenu animaciju od stvarno snimljenih pokreta što ovaj oblik animacije čini vrlo pogodnim za izradu specijalnih efekata u filmovima.

Fotorealistična animacija (eng. *photo-realistic animation*) se koristi prvenstveno za izradu animacija kojima se želi postići što veći prikaz stvarnosti. Ona koristi složene algoritme prikazivanja[3] kojima se dobivaju objekti s velikim brojem detalja kako bi što bolje oponašali objekte iz stvarnosti. Primjeri korištenja fotorealistične animacije je serijal igara *Final Fantasy* (JAPAN).

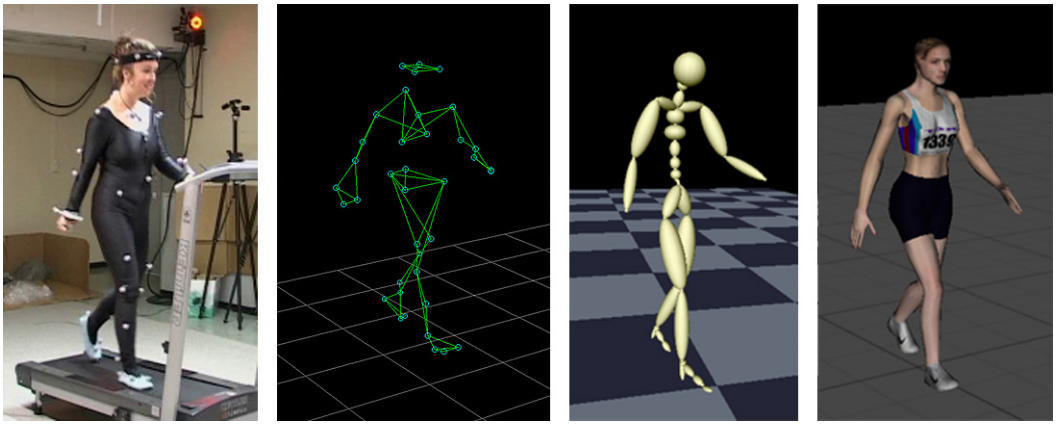
Slika 2.2.19: Fotorealistična animacija: *Final Fantasy*

Cel-sjenčanje (eng. *cel-shaded animation*) se koristi za oponašanje tradicionalne animacije uporabom grafičkih programa kako bi animacija izgledala kao da je crtana rukom. Ova tehnika se najčešće koristi kada se želi oponašati stil stripova. Cel-sjenčanje se odvija u 3 koraka : odabranom 3D modelu se prvo naglašavaju obrisi i konture, zatim se model boja osnovnom teksturom te se na kraju sjenča. Primjer cel-sjenčanja je igra *The Legend of Zelda: The Wind Maker* (JAPAN, 2002.)

[3] *Render* ili *renderiranje* je proces stvaranja slike od nekog modela uz pomoć posebnog programa.[2]

Slika 2.2.20: Cel-sjenčanje

Hvatanje pokreta (eng. *motion capture*) je proces snimanja kretanja objekta ili ljudi kako bi se animirali digitalni likovi. Kretanje se snima i prevodi u digitalni oblik uz pomoć posebne računalne opreme i kamera. Na objekt čiji se pokreti snimaju pozicioniraju se tzv.*markeri* na specifične točke tijela. Markeri se snimaju preciznim kamerama te se dobiveni podaci računalno obrađuju kako bi se dobili podaci o pozicijama, brzini i akceleraciji markera čime se dobiva digitalna reprezentacija pokreta. Ovakav postupak rezultira vrlo preciznom i realističnom digitalnom reprezentacijom pokreta ali zahtjeva skupu računalnu opremu, velik prostor i veći broj stručnjaka. Hvatanje pokreta korišteno je u filmovima *Avanture Tintina* (SAD, 2011.) te *Božićna priča* (SAD, 2009.)



Slika 2.2.21: Hvatanje pokreta

Skeletalna animacija (eng. *skeletal animation*) je najčešće korišten postupak računalne animacije. Model je predstavljen u dva dijela : površinom koja ocrtava lik (eng. *skin* ili *mesh*) i skupom povezanih kostiju (eng. *skeleton* ili *rig*) koje služe za animiranje objekta. Najčešće se koristi za animaciju ljudi ili životinja no može se koristiti za animaciju bilo kojeg objekta.

3. Grafički pogon Unity

3.1. Uvod u grafičke programe

Pojavom računalne grafike i animacije pojavila se i potreba za određenim specijaliziranim grafičkim programima. *Grafička programska oprema* je računalni program ili skup programa koji omogućuju pregledavanje i manipulaciju grafičkim podatcima.
Razvojem računalne animacije došlo je do velikog razvoja programa i platformi za izradu animacija, kako profesionalnih tako i jednostavnijih, za upotrebu "običnog" korisnika. Kako je tema ovog rada animacija u pogonu Unity, detaljnije o samom pogonu bit će u daljnjim poglavljima.

3.2. Povijesni razvoj pogona Unity

Unity (poznat još i kao **Unity3D**) je više-platformski[1] grafički pogon razvijen za izradu 2D i 3D video igara. Prvotno je razvijen za izradu mobilnih i web igara te upotrebu na sustavu *OS X*[2] no s vremenom se proširio na ostale sustave i dobio mogućnost izrade igara za igrače konzole i osobna računala.
Sam početak Unity-a usko je vezan uz tvrtku **Unity Technologies** koja je osnovana zbog razvitka samog pogona. Tvrtku su 2004. godine osnovali David Helgason, Nicholas Francis i Joachim Ante s ciljem razvitka pogona prihvatljive cijene za uporabu šire javnosti. Fokus tvrtke je "*demokratizirati razvitak igara*" te ga učiniti pristupačnim ljudima diljem svijeta. 2009. godine Unity Technologies lansirao je besplatnu verziju pogona nedugo nakon čega je zabilježen znatan porast registriranih korisnika. Danas ta brojka prelazi milijun registriranih korisnika od čega je tristo tisuća redovitih mjesečnih korisnika.

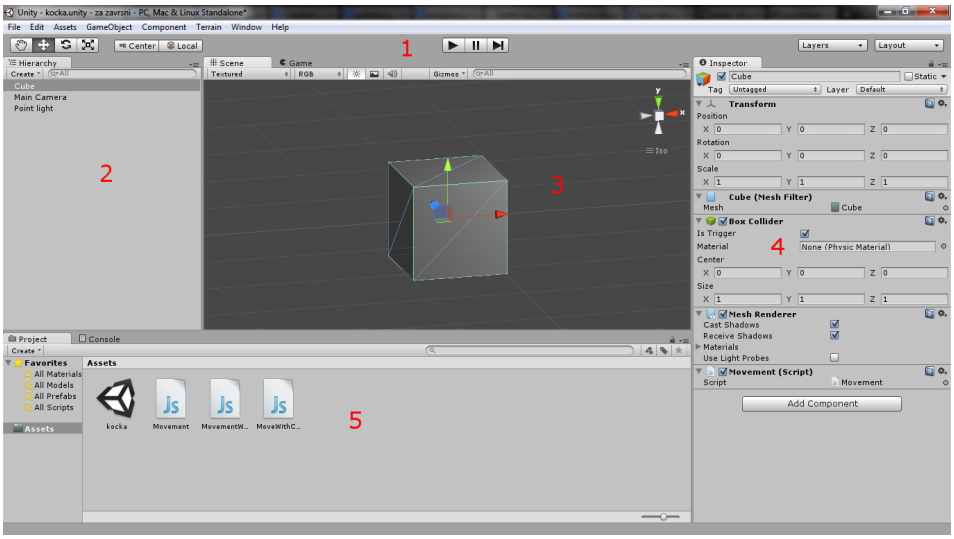
[1] eng. *cross-platform* - mogućnost korištenja programske opreme na različitim računalnim platformama.[1]

[2] *OS X* je operacijski sustav razvijen od strane *Apple Inc.* a korišten samo na *Mac* računalima.

3.3. Uporaba pogona Unity

Unity, pristupačan "običnom" korisniku, nudi grafičko korisničko sučelje intuitivno za korištenje. Sam pogon sadrži brojne mogućnosti i dodatke za lakšu izradu korisničkog proizvoda.

3.3.1. Korisničko sučelje



Slika 3.3.1: Korisničko sučelje

Korisničko sučelje, sastoji se od alatne trake (**1**) (eng. *Toolbar*), hijerarhije projekta (**1**) (eng. *Hierarchy*), prikaza scene (**3**) (eng. *Scene View*), inspektora (**4**) (eng. *Inspector*) koji služi za pregled parametara aktivnog objekta, te preglednik projekta (**5**) (eng. *Project Browser*) koji prikazuje sve objekte i uključene resurse (eng. *assets*) pojedinog projekta.

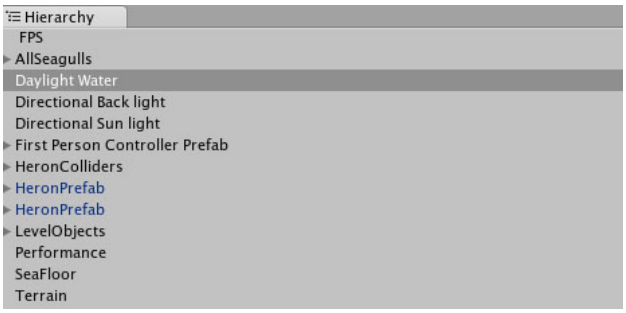
Alatna traka (eng. *Toolbar*)

Alatna traka sastoji se od pet osnovnih kontrola, svaka povezana s drugim dijelom uređivača (eng. *Editor*). Alati za transformaciju (**1** i **2**) služe za manipuliranje prikazom scene, *Play*,*Pause* i *Step* tipke (**3**) služe za prikaz igre, tj. animirane scene, padajući izbornik *Layers* (**4**) služi za odabir objekta koji se prikazuje u prikazu scene dok padajući izbornik *Layouts* (**5**) služi za kontrolu svih prikaza.

Slika 3.3.2: Alatna traka

Hijerarhija projekta (eng. *Hierarchy*)

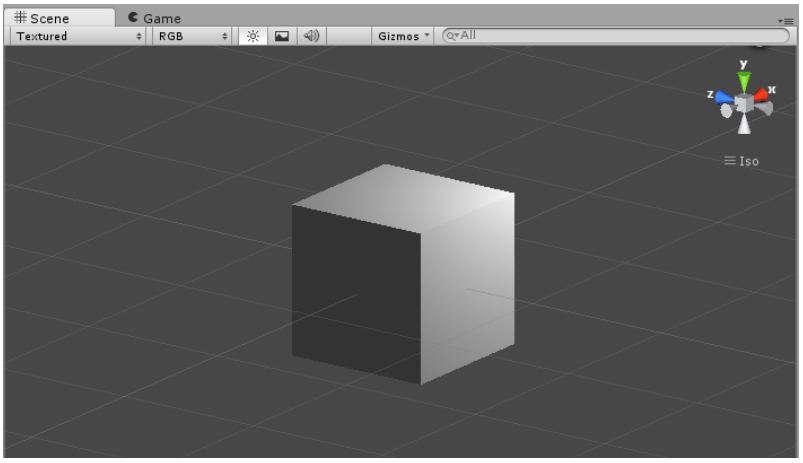
Hijerarhija sadrži sve objekte projekta u trenutnoj sceni. Unity koristi koncept nazvan **roditeljstvo** (eng. *Parenting*) koji koristimo kada želimo da neki objekt naslijedi svojstva drugog objekta. Tada objekt koji nasljeđuje nazivamo **dijete** (eng. *child*) dok je početni objekt **roditelj** (eng. *parent*). Objekt-roditelj može imati više objekata-djece, dok objekt-dijete može imati i svoje objekte-djecu.



Slika 3.3.3: Hijerarhija

Prikaz scene (eng. *Scene View*)

Prikaz scene je interaktivni "pješčanik" (eng. *sandbox*[3]) projekta. Prikaz scene se koristi za pozicioniranje okoline, igrača, kamere te svih ostalih objekata igre, odnosno projekta. Manevriranje i manipulacija objektima unutar prikaza scene ključne su funkcije u Unity-u. Vrlo je bitno obavljati ih čim brže te je u svrhu toga unutar pogona dostupan niz kratica na tipkovnici (eng. *HotKeys*) (v. Dodatak A).

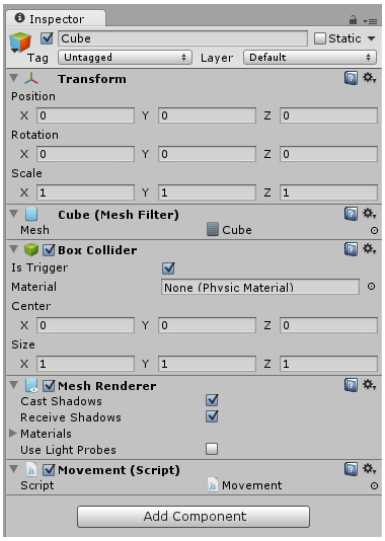


Slika 3.3.4: Prikaz scene projekta

[3] *Sandbox* je naziv za okruženje u kojem su točno određena ograničenja na to kojim sistemskim resursima aplet može zahtijevati pristup.[1]

Inspektor (eng. *Inspector*)

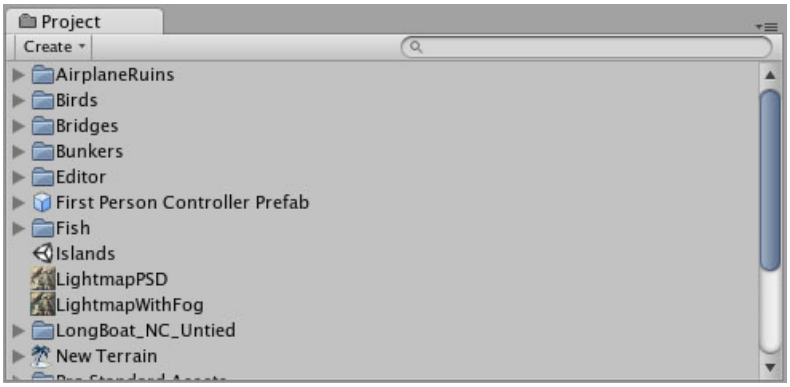
Inspektor prikazuje detaljne informacije o trenutno odabranom objektu. Kako uz objekte mogu biti vezane mreže točaka (eng. *mesh*)[4], skripte (eng. *scripts*)[5] [6], zvukovi te ostali grafički elementi, inspektor služi za modifikaciju istih. Svako svojstvo prikazano u inspektoru se može direktno mijenjati.



Slika 3.3.5: Inspektor

Preglednik projekta (eng. *Project Browser*)

U pregledniku korisnik može pristupiti svim objektima i ostalim datotekama koje su sadržane u projektu.



Slika 3.3.6: Preglednik projekta

[4]U računalnoj grafici *mesh* je skup vrhova, bridova i poligona koji određuje 3D objekt.
[5]Skripta je lista naredbi koja se može izvršiti bez uključivanja korisnika.[1]
[6]Skriptni jezici koje koristi Unity su *JavaScript*, *C#* i *Boo*[6].

4. Geometrijske transformacije

Osnova pri manipuliranju 3D objekata te osnova pri izradi animacija su geometrijske transformacije. Svaka animacija je u suštini niz transformacija, koje svojom kombinacijom pružaju željeni efekt. U nastavku će biti spomenute važnije transformacije te njihova upotreba u pogonu Unity.

Unutar Unity-a, svaki objekt u sceni (eng. *Scene*) sadrži komponentu **Transform**. Ona se koristi za manipulaciju položaja tj. translacije, rotacije i skaliranja objekta. Svaka komponenta **Transform** može imati roditelja (eng. *parent*) što omogućuje da se translacija, rotacija ili skaliranje primijenjuju na pojedinoj razini hijerarhije. Također, moguća je manipulacija i komponente **Transform** djeteta. Ilustrativni primjer izvornog koda dan je u nastavku[1]:

```
for (var child : Transform in transform) {
    child.position += Vector3.up * 10.0;
}
```

Gornji kod pomiče svu transformiranu djecu objekta za 10 jedinica pomaka prema gore.

Unutar komponente **Transform** nalazimo i dvije najvažnije transformacije osnovne za bilo kakve animacije : **translacija** i **rotacija** .

[1] Preuzeto sa [11].

4.1. Translacija

Translacija je transformacija koja svaki objekt u radnom prostoru pomiče za zadani pomak. Unutar komponente **Transform** postoje dvije ugrađene funckije translacije :

```
function Translate (translation : Vector3, relativeTo : Space =
    Space.Self) : void
```

Funckija pomiče objekt u smjeru i za vrijednost vektora **translation**. Varijabla **relativeTo** određuje u odnosu na što se objekt pomiče. Ako je zadano **Space.Self** kretanje se odvija u odnosu na koordinatni sustav objekta, tj. lokalni koordinatni sustav a ako je zadano **Space.World** kretanje se odvija u odnosu na globalni koordinatni sustav.

```
function Translate (x : float, y : float, z : float, relativeTo :
    Space = Space.Self) : void
```

Funkcija pomiče objekt za vrijednost **x** po X osi, za vrijednost **y** po Y osi i za vrijednost **z** po Z osi. Varijabla **relativeTo** je ista kao i u gornjem primjeru.

U oba slučaja varijabla **relativeTo** može biti i tipa **Transform** (*relativeTo : Transform*). Tada se u tom slučaju kretanje odvija u odnosu na koordinatni sustav objekta , tj. lokalni koordinatni sustav. Ako je **relativeTo** jednaka *null*, kretanje se odvija u odnosu na globalni koordinatni sustav.

4.2. Rotacija

Rotacija je transformacija koja svaki objekt u radnom prostoru rotira za zadani kut rotacije. Objekt se može rotirati oko osi globalnog koordinatnog sustava ili ako je objekt ovisan o drugom objektu tada se on rotira oko osi koordinatnog sustava objekta o kojem ovisi.

Unity za pretpostavljeni tip varijable rotacije koristi tip **quaternion** koji sadrži komponente x, y, z i w. Rotacija se također može zadati i kao tip **eulerAngle** koji sadrži komponente x, y i z.

Unutar komponente **Transform** postoje tri ugrađene funkcije rotacije:

```
function Rotate (eulerAngles : Vector3, relativeTo : Space = Space.
    Self) : void
```

Funkcija rotira tijelo za **eulerAngles.z** oko Z osi, **eulerAngles.x** oko X osi i **eulerAngles.y** oko Y osi. Varijabla **relativeTo** određuje u odnosu na koji koordinatni sustav se objekt rotira. Ako je **relativeTo** izostavljeno ili je zadano **Space.Self** objekt se rotira oko osi koordinatnog sustava objekta, tj. oko osi lokalnog koordinatnog sustava a ako je zadano **Space.World** tada se objekt rotira oko osi globalnog koordinatnog sustava.

```
function Rotate (xAngle : float, yAngle : float, zAngle : float,
    relativeTo : Space = Space.Self) : void
```

Funkcija rotira tijelo za iznos kuta **zAngles** oko Z osi, **xAngles** oko X osi i **yAngles** oko Y osi. Varijabla **relativeTo** je kao u gornjem primjeru.

```
function Rotate (axis : Vector3, angle : float, relativeTo : Space =
     Space.Self) : void
```

Funkcija rotira objekt oko proizvoljno zadane osi **axis** za iznos kuta **angle**. Varijabla **relativeTo** je kao u gornjim primjerima.

5. Animacije u okruženju Unity

Izrada animacija u pogonu Unity moguća je na više načina : uvozom animacija napravljenih u drugim grafičkim pogonima, stvaranje izravno unutar pogona uporabom pogleda animacije (eng. *Animation View*) te stvaranje pomoću tehnike *Mecanim*[1]. Svaka od ovih metoda koristi upravljanje animacijama putem skripti na čemu se ovaj rad i temelji.

5.1. Skripte (eng. *Scripts*)

Za razliku od drugih resursa vezanih uz objekte, kao što su teksture ili vezane mreže točaka (eng. *mesh*), skripte mogu biti kreirane unutar samog pogona Unity. Korištenje skripti sastoji se od vezanja skriptnih objekata nazvanih **ponašanja** (eng. *behaviours*) na objekte igre, odnosno objekte animacije. Određeni događaji pozivaju različite funkcije unutar skripte. Najčešće korištene funckije su : ***Update*** unutar koje se nalazi većina izvornog koda vezanog za ponašanje u animaciji (izuzev fizikalnog izvornog koda (eng. *physics code*[2])) a poziva se prije prikazivanja pojedinog kadra, ***FixedUpdate*** unutar koje se izvode fizikalna ponašanja u animaciji a poziva se na svaki fizikalni vremenski korak (eng. *time step*) te **izvorni kod izvan neke funckije** koji se izvodi kada se objekt učitava a može se koristiti za inicijalizaciju skripte.
U nastavku ovog rada bit će prikazani primjeri animacija u pogonu Unity, počevši od najjednostavnijeg kao što je rotacija kocke. Skripte su pisane jezikom *JavaScript*.

[1] *Mecanim* je nova, napredna tehnologija animacije, uvedena u Unity verziji 4, koja omogućuje prirodnije animacije živih i neživih objekata.

[2] Fizikalni izvorni kod je kod koji opisuje fizikalne zakone i procese.

5.2. Jednostavne animacija

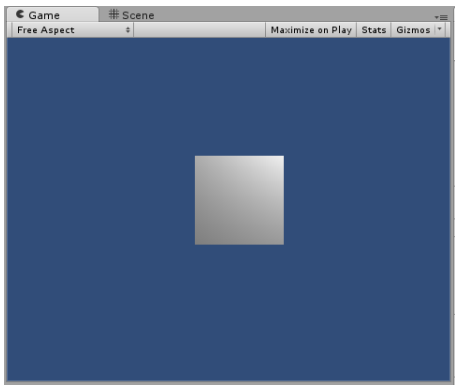
5.2.1. Rotacija kocke

Rotacija kocke je jedna od osnovnih animacija. Korišteni model kocke dostupan je kao jedan od osnovnih modela unutar samog programa Unity. Skripta za ovu animaciju je vrlo jednostavna. Izvorni kod priložen je u nastavku.

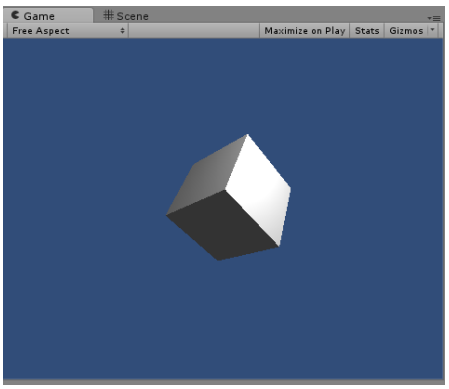
```
var brzina_x = 10.0;
var brzina_y = 10.0;
var brzina_z = 0.0;

function Update () {
    transform.Rotate(brzina_x*Time.deltaTime, brzina_y*Time.
        deltaTime, brzina_z*Time.deltaTime);
}
```

Funkcija ***transform.Rotate()*** je funkcija koja se nalazi u komponenti *Transform* objekta (kocke) a služi za rotiranje odabranog objekta. Kao argumenti funkcije zadaju se iznosi kuteva za koje želimo rotirati oko pojedine osi u 3D prostoru (X, Y i Z) . ***Time.deltaTime*** je član klase *Time* koja usklađuje pokrete preko jedne sekunde tako da se kocka rotira jednakom brzinom bez obzira koliko kadrova u sekundi računalo obrađuje. Korištene su i pomoćne varijable brzina_x, brzina_y i brzina_z pomoću kojih možemo izravno u *Inspektoru* definirati iznose kuteva za koje želimo rotirati kocku po pojedinim osima.



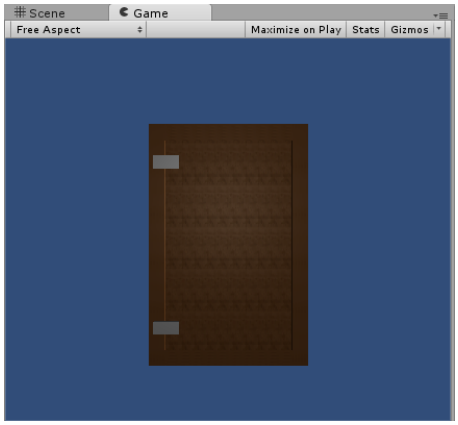
Slika 5.2.1: Kocka u početnom položaju



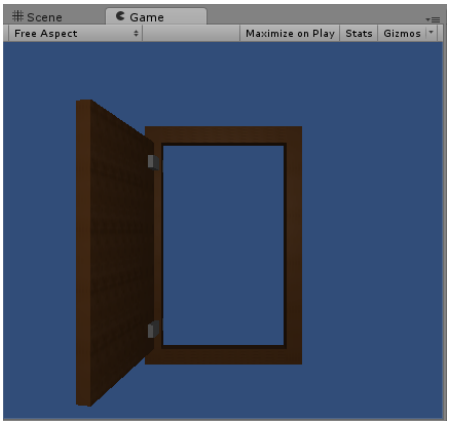
Slika 5.2.2: Zarotirana kocka

5.2.2. Otvaranje vrata

Otvaranje vrata možemo smatrati malo složenijom animacijom i to prvenstveno zbog posla oko pripreme modela. Sama skripta koja se koristi vrlo je jednostavna i sadržajem je slična skripti za rotaciju kocke. Model korišten u ovom primjeru izrađen je u samom pogonu Unity korištenjem jednog od osnovnih modela - kocke. Manipulacijom i modifikacijom više modela kocaka napravljen je okvir vrata, šarke vrata te sama vrata. Teksture i materijali preuzeti su kao besplatni dodaci preko *Asset Store*-a[3].



Slika 5.2.3: Zatvorena vrata



Slika 5.2.4: Otvorena vrata

Nakon stvaranja modela, kreirana je hijerarhija dijelova modela. Stvoreno je više cjelina koje tvore gotovi model. Korijenska cjelina je **DoorFrame** koja sadrži sve dijelove vrata - **Top**, **Bottom**, **Left** i **Right**. Cjelina **Left** sadrži cjelinu **Hinges** koja sadrži dijelove **Hinge_bottom**, **Hinge_top** i sama vrata **Door**. Skripta animacije pridružena je cjelini **Hinges**.

```
var hinge_y : float = 10.0;

function Update () {
        transform.Rotate(0, hinge_y*Time.deltaTime, 0);
}
```

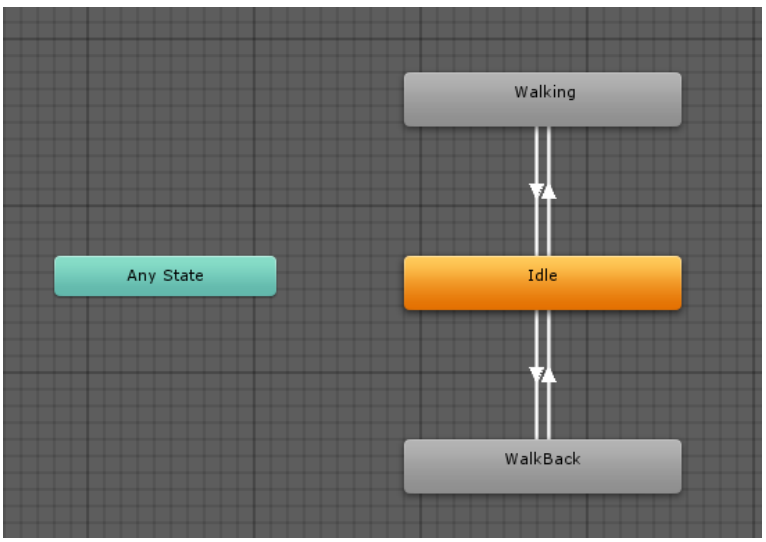
[3] *Asset store* je Unity-eva kolekcija raznih dodataka (modela, skripti i slično) stvorenih od strane profesionalaca i korisnika Unity-a dostupnih za besplatno ili plaćeno skidanje.

Slika 5.2.5: Prikaz hijerarhije dijelova modela vrata

5.3. Animiranje čovjekolikog modela

Animiranje čovjekolikog modela zahtjeva mnogo posla. Najjednostavniji način animacije uključuje korištenje tehnologije *Mecanim* zajedno sa uporabom skripti za upravljanje animacijama. Početni korak je izrada i uvoz modela. Kako bi izrada složenih scena bila što lakša, u ovom koraku moguće je "modelirati" najosnovnije animacije (eng. *animation clips*) unutar programa za modeliranje (*Blender*, *Maya* i drugi). Neke od tih animacija su animacija za stajanje (*Idle*), hodanje i slično. Model sa osnovnim animacijama preuzet je kao besplatni dodatak preko *Asset Store*-a. Sljedeći korak je podešavanje modela unutar *Mecanim*-a. U ovom koraku moguće su dvije opcije : podešavanje čovjekolikog modela (eng. *Humanoid character setup*) te podešavanje generičkog modela (eng. *Generic character setup*). Završni korak je oživljavanje lika što uključuje podešavanje osnovnih animacija (eng. *animation clips*), podešavanje izmjena animacija te upravljanje animacijama unutar izvornog koda.

Za animiranje modela, mora se konstruirati *Animation Controller* koji služi za podešavanje animacija te upravljanje animacijama unutar skripte. *Animation Controller* koristi automat stanja u kojem pojedina stanja predstavljaju osnovne animacije (eng. *animation clips*). U automatu stanja povezuju se animacije kako bi prijelaz među njima bio što prirodniji.



Slika 5.3.6: Prikaz automata stanja

Skripte koristimo za upravljanje animacijama unutar scene (igre). Unutar skripte možemo definirati, tj. kreirati događaje koji se izvode kada napravimo određenu akciju ili kada stisnemo određenu tipku na tipkovnici ili mišu.

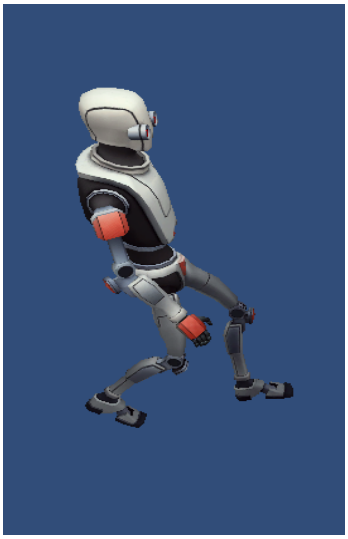
```
var anim : Animator;
var animSpeed : float = 1.5f;

function Start () {
        anim = GetComponent<Animator>();
}

function Update () {
        var h : float = Input.GetAxis("Horizontal");
        var v : float = Input.GetAxis("Vertical");
        anim.SetFloat("Speed", v);
        anim.SetFloat("Direction", h);
        anim.speed = animSpeed;
}
```

Prethodni izvorni kod pokazuje kako možemo upravljati animacijom. Varijabla **h** predstavlja unos koordinate po horizontalnoj osi koja nam služi za unos smjera kretanja (lijevo ili desno) dok nam varijabla **v** predstavlja unos koordinate po vertikalnoj osi a služi za kretanje prema naprijed ili prema nazad. U automatu stanja, kod podešavanja animacija, podešeno je da animacija hodanja ovisi o parametru **Speed** koja određuje da li je hodanje naprijed ili nazad, ovisno o unesenom iznosu. Također, podešen je prag (eng. *Threshold*) parametra **Speed** kako bi automat stanja znao kada treba prijeći u koju animaciju.

Slika 5.3.7: Hodanje lika naprijed



Slika 5.3.8: Hodanje lika unazad

6. Zaključak

Izrada animacija, ma koliko god jednostavne one bile, iziskuje dosta vremena i znanja. Najvažnije je pronaći adekvatan alat kojim će biti moguće ostvariti sve zacrtane želje i zadatke. U ovom radu korišten je programski pogon Unity, koji se može pronaći u dvije verzije - besplatnoj i licenciranoj. Iako besplatna, korištena verzija sadrži sve osnovne pa i napredne funkcije pogona. Jedina mana besplatne verzije je pojava tzv. vodenog žiga (eng. *watermark*) unutar izrađene igre te pojava ikone Unity-a na samom početku igre.

Iako naizgled intuitivan, za snalaženje unutar pogona potrebno je dosta vremena i znanja. Izrada animacija korištenjem skripti iziskuje kako vještinu rukovanja modelima tako i dobro poznavanje funkcionalnosti i mogućnosti Unity-a. Sami jezici za pisanje skripti jednostavni su za korištenje, no ako želimo izraditi nešto konkretno i napredno potrebno je veće znanje struktura podataka i ugrađenih funkcija Unity-a.

LITERATURA

[1] Panian, Ž., Informatički enciklopedijski rječnik, Zagreb: Europapres holding, 2005.

[2] Wikipedia, Computer graphics
http://en.wikipedia.org/wiki/Computer_graphics, svibanj 2013.

[3] Wikipedia, Animation
http://en.wikipedia.org/wiki/Unity_(game_engine), svibanj 2013.

[4] Wikipedia, Computer-generated imagery
http://en.wikipedia.org/wiki/Computer-generated_imagery,
svibanj 2013.

[5] Wikipedia, Computer animation
http://en.wikipedia.org/wiki/Computer_animation, svibanj 2013.

[6] Wikipedia, Unity - game engine
http://en.wikipedia.org/wiki/Unity_(game_engine), svibanj 2013.

[7] Unity, Manual
http://docs.unity3d.com/Documentation/Manual/index.html,
svibanj 2013.

[8] Unity, Components
http://docs.unity3d.com/Documentation/Components/index.html,
svibanj 2013.

[9] Unity, Script Reference
http://docs.unity3d.com/Documentation/ScriptReference/index.html, svibanj 2013.

[10] Unity, Scripting
http://docs.unity3d.com/Documentation/Manual/Scripting.html, svibanj 2013.

[11] Unity, Transform
http://docs.unity3d.com/Documentation/\linebreakScriptReference/Transform.html, svibanj 2013.

SAŽETAK

Animirani objekti u okruženju Unity

Težnja ovog rada je upoznavanje s konceptom animacije i razvijanje istih uz pomoć grafičkog pogona Unity. Prvo poglavlje predstavlja kratak uvod u samu temu završnog rada. U drugom poglavlju dan je pregled povijesti razvoja animacije te primjerci raznih metoda izrada animacija. Treće poglavlje donosi kratak uvod u povijest pogona Unity. Nakon predstavljanja sučelja pogona te njegovih osnovnih funkcionalnosti, četvrto poglavlje daje opis osnovnih geometrijskih transformacija koje predstavljaju temelj svih animacija. Također su prikazane strukture i funkcije spomenutih transformacija u samom pogonu Unity. Završno, peto poglavlje donosi programsko ostvarenje teme rada. Dani su primjeri osnovnih animacija u Unity-u.

Ključne riječi : Unity3D, animacija, 3D objekti

ABSTRACT

Computer animation in Unity

The main goal of this paper is to introduce the concept of animations and their development with the help of graphic software Unity. The first chapter presents a brief introduction to the topic of the thesis. The second chapter gives an overview of the history of development of animation and examples of various methods of animation. The third chapter provides a brief introduction to the history of the software Unity. After presenting the interface and its basic functionalities, the fourth chapter describes the basic geometric transformations that are the basis of all animations. Also there are structures and functions of these transformations shown in the software Unity. Finally, the fifth chapter provides program solution of the topic. Examples are given of basic animations in Unity.

Keywords : Unity3D, animation, 3D objects

DODATAK A

HOLD	+	Key	Function
File			
CTRL		**N**	New
CTRL		**O**	Open
CTRL		**S**	Save
CTRL	SHIFT	**S**	Save Scene as
CTRL	SHIFT	**B**	Build
CTRL		**B**	Build and run
Edit			
CTRL		**Z**	Undo
CTRL		**Y**	Redo
CTRL		**X**	Cut
CTRL		**C**	Copy
CTRL		**V**	Paste
CTRL		**D**	Duplicate
SHIFT		**Del**	Delete
		F	Frame (centre) selection
CTRL		**F**	Find
CTRL		**A**	Select All
CTRL		**P**	Play
CTRL	SHIFT	**P**	Pause
CTRL	ALT	**P**	Step
Assets			
CTRL		**R**	Refresh
Game Object			
CTRL	SHIFT	**N**	New game object
CTRL	ALT	**F**	Move to view
CTRL	SHIFT	**F**	Align with view
Window			
CTRL		**1**	Scene
CTRL		**2**	Game
CTRL		**3**	Inspector
CTRL		**4**	Hierarchy
CTRL		**5**	Project
CTRL		**6**	Animation
CTRL		**7**	Profiler
CTRL		**9**	Asset store
CTRL		**0**	**Asset server**
CTRL	SHIFT	**C**	Console
CTRL		**TAB**	Next Window
CTRL	SHIFT	**TAB**	Previous Window
	ALT	**F4**	Quit

HOLD	+	Key	Function
Tools			
		Q	Pan
		W	Move
		E	Rotate
		R	Scale
		Z	Pivot Mode toggle
		X	Pivot Rotation Toggle
CTRL		**LMB**	Snap
		V	Vertex Snap
Selection			
CTRL	SHIFT	**1**	Load Selection 1
CTRL	SHIFT	**2**	Load Selection 2
CTRL	SHIFT	**3**	Load Selection 3
CTRL	SHIFT	**4**	Load Selection 4
CTRL	SHIFT	**5**	Load Selection 5
CTRL	SHIFT	**6**	Load Selection 6
CTRL	SHIFT	**7**	Load Selection 7
CTRL	SHIFT	**8**	Load Selection 8
CTRL	SHIFT	**9**	Load Selection 9
CTRL	ALT	**1**	Save Selection 1
CTRL	ALT	**2**	Save Selection 2
CTRL	ALT	**3**	Save Selection 3
CTRL	ALT	**4**	Save Selection 4
CTRL	ALT	**5**	Save Selection 5
CTRL	ALT	**6**	Save Selection 6
CTRL	ALT	**7**	Save Selection 7
CTRL	ALT	**8**	Save Selection 8
CTRL	ALT	**9**	Save Selection 9

Slika 1: Unity kratice na tipkovnici za platformu Windows