SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

MARINA POKRAJAC

3D ANIMACIJA I OPEN SOURCE

DIPLOMSKI RAD

Zagreb, 2015

Sveučilište u Zagrebu
Grafički fakultet

MARINA POKRAJAC

3D ANIMACIJA I OPEN SOURCE

DIPLOMSKI RAD

Mentor:

Izv. profesor doc.dr.sc. Lidija Mandić

Profesor Luca de Mata

Student:

Marina Pokrajac

Zagreb, 2015

Rješenje o odobrenju teme diplomskog rada

Željela bih se zahvaliti svojoj mentorici doc.dr.sc. Lidiji Mandić i ko-mentoru profesoru Luci de Mati, kolegama iz L & C-a te svojoj sestri Mariji Pokrajac i roditeljima za pomoć i podršku tijekom izrade ovog rada. Bez njih ovaj rad ne bi bio moguć. Hvala vam.

Sažetak

3D animacija zadnje je desetljeće doživjela ekspanziju-kako u kvaliteti tako i u primjeni. Danas postoje razni softveri koji veoma realistično stvaraju 3D modele i animacije. Sam proces stvaranja 3D animacije sastoji se od više faza (planiranja, dizajniranja, modeliranja, teksturiranja, postavljanja kamera i rasvjete, animiranja, renderiranja, *compositinga* itd.) koje su neophodne za jednu uspješnu produkciju. Iako sveprisutna, 3D animacija je tehnički i ekonomski veoma zahtjevna, kako zbog znanja i vremena potrebnog za njenu provedbu, tako i zbog visokih cijena softvera koji su za to potrebni. Stoga se postavlja pitanje može li jedan besplatan i *open source* softver proizvesti dovoljno kvalitetan sadržaj koji je u stanju konkurirati ostalim tzv. *closed source* softverima. U ovom se rada predstavila mogućnost izrade jedne 3D animacije u *open source* programu i istražile su se njene mogućnosti i prednosti, te se na primjeru njezine izrade se opisao proces stvaranja CGI (3D) animacije od samog početka pa sve do kraja. *Open source* ili otvoreni kod je softver čiji je izvorni kod ili nacrt dostupan na uvid, korištenje i daljnje dijeljenje/distribuciju. *Open source* softver korišten za svrhe ovog rada je Blender. Svake se godine usavršava i nadograđuje te čak i sam korisnik može sudjelovati u njegovoj izradi. Ovaj rad je za cilj imao približiti način rada u *open source* softveru, istražiti njegove mogućnosti i kako one utječu na budućnost animacije. Isto tako, u ovom radu savladala su se neka od osnovnih pravila, vještina i organizacija rada potrebnih za kompletnu produkciju jedne 3D animacije.

Ključne riječi:

3D, Animiranje, 3D animacija, Open-source, Blender

Abstract

Over the last decade, 3D animation has undergone a huge expansion in quality and in its application. Today there are various kinds of softwares that can create very realistic 3D models and animations. The process of creating 3D animation consists of several stages (planning, designing, modeling, texturing, camera and lighting imposing, animation, rendering, compositing and so on) that are necessary for a successful production. Although omnipresent, 3D animation is still technically and economically very challenging, partly because of the knowledge and the time required for its implementation, but also because of the high prices of the softwares. The question this thesis shall answer is if free and open source software is capable of producing good quality content that may compete with other so-called closed source softwares. This paper has introduced the possibility to create a 3D animation in an open source program and has explored its features and benefits. In the process of making one it has been explained how to make a 3D CGI animation from scratch till the very end of production. Open source software is when the source code or design of the software itself is available to the general public for viewing, usage and further distribution. Software used for the purposes of this paper is called Blender. Each year, this software has been improved and upgraded, and users can even participate in its development. This thesis strives to explain how open source software works, to explore its capabilities and how they affect the future of animation. Also, in this paper, the basic rules, skills and work organisation required to complete a production of a 3D animation are explained and mastered.

Keywords:

3D, Animation, 3D animation, Open-source, Blender

Sadržaj

Uvod

Ovaj rad ima za cilj ne samo upoznati svijet 3D-a i 3D animacije i pokazati kako napraviti jednu 3D animaciju u *open source* softveru već ima i približiti svijet *open sourcea* i *open source* softvera sa svrhom kreiranja 3D animacija. Raščlanila se politika i problematika *open sourcea* koja svoje korijene povlači iz pokreta slobodnog softvera (eng. *free software movement*). Za shvaćanje ideje i svrhe *open sourcea* bitno je razumjeti razlike između *open* i *free* softvera, te kakvo značenje te riječi nose. Ne treba miješati ta dva pojma jer predstavljaju dvije stvari koje su utemeljene na donekle različitim idejama (iako u nekim segmentima vrlo sličnim). Pobornici jedne ne moraju nužno biti ili čak uopće nisu, pobornici druge. *Slobodno* odnosno engleski *free* ne znači uvijek besplatno. Ni kod *open sourcea* otvoreno ne znači da ne postoje nikakva uređena pravila koja definiraju tu slobodu. S obzirom da su i *free* i *open* softveri, uglavnom, zakonski pokriveni istim ili vrlo sličnim licencama, dotaknulo se i vrlo kompleksno pitanje licenci i autorskih prava. Licencama je definirano radi li se o softverima otvorenog izvornog koda, odnosno *open source* softverima ili o *closed source* softverima. Upravo njima se neki softver oslobađa ili ograničava. Licence su te koje ideje i kriterije *open source* i *free* softvera sprovode u praksu. Upravo kako bi se razumjeli ovi termini neophodno je minimalno znanje o načinu na koji su računalni programi, odnosno softveri izgrađeni, te što čini i predstavlja jedan softver. *Open source* ili na hrvatskom jeziku **otvoreni izvorni kod** jest upravo ono što i sam naziv govori. Znači da izvorni kod nekakvog programa tj. softvera jest otvoren, odnosno javno dostupan svima na uvid. I ne samo na uvid, već ga je dozvoljeno modificirati, distribuirati i tako dalje. U ovom se radu analiziralo zašto bi neki softver trebao biti otvoreni i kakve to posljedice nosi - pozitivne ili negativne. Praksa je do sada pokazala da su zahvaljujući pokretu otvorenog izvornog koda širom svijeta pokrenute mnoge inovacije i kreacije koje su na ovaj ili onaj način pomogle društvu ili pojedincima. Pokret otvorenog izvornog koda potiče na međusobnu suradnju i pomaganje sa svrhom zajedničkog napretka, što se u mnogočemu kosi s dosadašnjom

praksom izdavača i autora komercijalnih *closed source* proizvoda i softvera. Pritom se misli na softvere i proizvode koji su zaštićeni *copyright* licencama koje onemogućavaju ili znatno otežavaju bilo kome da iz njihova koda uči ili razvije bilo kakve inovacije bez dopuštenja. Iz ovih razloga, i sam osnivač Blendera je tadašnje softvere smatrao nedostatnim za svoje potrebe, te sam odlučio pokušati riješiti probleme s kojima se susretao. U slučaju Blendera ponudio je i svima ostalima da javno sudjeluju i pomažu u izgradnji novog softvera. Ideja je to kojoj su se mnogi pridružili, a živi do današnjeg dana. Važno je reći da je takva pristup rezultirao razvojem snažne, podržavajuća i kolaborativna *online* i *offline* zajednica. Animiranje u *open* i ostalim komercijalnim softverima u bazi se razlikuje vrlo malo ili nimalo. Kako bi se stvorio jedan animirani film moraju se proći određene faze. Varijacije su male i ovise samo o odabranim digitalnim medijima u kojima se sprovode. U ovom su se radu duboko proučavali procesi koji slijede prije, tijekom i nakon samog animiranja kako bi se savladala produkcija 3D animacije. **Animiranje** je samo jedna od mnogih faza koje se moraju savladati da bi nastala animacija. Prije animiranja mora se isplanirati čitava produkcija, napraviti *storyboard*, razviti *character design* i definirati ostale objekte na sceni. U fazi planiranja odnosno pripreme za animiranje još uvijek se ne modelira u 3D softveru, no ono je ipak neophodno za stvaranje kvalitetnog i promišljenog proizvoda. Nakon faze planiranja slijedi faza konstrukcije; to je ujedno i faza u kojoj se započinje koristiti **3D softver**, ovdje se izrađuju 3D modeli i sve ostalo potrebno da bi se ti modeli zgotovili, a to je teksturiranje i *rigging.* 3D modeliranje se može objasniti metaforom i reći da je poput kiparenja te je za njega je potrebno vremena, strpljenja i iskustva. Umjesto krutog kamena oblikuje se geometrijska mreža (eng. *mesh*) sastavljena od ploha (eng. *faces*), uglova (eng. *edges*), i vrhova (eng. *verticies*). U ovoj je fazi također važno odrediti parametre kamera i (po mogućnosti) rasvjete na sceni, na način na koji su definirani u *storyboardu.* Svaka izmjena je moguća, ali znatno usporava i otežava vremensku optimizaciju cjelokupne produkcije. Nakon što su svi parametri postavljeni i svi 3D modeli tehnički i umjetnički definirani, animiranje može biti započeto. Dakako, ono može biti znatno olakšano ako su sve prijašnje faze korektno odrađene. Računalo umjesto animatora može učiniti

dosta toga, ali ipak ne može sve. Ljudski ''dodir'' je neophodan da bi se postigla realističnost, ali prije svega uvjerljivost lika koji se animira. Animiranje je faza sama za sebe, bez podfaza i kao takva je vremenski najzahtjevnija. Teoretski gledano, način na koji se izvode pokreti, to jest, animiranje nije se mnogo promijenio od samih početaka animacije do danas, no mediji s kojim se to izvodi jesu. Za dobro animiranje (lat. *animatio* znači darivanje života, živo biće) odnosno za ''udahnjivanje'' života, osim iskustva i vježbe potrebno je i dobro izvježbano promatračko oko. Nakon finalnih rendera, ako je to potrebno, produkcija može dalje prijeći u *compositing*. Ako finalnu animaciju nije potrebno dodatno poboljšati nekakvim specijalnim efektima, prijelazima iz kadrova, mijenjanjem tonaliteta ili kontrasta i tako dalje animacija može biti pripremljena za eksport u finalni format. Ponekad se scena razdjeljuje na slojeve, radi bržeg renderiranja, što omogućuje ponešto veću manipulaciju u *compositingu*, pa je potrebno prije finalnog eksportiranja sve dijelove ujediniti u jednu cjelinu. Može se reći da je svrha 3D modeliranja u animaciji, na koncu, iako se govori o trodimenzionalnom prostoru, doduše virtualnom, stvaranje dvodimenzionalnih (2D) slika, jer to je zasad jedini mogući način na koji 3D objekti mogu biti prikazani na ekranu/platnu. Granica između 2D grafike i 3D grafike ponekad može biti tanka, pa iz tih se razloga pobliže pozabavilo pitanjem 3D i njegovom pozicijom u svijetu računalne grafike. Sve ono što se teoretski raščlanilo i proučilo u prvom dijelu rada, u drugom, eksperimentalnom dijelu rada, sprovelo se i u praksi. Proučavanjem slika i opisa mogu se iščitati urađeni koraci koji su potrebni za sprovođenje jedne produkcije, od 3-4 minute CGI animiranog filma do samog kraja. U ovom radu suočeni su *open source* i 3D animacija korištenjem praktičnog primjera. Na koncu, korišteni softver je uspoređen s ostalima na tržištu te su prokomentirane njegove mogućnosti, pozitivno i negativne posljedice te budućnost CGI animacije u *open source* softverima.

1. Uvod u *open source*

Da bi se neki program smatrao *open sourceom*, njegov izvorni kod mora biti slobodno dostupan svima. Svi moraju imati mogućnost uzeti taj kod, modificirati ga i prosljeđivati vlastite varijante tog istog programa. Korisnici također moraju imati mogućnosti dijeljenja kopija originalnog programa u neograničenim količinama. Program se mora moći koristiti u bilo koje svrhe i ne smije biti nikakvih pristojba za licence ili drugih restrikcija prema softveru.

Closed-source program ima suprotne značajke – posjeduje licencu koja pravno ograničava korisnika te mu iznimno otežava ili ne dozvoljava uvid u originalni kod. Također, ne mora značiti da *open source* proizvodi (softveri) moraju uvijek biti u potpunosti besplatni. U sljedećim poglavljima obrazloženo je što točno znači i predstavlja termin *open source* kao i *open source* pokret i njegova online zajednica. Kako bi se bolje razumio koncept otvorenog koda i ostvarila konkretnija cjelokupna slika, ukratko se objasnilo što je *software* (hrv. softver, dalje u tekstu), na koji način je izgrađen, kako funkcionira, te kako može postati *open*. Osvrnulo se i na licence koje definiraju slobode i ograničenja korisnika i autora/kreatora *open* i *closed* softvera te na problematiku koja se tu javlja, jer iako govorimo o kreativnim i stvaralačkim "slobodama", i one se moraju na neki način definirati da bi funkcionirale kao takve.

1.1. Softver

Prije nego se detaljnije upusti u definiranje i proučavanje onoga što zapravo *open source* predstavlja; *open source* softvera, te generalno softvera koji se koriste za 3D modeliranje i animaciju, bilo bi dobro razjasniti što u biti jest softver. Ukratko se je definirano i razjašnjeno što on jest, čemu služi, te kako funkcionira. Sve to radi stjecanja jasnije slike i boljeg shvaćanja sveukupne ideje *open sourcea.*

Softvera jest bilo kakav set instrukcija koje su računalno-čitljive te kao takav upućuje procesor računala kako da izvede određene operacije. Računalni softver sastoji se od *hardware*-a koji je fizička komponenta računala, dok softver čini onaj neopipljivi dio računala kao što su operacijski sustav, programi i podaci koji se nalaze na računalu, dok hardware[1] predstavlja mehaničke i elektroničke dijelove računala, to jest one opipljive.

Hardware i *software* se ne mogu koristiti jedan bez drugoga. To znači da softver ne može izvršiti zadatke koji su mu zadani ukoliko ne postoji *hardware* koji će to omogućiti. Računalni softver sastoji se od računalnih programa, datoteka i drugih pridruženih dokumenata. Najjednostavniji nivo izvršnog koda sastoji se od instrukcija napisanih nekim od računalnih jezika[2]. Dakle, softver upravlja hardverom, obično softver sa tvrdog diska se učitava na RAM memoriju odakle se prosljeđuje procesoru koji izvršava naredbe koje sadrži neki program. Upravo zbog ovog procesa u svijetu 3D-a i animacije RAM memorija nekog računala predstavlja važnu stavku zato što 3D alati i softveri trebaju mnogo RAM memorije da bi u što kraćem vremenu mogli izvršiti neke radnje koje u pojedinim slučajevima mogu biti izrazito zahtjevne za izračun na računalu. Najniži nivo softvera jest binarni kod, najjednostavniji oblik programa koji je obično teško izmijeniti. Zbog toga se softveri češće pišu u programskim jezicima višeg nivoa koji su mnogo razumljiviji od binarnog koda[3]. Za provođenje

[1] Hrv. sklopovlje, u nastavku teksta hardver
[2] Računalni jezik sastoji se od binarnih vrijednosti.
[3] Binarni sustav predstavlja pozicijski brojevni sustav s bazom 2. To znači da u tom brojevnom sustavu za označavanje brojeva koristimo 2 znamenke, i to: 0 i 1. Kako je to brojevni sustav s najmanjom bazom, iz naziva njegove znamenke na engleskom jeziku BInary digiT nastalo je ime za najmanju količinu informacije BIT. Široko se koristi u tehnici, budući da je potrebno

programskih naredbi koristi se *compiler* koji ponovno taj kod prevodi u najniži računalni kod koji računalo razumije, to jest ponovno u binarni (0 i 1). Kada se izvede ovaj korak, izvorni kod u kojem je program napisan gotov je nemoguće iščitati, pa se može reći da je ovaj proces gotovo ireverzibilan, i upravo je to jedna od najznačajnijih razlika između *open source* i *closed-source* softvera, što se detaljnije razjašnjava u nastavku rada.

Ovisno o svrsi kojoj služe, postoje razne vrste softvera, pa ih se tako može razvrstati u sljedeće skupine:

Sistemski softver: to su programi bitni za rad računala i pomažu u njegovu radu. U ovu skupinu spadaju operacijski sustav, izbornik i driveri uređaja.

Aplikacijski softver: to su programi koji izvršavaju po jednu ili više uskih zadaća. U ovu skupinu, npr. spadaju baze podataka i računalne igre.

Programski softver: su programi koje developeri/programeri koriste za vrijeme stvaranja programa. *Notepad*[4] i programski prevoditelji su jedni od takvih.

S vremenom proizvođači izdaju novije i kvalitetnije inačice svojih softvera pa se tako rastućim brojevnim nizom (veći broj novija verzija) daje do znanja o kojoj se inačici radi. Ponekad se niz započne od 2.0 ili 3.0 ili se pak preskoči neki broj, o tome odlučuje proizvođač ako smatra da je softver već dovoljno unaprijeđen. Kada se radi o softveru otvorenog koda tada su dostupne razne verzije, od starih pa do novijih, onih stabilnijih provjerenih inačica, ali i beta verzije, tj. verzije koje još nisu stabilne jer nisu dovoljno ispitane, ali ih se može koristiti na vlastitu odgovornost, te ukoliko se uoči kakav problem može se dojaviti autorima (koji tako testiraju probnu verziju) i uz pomoć korisnika grade novi, bolji i stabilniji program. Takav pristup uglavnom nije praksa kod proizvođača komercijalnih softvera.

---

razlikovati samo dva stanja za prikaz znamenaka. Tehničke dobrobiti proizlaze iz pojednostavljenja sklopova i velike razine šuma koje uređaj može neometano podnositi. Stoga danas digitalni uređaji gotovo isključivo koriste binarni sustav. Posljedično je korišten u računalima, pa stoga i općenito u informatici i programiranju.

[4] Hrv. Obrađivač teksta

1.2. Politika *Open sourcea*

Generalno izraz *open source* ili otvoreni izvor se odnosi na bilo kakvu informaciju, bilo pisanu ili audio-vizualnu koja je slobodno dostupna javnosti. Kada se taj izraz upotrebljava uz neki softver tada se misli na njegov izvorni kod i/ili dizajn koji je dostupan javnosti na uvid, korištenje, izmjene i daljnje dijeljenje. Iako upotrebljavamo izraz *slobodan*, postoje različite vrste licenci koje detaljnije definiraju slobode i prava autora i korisnika softvera. Kod *open source* softvera licenciranje se uglavnom primjenjuje tako da autor zaštiti ime softvera i/ili da se kod daljnjeg modificiranja i dijeljenja uvijek naznači tko je autor/kreator prve verzije programa, dok se od korisnika uglavnom traži da se sačuva informacija o autorstvu. Neki od poznatih *open source* softvera danas jesu: *Firefox Mozzila, Mediawiki, Joomla, Linux* operativni sustav, *Apache*, *PNG*, *Arduino* matične ploče i tako dalje. Ukratko se može reći da je cilj pokreta otvorenog izvora da svi oni koji to požele mogu sudjelovati u razvoju neke aplikacije, neograničeno ju dalje distribuirati, raditi i koristiti njene modifikacije u svrhu naobrazbe ali i druge svrhe, sve to bez ikakvih ili minimalnih ograničenja (od strane opskrbljivača) koje se u tom slučaju ne odnose na samu aplikaciju.

Važno je razjasniti da kada se govori o slobodi slobodnih (eng. *free softwares*) i *open* softvera tada se ta sloboda odnosi na stvaralačku i kreativnu slobodu, a ne na financijsku slobodu u smislu besplatnosti. S obzirom da engleski jezik koristi riječ *free* za besplatno kao i slobodno važno je razjasniti na koji način se u ovom kontekstu ova riječ koristi. Može se reći da je *open source* pokret nastao više iz praktičnih, a manje zbog etičko-ekonomsko-zakonskih razloga. Fokus je na praktičnim i pragmatičnim prednostima zašto bi jedan softver trebao biti *open* ili *free*. Dakle, *open source software* ne znači nužno da je i besplatan softver. Na softveru se može moći ekonomski zaraditi ili ga prodavati, ali dokle god je njegov izvorni kod i dalje dostupan široj javnosti te je moguće dijeljenje i modificiranje, on jest i dalje *open source* softver. Vrlo često *open source* kompanije ili fondacije ostavljaju na korisniku na izbor da odluči želi li i koliko želi platiti za softvere putem donacija. U nekim su slučajevima nastale tvrtke koje prodaju *open* proizvode zato što garantiraju tehničku podršku korisnicima

koja je ujedno i najveća komercijalna slabost jednog *open source* proizvoda bez obzira što postoje *online* zajednice koje su se pokazale iznimno korisnima i aktivnima u slučaju ikakvih problema, no to ipak ponekad nije dovoljna garancija. Također, termin *open source,* danas se koristi za opisivanje raznih drugih aktivnosti i projekta koji nemaju nikakve veze sa softverom, pa tako nema ni izvornog koda pa se ne mogu ni primijeniti uobičajene licence. U ovom kontekstu termin *open source* više nema prvotno značenje već samo poziva na otvoreno sudjelovanje ili transparentnost. [1]

1.3. Prednosti *Open Sourcea*

Većina softvera koji su kupuju ili skidaju dolaze samo *u compiled ready-to-run* verziji (slob. prijevod; sastavljena i spremna za pokretanja verzija). Programski kod koji je developer napisao, poznat kao izvorni kod, da bi mogao biti prepoznat od strane računala mora ga se sprovesti kroz *compiler* (specijalni program koji "sastavlja" kod). Kod većine aplikacija, kad se jednom napravi njihova "sastavljena" – *compiled* verzija koda, iznimno ju je teško modificirati, te je gotovo nemoguće vidjeti kako je developer napisao različite dijelove programa. Većina proizvođača komercijalnih softvera vidi to kao prednost jer na taj način sprječava druge kompanije od kopiranja njihovog koda i mogućeg njegovog korištenja u konkurentskim proizvodima. Također, daje im određenu kontrolu nad kvalitetom i značajkama u određenog proizvoda.

Open source softver se bazira na potpuno suprotnim načelima. Izvorni kod nalazi se s *kompiliranom* – sastavljenom verzijom, a modificiranje i prilagodba se u biti potiču. Softverski *developeri* koji podržavaju ideju otvorenog izvora vjeruju da, ako se dopusti svima koji su zainteresirani za modifikaciju izvornog koda, aplikacija će dugoročno gledano biti bolja, korisnija i s manje pogrešaka.

Kako bi se neki softver smatrao *open source* određeni kriteriji moraju biti zadovoljeni; program mora biti dostupan, to jest mora se moći slobodno distribuirati, izvorni kod mora biti uključen uz kompiliranu verziju koda, svakome

mora biti dopušteno da taj kod modificira, modificirane verzije mogu se ponovno dijeliti, licenca softvera ne smije zahtijevati isključivanje drugog softvera ili ometati njegove operacije. [1] Na primjer, originalni developer se odlučuje podijeliti izvorni kod s javnosti, te tako omogućiti drugim developerima da naprave modifikacije i implementiraju poboljšanja, koja kasnije mogu biti nadograđena u prvu verziju softvera koji s novim poboljšanjima može biti pušten i u službeni optjecaj.

Ono što uglavnom zabrinjava korisnike, jedan je od razloga zašto *open source* softveri nisu u jednakoj mjeri rašireni kao i komercijalni "ne-*open source* softveri jest to da *open source* softverima manjka garancije i tehničke podrške. Upravo zbog toga što se potiče modificiranje i prilagođavanje, a organizacija podrške je gotovo u cijelosti prepuštena volonterima, gotovo je nemoguće imati pouzdanu tehničku podršku. Međutim, iako uglavnom nema službene tehničke podrške zajednica korisnika *open source* softvera veoma je otvorena za komunikaciju, diskusiju i zajedničko rješavanje problema. Tako da će se na različitim forumima i stranicama uglavnom pronaći rješenje za moguće probleme. S obzirom da *open source* poziva na dijeljenje i razmjenu, korisnici *open source* softvera ne libe se podijeliti svoje kreacije, i s procesom nastanka, kako bi pridonijeli zajednici ili kako bi zajednica pridonijela njima, da bi se tako u kolaborativnom duhu napravilo neko djelo. Kod korisnika *closed-source* softvera to nije čest slučaj zato što oni uglavnom stoje iza ideja tih kompanija koja su izdale te softvere, pa tako ni otkrivanje procesa kreacije (već samo prezentacije konačne verzije) nije učestala praksa. Što više, zbog toga što je softver plaćen pa se i uradci njime napravljeni žele naplatiti kako bi bili isplativi, dok kod *open source* softvera financijski balans nije nužno jedno od glavnih vodilja pri stvaranju nekog projekta.

1.4. Slobodan Softver

Slobodan softver (eng. *free softver*) je produkt koji stavlja u praksu ideologiju slobodnog pokreta to jest pokret slobodnog softvera bori se kako bi za korisnike računala izborio slobode koju donosi slobodni softver. Slobodan softver stavlja korisnika u poziciju da sam odlučuje o tome što želi s danom slobodom odnosno na koji način upravljati danim softverom. Kod ne-slobodnih softvera kontrola nad softverom nije u rukama korisnika već u rukama developera. Najjednostavnije rečeno *free* softver znači da korisnik ima pravo koristiti, kopirati, raspačavati, učiti, promijeniti i poboljšati sam softver. Važno je ponovno naglasiti da kada govorimo o slobodi tada mislimo na slobodu korisnika da softver koristi, kopira, izmjenjuje i tako dalje, a ne o besplatnosti ili cijeni. Dakle, ovdje govorimo o etičkim slobodama a ne o doslovnim slobodama.[5] Može se reći da je glavni moto pokreta slobodnog softvera cijeniti slobodu koju nam taj softvera omogućava zbog slobode same, a ne samo isključivo zbog tehničke i praktične pogodnosti koje ta sloboda nosi. Pokret *open sourcea* naime baziran se upravo oko tih praktičnih pogodnosti. Točnije rečeno *free* softver znači da korisnik programa ima 4 ključne slobode:

- Slobodu da koristi softver za bilo koje svrhe i na bilo koji željeni način.
- Slobodu da proučava kako program radi, te da ga adaptira svojim potrebama. Pristup izvornom kodu je preduvjet.
- Slobodu da distribuira kopije, kako bi mogao pomoći drugima.
- Slobodu da poboljša program i izbaci novu poboljšanu verziju u javnost, tako da se cijela zajednica može okoristiti njime. Pristup izvornom kodu je uvjet za to. [1]

[5] Engleska riječ *free* može značiti slobodno ali i besplatno. Stoga je važno razjasniti na što se točno misli kada se koristi riječ *free* za *free* softver, te u kojem ju kontekstu treba prevesti. S obzirom da u hrvatskom jeziku ne postoji taj problem, u engleskom jeziku može doći do polemika i krivog shvaćanja ideologije *free* softvera. Na koncept slobodnog softvera treba gledati kao na slobodu govora, a ne kao na besplatno (*free*) pivo kako to pobornici često vole objašnjavati.[1] Kako bi se pokušala raščistiti pomutnja oko toga što riječ “slobodan” u slobodnom softveru znači, s obzirom da engleski jezik je u tom pogledu ograničen, pokušalo se s raznim alternativama, poput francuske i španjolske riječi *libre*, ali ipak nije se pronašlo konačno zadovoljavajuće rješenje. U hrvatskom jeziku zadržan je prijevod slobodan softver jer ipak najtočnije opisuje koncept i ideologiju *free* softvera od ostalih sinonima.

Zbog razvoja u tehnologiji i *networking*-u (hrv. umrežavanju) te slobode postaju još preciznije od kada su postavljene 80-ih godina kada su se razvijale zajedno s GNU operativnim sustavom. Pokret slobodnog softvera od tada je znatno uznapredovao.

Dakle, gore navedene slobode trebaju biti omogućene ako neki softver želimo nazvati slobodnim. Važno je ponoviti da je slobodan softver primarno fokusiran na korisničke slobode, a manje na ekonomsko-pravne slobode. Iz tog razloga te slobode moraju biti moguće bez obzira jesu li programi izmijenjeni ili ne, te da li se traži ili ne traži naknada za njihovu distribuciju ili korištenje. One samo moraju biti omogućene svima i svugdje. Ukratko to znači da za potraživanje tj. korištenje tih sloboda nije prethodno potrebno tražiti dopuštenje ili platiti odobrenje, niti imati obvezu spominjanja vlasnika/autora/tvrtki, ni obvezu ikoga upozoriti na objavljene izmjene koje su učinjene na programu. Korisnik je slobodan pokretati program u koje god svrhe mu trebaju (tržišna upotreba i distribucija uključena) i na kojim god operativnim sustavim te ga isto tako distribuirati nekom drugom te od njega ne zahtijevati nikakve uvjete za korištenje[6] (prodaja nije isključena). Ako izvorni kod nije jasan i čitljiv, tada to nije pravi izvorni kod i ne može se smatrati izvornim kodom, te samim time takav softver nije slobodan. Nije dovoljno da je samo izvorni kod dostupan, a druge slobode su onemogućene. Ako su licence odviše restriktivne ili onemogućuju potpune slobode korištenja i modificiranja softvera, softver se tada ne može smatrati slobodnim. Slobode moraju biti trajne i neopozive, sve dok su korištene u dobroj namjeri.[1]

Pokušali se ukratko pobliže raščlaniti 4 osnovne slobode, međutim njihovo objašnjenje je puno detaljnije i kompleksnije te da bi se ispravno shvatile i koristile zahtijevaju svjesno promišljanje. Ponekad su granice između toga zadovoljava li neki softver status slobodnog softvera ili ne tj. odgovara li njegova licenca precizno definiranim kriterijima slobodnog softvera veoma tanke. Važno

[6] Na primjer, copyleft je – vrlo pojednostavnjeno – pravilo koje vam pri distribuciji programa zabranjuje ograničavanje slobode drugima. Ovo pravilo nije u sukobu s osnovnim slobodama; zapravo ih štiti.

je za reći i to da se kriteriji koji definiraju slobodni softver konstantno ispravljaju i usavršavaju. Isto tako, kriteriji slobode kod slobodnih softvera mogu se primijeniti i na druge sfere djelovanja, a da se nužno ne radi o računalnom softveru. To mogu biti i djela koja utjelovljuju korisno znanje. Svaka vrsta djela može biti slobodna i podložna kriterijima slobode slobodnog softvera.

1.5. Softverske licence

Kada se govori o pravima i slobodama korisnika i/ili autora kod *open source* softvera ali i *closed-source* softvera mora se dotaći i teme softverskih licenci. Na koncu koliko je neki softver slobodan ili *open source* biti će definirano na ovaj ili onaj način nekom od licenci. Pritom ne treba miješati licenciranje i kriterije pokreta *open sourcea* koji bi trebali biti zadovoljeni da bi se neki softver mogao prozvati *open source* softverom. Kriteriji su linije vodilje dok su licence puno konkretnije zato što su zakonski pokrivene. Ako se kriteriji formuliraju u licencu postaju legalno važeći, pa se tako razvilo par vrsti licenci koje pogoduju *open source* i *free* ideologijama. Važno je napomenuti da su licence dokumenti koji dozvoljavaju i definiraju prava na korištenje, dakle, gotovo svi softveri su licencirani a ne prodani. Kupnjom softvera mi zapravo ne kupujemo softver već kupujemo prava na korištenje i to pod određenim uvjetima i u određenom vremenskom roku. Licence nam dopuštaju da koristimo softver na ne-ekskluzivnoj bazi pod određenim uvjetima. Softverska licenca ne predaje vlasnička prava korisniku po danoj cijeni, već je jedna vrsta jednokratnog iznajmljivanja za navedenu cijenu. U ovom poglavlju razjasnilo se što su softverske licence, koga zastupaju i čemu zapravo služe. Svi se softveri, slobodni i “ne-slobodni”, moraju na neki način zaštiti ili osloboditi licencama. Pa stoga, kako bi se ukompletirala sveopća sliku o *open sourceu* mora se proučiti i pokušati shvatiti na koji način se u njihovu slučaju primjenjuje licenciranje.

Softverska licenca je pravni dokument kojim je regulirano korištenje i distribuiranje softvera. S obzirom da su licence pravni dokumenti uvelike ovise i

zakonodavnom sistemu zemlje za koju su stvorene. Nije nužno da sve licence vrijede jednako svugdje u svijetu. Uglavnom su licence kod “ne-otvorenih”, “ne-slobodnih” komercijalnih softvera postavljene tako da štite prava autora u što većoj mjeri, a ne prava korisnika, za razliku od licenci otvorenih i slobodnih softvera. Gotovo svaki softver je zaštićen *copyright*-om ili *copyleft-om* (o čemu se pobliže govori u poglavlju 1.5.1.). Iz tih razloga licence se mogu podijeliti na **vlasničke licence** i na one **slobodnog** i **otvorenog koda**. Zajednička tema objema vrstama, ali ujedno i njihova glavna razlika jest njihov stav naspram korisničkih sloboda. To jest one definiraju uvjete na koji način korisnik može dalje distribuirati i/ili kopirati taj softver. S obzirom na pravnu prirodu licenci nije se detaljnije ulazilo u njihovo značenje, već se općenito prošlo neka od svojstva koja su uglavnom karakteristična za većinu licenci oba tipa. Postoje više varijacija sličnih licenci i ukratko su opisane neke od najraširenijih. Jedno od svojstva vlasničke softverske licence jest da neka od prava koja se odnose na softver su rezervirana samo za onoga koji je taj softver objavio. Za takve licence je tipično da definiraju koliko puta ga je dozvoljeno instalirati te pod kojim uvjetima ga je dozvoljeno (ako je dozvoljeno) distribuirati. U svakom slučaju vlasništvo nad softverom uvijek ostaje u rukama izdavača te ukoliko korisnik ne prihvaća sve uvjete u licenci nema pravo na korištenje tog softvera. Također, vlasničke licence uglavnom strogo zabranjuju *revers engineering (back engeniring)*[7] što s druge strane licence *open* i *free* softvera potiču. Također postoje dvije vrste *open source* licenci; one koje imaju minimalne zahtjeve kako neki softver treba biti distribuiran i nazivaju se **popustljive licence** (eng. *permissive licences*), te one koje žele zaštiti slobode korisnika tako što osiguravaju da i svi sljedeći korisnici imaju ista ta prava i te se nazivaju ***copyleft*** licencama. Primjeri za *copyleft* [1] licence su BSD i MIT licence, te jedna od

[7] Je proces ekstrakcije znanja ili informacija o dizajnu iz bilo koje ljudske kreacije/rukotvorine te re-produkcija ili ponovna proizvodnja koja se temelji na toj ekstraktiranoj informaciji. Proces uglavnom uključuje rastavljanje na mehaničke dijelove, računalne programe ili na kemijske i organske elemente i analiziranje tih dijelova u detalje. *Reverse engineering*-om uglavnom se NE krše nikakva vlasnička prava jer se svodi na traženje i rješavanje problema npr. U poslovnim modelima ali i u rješavanju zločina. Međutim, može se koristiti da se razotkrivanje dizajna nekog softvera kako bi se odstranila zaštita od kopiranja ili na temelju tog dizajna kreirala druga verzija. U tom slučaju *revers engineering* se kosi s vlasničkim pravima. U svakom slučaju u *revers engineeing* procesu cilj nije stvaranje kopija ili mijenjanje uratka već je to deduktivna analiza pojedinih značajki dizajna od samog proizvoda, s što manjim ili nikakvim dodatnim znanjem o proceduri kreiranja originalnog uratka.

najpoznatijih *copyleft* licenci je GNU GPL licenca. Moguće je i dualno licenciranje softvera tj. softver podliježe dvjema licencama. Što se tiče Hrvatske, one uglavnom prate američke i europske modele. Tvrtke i osobe koje nemaju licence za softvere koje koriste ili ih nisu nabavile u određenim propisanim rokovima novčano se kažnjavaju te je moguće pokretanje parničnih i prekršajnih postupaka.[11] Isto tako, pružatelji usluga, odnosno izdavači ili autori proizvoda licencom se obvezuju na podršku korisnicima, to je uglavnom više slučaj kod *closed-source* licenci nego kod *open source* licenci. U nastavku, kratak opis nekih od najraširenijih licenci.

EULA je jedna od najraširenijih licenci *closed-source* softvera i kratica je za *end-user-licence-agreement* (slob. Prijevod; ugovor o licenci krajnjeg korisnika). Korisnik pristaje na plaćanje privilegije korištenja softvera, te potvrđuje da pristaje na ograničenja koja su detaljno definirana licencom. Obično prilikom instalacije softvera od korisnika se traži da potvrdi zahtjeve prije nego nastavi s instalacijom, ili pak čim fizički otvori pakiranje proizvoda ili korištenje proizvoda ili skidanje instalacijskog paketa itd. se podrazumijeva da je pristao na zahtjeve. Korisnik može i odbiti zahtjeve, pritiskom na ikonu “ne prihvaćam” ili jednostavnim nekorištenjem proizvoda ili traženjem povrata uloženih sredstava, logično u tom slučaju korisnik nema više prava da taj softver koristi.[10]

GNU GPL, kratica za GNU General Public Licence[8], je opća javna licenca koja je dizajnirana specifično za obranu slobode svih korisnika softvera. Ujedno je i *copyleft* licenca. Jedna od najraširenijih licenci za slobodan i *open source* softver. Korisnici imaju pravo na korištenje softvera u bilo koje svrhe (pa i one komercijalne), pravo na izradu kopija i pravo na proučavanje, mijenjanje i redistribuciju modificiranog programa. Isto tako sve verzije i modifikacije proizašle iz prvotnog softvera trebaju biti distribuirane pod istom licencom , to jest garantirati iste slobode, za razliku od MIT, BSD i Apache koje to ne zahtijevaju. Također, postoji više verzija ove licence poput, GPL version 3 (ujedno i najnovija), GPL version 2, LGP itd. GPL kod može biti prodan, ali iz njega ne smije proizaći nijedan vlasnički softver, iako softveri pod GPL licencom

[8] Originalno ju je kreirao Richard Stallman za projekt GNU, a o kojoj se danas brine Free Software fondacija.

mogu biti korišteni kao alati za stvaranje jednog vlasničkog softvera, na primjer *Wordpress i Blender* koriste ovu licencu. U praksi, mnogi GPL programi se distribuiraju putem interneta, te je izvorni kod objavljen preko FTP- ili HTTP-a, u skladu s licencom. Također GPL je pod *copyright*-om, što znači da nositelj licence nema pravo raspačavati ju ili modificirati osim pod uvjetima licence. [10]

MIT licenca stavlja djelo u javno vlasništvo[9]. Daje dopuštenje da bez restrikcija, uključujući i pravo na beskonačno korištenje, kopiranje, modificiranje, izdavanje, raspačavanje, sub-licenciranje i/ili prodaju kopija softvera pod jedinim uvjetom da se uključi ista *copyright* izjava. Djela pod ovom licencom mogu biti korištena za privatne i komercijalne svrhe. Postoje više varijacija ove licence.[10]

BSD Licenca je slična MIT licenci, osim što u jednoj svojoj stavci govori da ime originalnog vlasnika *copyright*-a se ne može koristiti kako bi se poduprli bilo koji derivati djela bez prethodnog dopuštenja. [10]

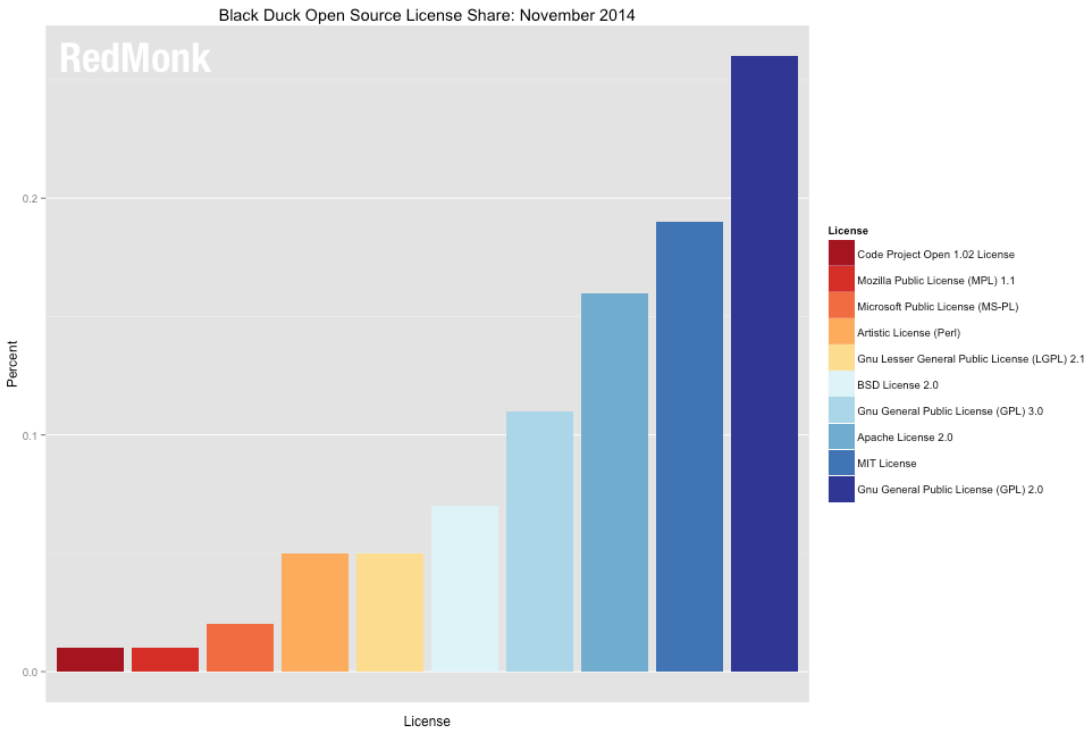
Apache Licenca je *free* softver licenca i kompatibilna je s GNU GPL (verzija 3) licencom. Također, ona ne zahtjeva primjenu iste licence za sve derivate prvotnog djela. To znači da kod pod ovom licencom može biti korišten u otvorenom, slobodnom i vlasničkom softveru (poput MIT i BSD licenci). Također, nameće pravilo da u bilo kojem licenciranom dokumentu (originalni *copyright*) obavijesti o zaštitnom znaku i pripisivanju prava moraju biti sačuvani. Sa svakim modificiranim djelom mora biti uključena obavijest o promijeni, te bilo koje već postojeće obavijesti moraju biti također uključene. Sve te obavijesti moraju biti distribuirane u tekstualnom dokumentu i u izvornom kodu ili dokumentaciji (to ju razlikuje od MIT i BSD licenci). [10]

Creative Commons (CC) licence stvorila je *Creative Commons* neprofitna organizacija koja omogućava dijeljenje i korištenje kreativnih radova i znanja kroz besplatne i legalne alate. Ova organizacija pruža besplatne i jednostavne za korištenje *copyright* licence koje na jednostavan i standardiziran način daju prava široj javnosti da dijeli i koristi nečiji kreativni rad, ali pod uvjetima koje je sam autor odredio, odabirom jedne od nekoliko ponuđenih verzija licenci. Ideja

[9] Javno vlasništvo (eng. public domain) je pravno gledano javno dobro, pa kao takvo ne podliježe niti zakonu o autorskim pravima niti zakonu o vlasništvu.

za ovakve licence nastaje jer klasičan *copyright* sistem nije bio odgovarajući za potrebe ovakvog tipa. S obzirom da on uglavnom u potpunosti ograničava širu javnost da koristi taj rad u bilo kakve svrhe bez odobrenja kreatora. *Creative commons* licence popunjavaju upravo tu prazninu gdje autor želi dati određena prava, za korištenje i dijeljenje, široj javnosti. Pa tako CC licence na jednostavan način dopuštaju da se običan *copyright* uvjet "sva prava zadržana" promijeni u "neka prava zadržana". CC licence nisu alternative *copyright*-u već one funkcioniraju paralelno s *copyright*-om, zato što omogućavaju prilagodbu *copyright* uvjeta ovisno o potrebi. Dakle, CC licence su podobne ako autor želi u nekoj mjeri dati za pravo široj javnosti da koristi i dijeli njegovo djelo ili pak modificira i doradi. [26]

Ovakav način licenciranja otvara ogromnu bazu kreativnih radova koji se smiju koristiti u razne svrhe. Od glazbe do umjetničkih dijela i fotografije. Samim time upotrebom tih radova pridonosimo i širimo zajednicu koja podržava ovakav način licenciranja. Na ovakav način se upravlja legalnom i tehničkom infrastrukturom koja maksimizira digitalnu kreativnost, dijeljenje i inovaciju. Ideja univerzalnog pristupa istraživanjima, edukaciji i kulturi je omogućio internet, no zakonodavni i socijalni sistemi ne dozvoljavaju uvijek da se takve ideje i ostvare. Ideja *copyright*-a nastaje davno prije interneta te se može reći da su te metode zaštite teško primjenjive pod današnjim okolnostima te ih se može smatrati nedostatnim.



Graf 1: *Grafikon u postocima prikazuje popularnost među open source licencama (izvor: http://redmonk.com/sogrady/2014/11/14/open-source-licenses/)*

U ovom radu nisu opisane sve licence koje postoje, jer ih ima iznimno mnogo i neke se razlikuju u nijansama, međutim ovo su neke koje nas se najviše dotiču s obzirom da je cilj shvatiti na koji način one utječu na *open source* softvere i njegove derivate te u konačnici i produkte tih softvera. Također, više pažnje je posvećeno shvaćanju *copyleft-a* i licenci koje pod njega spadaju nego *copyright* s obzirom na opširnost tematike pokušalo se ostati u okvirima glavne teme.

1.5.1. *Copyright vs. Copyleft*

Copyright zakoni su iznimno kompleksni. Svaka zemlja ima svoje *copyright* zakone odnosno zakone koji štite intelektualno vlasništvo[10], ali postoje i međunarodni ugovori koji tada *copyright* prava proširuju na više zemalja. Dakako, tako nešto ovisi i o tome koje su zemlje potpisnice takvih i sličnih konvencija[11], čime pristaju na međunarodne *copyright* zakone. Ipak, zakoni o zaštiti intelektualnog vlasništva u većini zemalja svijeta slijede iste principe i razlikuju se samo u nijansama. Zbog iznimne kompleksnosti nije se ulazilo u detalje nego se samo usporedio *copyright* i njegova kontra-inačica *copyleft*[12]. Kako bi se bolje razumjelo *copyleft* pokret površno se dotaklo objašnjenja *copyright*-a. Dakle, gotovo sve vlasničke licence su ujedno i *copyright* licence, dok gotovo sve slobodne i *open source* licence su *copyleft* licence. Kada se stvara neko autorsko djelo[13] automatski se posjeduje autorsko pravo za to djelo, ekskluzivno je pravo autora da posjeduje svoje djelo. Autor kontrolira tko još može koristiti njegov uradak i na koji način. Za legalno definiranje te kontrole služe licence koje postavljaju smjernice korištenja. Djela koja su zaštićena *copyright*-om također se znaju navoditi u hrvatskom jeziku i kao intelektualno vlasništvo. Važno je za napomenuti da ako je neko djelo napravljeno od pojedinca za neku tvrtku ili pod nekom tvrtkom, tada ta tvrtka polaže prava na *copyright* a ne sam autor djela.[11]

Copyleft je općenita metoda za oslobođenje programa (ili nekog drugog djela), koja zahtijeva da sve izmijenjene i proširene inačice programa budu isto tako slobodne. *Copyleft* kaže da svatko tko ponovno distribuira softver, sa ili bez izmjena, mora proslijediti slobodu za daljnje kopiranje i izmjenu. *Copyleft,* dakle,

[10] Zaštita intelektualnog vlasništva obuhvaća prije svega zaštitu tehničkih izuma (patenti), zaštitu komercijalnih oznaka (žigovi), zaštitu industrijskog dizajna, te zaštitu autorskih prava.
[11] Jedan od takvih je i Bernska konvencija o zaštiti književnih i umjetničkih djela. To je međunarodni ugovor koji uređuje zaštitu autorskih prava, prvobitno potpisan 1886. godine u Bernu, Švicarska. Preko 160 zemlja svijeta je potpisalo ovu konvenciju.
[12] . “Left” u “copyleft” nije referenca na glagol “ostaviti” (“to leave”) već samo na smjer koji je inverzni od “right”.
[13] Autorsko djelo je originalna intelektualna tvorevina iz književnog, znanstvenog i umjetničkog područja koja ima individualni karakter, bez obzira na način i oblik izražavanja, vrstu, vrijednost ili namjenu.

garantira da svaki korisnik ima istu slobodu. Razlika između *copyleft*-a i javnog vlasništva (eng. *public domain*) jest taj da stavljanjem nekog djela u javno vlasništvo, to jest, odricanjem autorskih prava, ljudima (javnosti) se dopušta da taj program (i njegova poboljšanja) dijele. Međutim, isto to se dopušta i ljudima koji ne podupiru *free softver* ideologiju te im se tako daje za pravo da taj softver pretvore u vlasnički softver. Dopušteno je raditi izmjene i distribuirati takav softver kao vlasnički proizvod. Oni koji prime taj softver u izmijenjenom obliku nemaju iste slobode kao i oni koji imaju originalni softver, zato što su u prijelazu na vlasnički softver uklonjene. Ono za što se *copyleft* zalaže jest upravo obrana tih sloboda. Tako da umjesto termina javno vlasništvo pobornici slobodnog softvera i GPL licenci preferiraju korištenje termina *copyleft*, koji je mnogo precizniji u obrani sloboda korisnika, odnosno primjenjuje se *copyleft* na softver. Ovdje se radi o igri riječi *copyright* i *copyleft*. Naime, copyleft je nastala od riječi *copyright*, kako bi se istakla drastična razlika između to dvoje. Ono što nije u mogućnosti učiniti copyright, čini to *copyleft*. Dok *copyright* zakonski pokušava zaštiti prava autora i izdavača, *copyleft* radi upravo suprotno štiti prava korisnika. Savršeno je jasno da jedan softver ne može biti u isto vrijeme pokriven *copyright*-om i *copyleft*-om, ali isto tako to su dva posve različita konteksta, te u nekim slučajevima kompanije se mogu odlučiti da je njihov proizvod pokriven jednom od copyleft licenci ali da je neki drugi dio tvrtke poput njenog imena ili vizualnih identiteta zaštićen *copyright*-om. Na *copyleft* se sam po sebi gleda kao poticaj za stvaranje novih slobodnih softvera ili za poboljšanje starih. Dakle, ako netko želi modificirati i/ili poboljšati taj program, *copyleft* mu za to i daje dopuštenje.

Da bi se program stavio pod *copyleft*, prvo se mora napraviti izjava da je program pod autorskim pravom *(*pod *copyright*-om) i tada se dodaju distribucijska pravila, koja su zakonski instrument koji daje svakome pravo da koristi, izmjenjuje i ponovno distribuira kod programa, ili bilo koji drugi program derivirani iz tog koda, ali samo ako su distribucijska pravila neizmijenjena. Dakle, tako i slobode postaju zakonski nerazdružive. U slučaju da ne budu ispoštovani uvjeti *copyleft* licence, moguća je zakonska tužba od strane originalnog autora pod *copyright* zakonom. Dakle, *copyleft* dobiva svoju legalnu

moć od korištenja *copyright*-a na softverskim programima. *Copyleft* je način korištenja autorskih prava nad programom, ne znači odustajanje od autorskih prava tj. odustajanje bi napravilo *copyleft* nemogućim. *Copyleft* je općeniti koncept, što znači da korištenjem određenih licenci koje podržavaju koncept *copyleft*-a, ga ujedno i možemo realno implementirati. *Copyleft* se primjenjuje samo onda kada osoba želi re-distribuirati program. Osoba je slobodna raditi modificirane verzije za sebe bez obveze da te promijene objavi, dokle god te modifikacije nisu dalje distribuirane. Također, vrlo važno je i to da se *copyleft* primjenjuje isključivo na softver a ne na njegove produkte (osim ako nije derivat tog softvera).

1.6. Razlika između *Open* i *Free* softvera

Zašto se toliko priča o slobodi? Na koji način paragonirati slobodan softver i softver izvornog koda? Stallman[14] smatra da su spomenute slobode iznimno važno ne samo zbog individualne koristi nego i zbog društva kao cjeline jer promoviraju društvenu solidarnost, tj. razmjenu i suradnju. S obzirom da naše društvo postaje sve više digitalizirano, slobodan softver, postaje temeljan za slobodu općenito. Stvar je slobode a ne cijene. Tvrdi da mnogi danas koriste slobodne softvere a da zapravo ne znaju etičke razloge zašto su ti softveri razvijeni, te zašto uopće postoji pokret slobodnog softvera.[24]

Pokret slobodnog softvera okvirno nastaje 80-ih godina prošlog stoljeća. Pa, tako početkom 80-ih razvojem slobodnog operativnog sustava GNU, kada je i kreirana i opća javna licenca (eng. *GNU GPL – General Public License*)[15] kreće priča o slobodnom i *open-source* softveru. Kako se nisu se svi pronašli u ideologijama slobodnog softvera pokrenut je novi pokret; pokret otvorenog izvornog koda (*open-source* movement). Termin *open-source* se koristi kako bi se razlikovao od ideologije slobodnog (*free*) softvera. Ta dva termina opisuju gotovo istu kategoriju softvera, ali im se gledišta temelje na različitim vrijednostima. Slobodni softver je društveni pokret, dok je *open source* metodologija razvoja. Termin "otvoreni izvorni kod" je ubrzo postao udružen sa idejama i argumentima temeljenima samo na praktičnim vrijednostima, kao što su izrada i posjed moćnog i pouzdanog softvera, i to postaje glavna misija svih *open source* softvera. Pobornici pokreta slobodnog softvera smatraju da, iako govore o istom ili vrlo sličnom softveru treba razlikovati slobodan i *open* softver zato što različite riječi prenose različite ideje. Pokret slobodnog softvera, predstavlja slobodu, dakle ne prihvaća uvriježeno stajalište *open source* softvera. Koliko se zna, sav postojeći slobodan softver bi se kvalificirao kao otvoreni izvorni kod. Gotovo sav softver otvorenog izvornog koda je slobodan

[14] Richard Stillman je računalni programer i aktivist za slobodni softver. Osnivač je *Free software foundation-a,* lansirao je *GNU projekt, razvio GNU compiler collection i GNU Emacs*, tvorac je i najpoznatije licence za slobodni software GNU GPL.
[15] Jedna od najčešćih licenca i *open source* i *free* softvera. Licenca dizajnira specifično za obranu slobode korisnika.

softver, ali postoje iznimke. Neke licence otvorenog izvornog koda su previše restriktivne, te prema tome se ne kvalificiraju kao slobodne licence. Također, mnogi proizvodi koji sadrže računala (uključujući mnoge *Android* uređaje) dolaze sa izvršnim programima koji odgovaraju izvornom kodu slobodnog softvera, ali ti uređaji ne dozvoljavaju korisniku da instalira izmijenjene verzije tih izvršnih datoteka; samo ovlaštena kompanija ima mogućnost da ga izmjeni. Te izvršne datoteke nisu slobodan softver iako je njihov izvorni kod slobodan softver. Kriteriji za otvoreni izvorni kod ne prepoznaju ovaj problem; oni su zabrinuti samo s licenciranjem izvornog koda, tvrdi Stillman.[24] Važno je naglasiti da i *open-source* i *free* softveri koriste GNU GPL licence. Većina licenci otvorenog izvornog koda se kvalificira kao licenca slobodnog softvera. Tako da ne možemo napraviti razliku između to dvoje samo na temelju licenci koje koriste. Dok pokret *open sourcea* cilja na to da se zajednički izgradi bolji softver koji će svi moći koristiti, *free* softver pokret upravo to smatra krivom ideologijom. Zato što u konačnici moćni i pouzdani softveri će ograničiti slobode korisnika. Pod pritiskom industrije i velikih kompanija sve je više softvera dizajnirano specifično da ograniči korisnika.[16] Za zaključiti jest da je pitanje o razlici otvorenog i slobodnog softvera etičke prirode. Dok je slobodan softver isključivo vođen etičkim i moralnim odgovornostima društva kao cjeline naspram slobodnog korištenja softvera. Slobodan softver mnogo dublje zalazi u pitanje sloboda i proširuje to pitanje na aspekte društva i širih digitalnih medija dok pokret OIK-a gleda na otvorenost s pragmatičnije strane, više se okreće praktičnim slobodama od kojih društvo ali ponajviše i pojedinac može profitirati.

[16] Poznato i kao Digitalno Upravljanje Restrikcijama (Digital Restrictions Management) (DRM), što je ujedno upravo suprotno od onoga što propagira pokret slobodnog softvera.

2. Uvod u računalnu grafiku

Kako bi se bolje shvatio svijet 3D-a i animacije, važno je definirati što zapravo u ovom kontekstu znače riječi *grafika* i *računalna grafika*. Može se reći da izraz računalna grafika obuhvaća gotovo sve vizualne tvorevine koje su napravljene pomoću računala, ili sve što nije zvuk ili tekst. Računalna grafika se može podijeliti na rastersku[17] i vektorsku grafiku ali i na dvodimenzionalnu i na trodimenzionalnu. Vektorska grafika jest prikazivanje slike pomoću geometrijskih likova koji su temeljeni na matematičkim jedinicama koje generira računalo. Upravo zbog takvog načina stvaranja slike njezina kvaliteta na računalu može biti visoka, te ju samim time računalo lakše čita to jest prikazuje/stvara. Jednostavno rečeno, ako pri velikim povećanjima razlučivost detalja na slici ostaje jednaka (ili je minimalno narušena) kao i u normalnoj veličini tada govorimo o očuvanju kvalitete.[18] Vektorske se slike vrlo često i lako prevode u rasterski format, dok je obrnut proces teško izvediv bez znatnije promjene i gubitka razlučivosti originala. Rasterska slika, odnosno bit mapa[19], pohranjena je u memoriju i sadrži podatke za svaki pojedinačni piksel neke slike. Dakle, njen prikaz ne ostvaruje se pomoću računalno generiranih geometrijskih oblika, temeljenih na matematičkim jedinicama. Pojam vektorska grafika je većinom korišten u kontekstu dvodimenzionalne računalne grafike. Softveri za stvaranje računalne grafike su različiti ovisno o tome kojoj vrsti

[17] Rasterska grafika ili bitmap je podatak koji predstavlja pravokutnu mrežu piksela ili obojenih točaka, na nekom grafičkom izlaznom uređaju kao što je monitor ili papiru. Svaka boja pojedinog piksela je posebno definirana tako da npr. RGB slike sadrže tri bajta po svakom pikselu, svaki bajt sadrži jednu posebno definiranu boju.

[18] Razlučivost ili rezolucija veličina kojom se definira mogućnost razdvajanja/razaznavanja sitnih detalja kojom se opisuje kakvoća slike. Kod klasičnog fotografskog filma razlučivost se određuje brojem linija koje se mogu razaznati na dužni milimetarskog filma. U digitalnoj tehnici razlučivost se mjeri ovisno o mediju. Za zaslone i digitalne fotografije odnosno u rasterskoj grafici razlučivost je broj piksela od kojih je sastavljena slika, po vertikali i horizontali, odnosno dpi (*dots per inch*). Za papirnate dokumente razlučivost je broj točkica od kojih je sastavljena slika po kvadratnom inču.

[19] Bit mapa “Bitmap” definira prostor prikaza i boju za svaki piksel ili “bit” u tom prostoru. GIF i JPEG formati su primjeri grafičkih slikovnih datoteka koji sadrže bit mape. One ne trebaju sadržavati bit s informacijom o različitim bojama za svaki piksel u svakom redu, već mora sadržavati informaciju koja ukazuje na novu boju dok ju medij za prikaz redom skenira. Jednobojnija slika će zahtijevati manju mapu, zato što se koristi rasterska metoda određivanja sliku, slika ne može biti uvećan a da ne izgubi na kvaliteti.

grafike su namijenjeni pa tako razlikujemo one koji služe za stvaranje 3D grafike i one koje služe za stvaranje 2D grafike. [37]

Gotovo svako 3D prikazivanje je izvršeno pomoću 3D ili 4D vektorske tehnike, pomoću točaka, linija i poligona. 3D grafika se sva bazira na vektorima, matricama i trigonometriji. Zbog toga je potrebno barem osnovno poznavanje istog (o čemu se detaljnije govori u nastavku). Iako se grafiku može podijeliti na dvodimenzionalnu i na trodimenzionalnu, 3D grafike koriste trodimenzionalnu reprezentaciju geometrijskih podataka pohranjenih u računalu isključivo radi izvođenja izračuna i renderiranja 2D slika. Često će se spominjati pojmovi poput **rendera** i **renderiranja**[20], pa ih je stoga važno i razjasniti jer renderiranje je velika i važna poddisciplina računalne grafike, a posebice kada se govori o 3D-u. Renderiranje u svijetu 3D-a predstavlja generiranje slike nastale isključivo u 3D softveru od 3D modela i ostalih 3D elemenata. Dokument s informacijama jedne 3D scene, koja sadrži objekte definirane točno zadanim jezikom ili točno zadanom podatkovnom strukturom, sadrži kao opise te virtualne scene informacije o geometriji, točki gledišta, teksturama, svijetlu i sjenama. Ti podaci se šalju dalje u program koji će izvršiti renderiranje, odnosno program koji će te podatke procesirati i prevesti u digitalnu sliku ili rasterski format te slike.

Riječ render se koristi i u video montaži kada se opisuje proces računalnog kalkuliranja specijalnih efekata u video produkciji prilikom finalnog eksportiranja videa, koje dakako sprovodi računalo. U 3D grafici render može biti *real time* ili *spor (*eng. *slow render)* kada se radi *predrender* (eng. *pre render*). *Real time* render je tipičan za video igre, dok je *pre render* tipičan prilikom produkcije animiranih filmova.

[20] Riječ render engleskog je porijekla i znači iskazati, predočiti, dati, prikazati, žbukati odnosno finalizirati. Renderirati, općenito rečeno, jest prikazati neki objekt definiranjem svojstava njegove površine i izvora svjetla.

2.1. Uvod u 3D

3D znači da osim *x* i *y* postoji i treći z pogled (eng. *view*) koji se još naziva i *depth of field* odnosno dubina polja. Dakle, osim samo horizontalnog *x* i vertikalnog *y* imamo sprijeda i straga pogled tj. *z* smjer gledanja na 3D objekt. 3D svijet se sastoji od poligona, čestica (eng. *particles*), mreža (eng. *meshes*) i *nurbsa*[21] i mnogih drugih elemenata. Svijet animiranog filma nije jedini format u kojem se koristi 3D; tu su još reklame, video igre te specijalni efekti u filmovima. Kada se govori o 3D-u odnosno 3D slikama i animiranim filmovima tada se isključivo misli na CGI slike. CGI je kratica za *Computer-Generated-Imagery*[22] što bi prevedeno na hrvatski jezik značilo *računalno-generirana slikovna izdanja*. Dakle, samo animacije koje su izrađene od CGI-ja mogu se svrstati u kategoriju 3D animacija. Budući da se granice između pojedinih vrsta animacije sve više brišu teško je tako nešto konačno definirati, međutim, u pravilu 3D slikom ili animacijom smatraju se samo one 3D grafike koji su nastale od CGI-ja.

3D računalne grafike se često spominju kao 3D modeli, a ne kao 3D grafike jer model je sadržan u grafičkoj podatkovnoj datoteci i tehnički nije grafička jedinica s obzirom da nije još renderiran, to jest vizualno prikazan. Zahvaljujući 3D printerima, danas 3D modeli nisu ograničeni samo na 3D prostor (na softvere za kreiranje i prikazivanje 3D modela). Proces kreiranja 3D računalnih grafika se može podijeliti na sljedeće osnovne faze: 3D modeliranje koje opisuje proces formiranja/modeliranja oblika i izgleda objekta, animiranje koje opisuje kretnje i položaje objekata unutar scene (ako je objekt animiran), definiranje rasvjete, materijala i tekstura, te 3D renderiranje koje stvara sliku objekta odnosno stvara 2D reprezentaciju 3D modela. Renderiranje je zadnja faza koja se odvija u 3D softveru (ili u render softveru) i od 3D kreatora zahtijeva minimalno ulaganje truda i vremena, iako to ne znači da je ujedno i najbrža faza, zato što u ovoj fazi

[21] Nejednolika racionalna osnovna krivina (NURBS) je matematički model koji se najčešće koristi u računalnoj grafici za stvaranje i prikazivanje krivulje i površine. Ona nudi veliku fleksibilnost i preciznost za rukovanje sa analitičke (površine određene zajedničkom matematičkom formulom) ali i modeliranim oblicima.
[22] U nastavku kao engleska kratica CGI

glavnu zadaću obavlja isključivo računalo i to ovisno o kompleksnosti 3D grafike i tehničkim mogućnostima kojima računao raspolaže. Zbog toga ova faza može biti jako dugotrajan proces, proces koji ponekad može trajati danima ili tjednima ovisno o veličini projekta. Stoga ovu zadnju fazu nikako ne treba zanemarivati kada se postavljaju vremenski okviri nekog 3D projekta. Renderiranje jest proces generiranja slike iz modela koji je napravljen u nekom od programa. Model jest opis trodimenzionalnog objekta u striktno definiranom jeziku ili *data* (podatkovnoj) strukturi. Sadržava geometriju, teksture i informacije o svjetlosti. Krajnja slika može biti digitalna slika ili rasterska grafička slika. Renderiranje čini upravo to - računalno kalkulira i kombinira sve parametre koji su prije definirani radi stvaranje plošne 2D slike. Ona se može dalje koristiti, po želji joj se mijenja format i "površinski" se modificirati (kao fotografija), ali u suštini takva je slika definitivna i nepromjenjiva. Render može biti *real time* (npr. u video igrama ili računalnim simulacijama) ili *non real time* (potrebno je mnogo više vremena, a renderi su kvalitetniji). Da bi se razumjela kompleksnost 3D prostora i objekata te na koji način se oni izrađuju, tj. modeliraju te kako ih računalo percipira mora se ponajprije upoznati s poligonalnim modeliranjem, a potom i ostalim načinima modeliranja. U sljedećim poglavljima raščlanjena je upravo ove tematika.

3. 3D animacija

Animator koristi računalo kako bi generirao sekvencu nepokretnih slika, koje daju iluziju kretanja kroz trodimenzionalni svijet kada se pregledaju u slijedu. Za takvo nešto potreban je 3D softver poput 3D Maxa, Maye ili Blendera koji su namijenjeni animaciji, te dobro računalo (iz razloga navedenih u prijašnjim poglavljima). Umjesto ručnog crtanja svakog detalja *frame* po *frame*, koristi se računalo za "crtanje" svakog pojedinog *framea*. Animator daje softveru naredbe o tome kako objekt treba izgledati i kako se treba kretati, te u kojim vremenskim okvirima. U praksi to bi otprilike ovako izgledalo - objekt A nalazi se u na poziciji *x1,y1,z1* u koordinatnom sustavu, dakle u prostoru, te u vremenu *f1* tj. *frameu* 1, te se pomiče na poziciju *x2,y2,z2* u roku od jedne sekunde što odgovora 25 *frameu*. Također, slične upute se daju i kameri tako da zna u što i kada treba gledati i iz koje perspektive. Pa tako i kamera ima svoje koordinate u 3D prostoru. Da bi se išta moglo vidjeti kroz kameru, to jest da scena ne bi bila potpunom mraku moramo postaviti i svjetla. Svjetlo, bilo da se radi o suncu ili rasvjetnim tijelima također, mora imati svoje parametre. U biti, parametri koji se odrede i postave u virtualnom 3D svijetu jesu pokušaji simulacije realnog svijeta u kojem, 3D umjetnik, određuje sve uvjete kretanja i postojanja. Tek nakon što su određeni i postavljeni svi potrebni parametri koji zadovoljavaju zacrtana očekivanja, računalo može na temelju tih brojki generirati sliku. To generiranje slike, kao što je već rečeno, tj. računalno kalkuliranje u animaciji, naziva se *renderiranje*. Ono što računalo u ovoj specifičnoj situaciji kalkulira (računa) jest putovanje koje će objekt A učiniti od točke 1 do točke 2 ili gdje točno će se objekt nalaziti (u trodimenzionalnom koordinatnom sustavu) u svakom *frame* –u od 1 do 25 (što u ovom slučaju odgovara 1 sekundi animacije). Istovremeno, računalo računa kako svaki *frame* izgleda kroz virtualni objektiv kamere koja je prije postavljena. Svakako je poželjno prije finalnog rendera napraviti nekoliko probnih rendera pri nižoj rezoluciji, kako bi se na vrijeme izbjegle pogreške ili neočekivane situacije. Tek kada su svi uključeni u proces stvaranja 3D animacije u potpunosti zadovoljni i gotovi, a napravljeni su i testni renderi i pred-

vizualizacije, sljedeći korak bio bi finalno renderiranje. Generiraju se visokokvalitetne sekvence slika koje će kasnije činiti dijelove animiranog filma. Vrlo često, kako bi se ubrzao proces renderiranja i omogućila veća manipulacija u *compositingu*, scena se razlaže na više elemenata. Neki od mogućih načina razlaganja scene - jesu na *background* i *foreground*, na statične i pokretne elemente ili se čak odvajaju vizualni efekti poput dima, kiše i sjena i drugo. Nakon odrađenog rendera vrši se *compositing* u kojem se, ukoliko je potrebno se spajaju razložene scene te se površinski modificiraju jer su to sada slike (često i rasterizirane). Iako neki 3D softveri nude mogućnost *compositinga* unutar softvera, najčešće se on obavlja u nekom vanjskom *compositing* programu. Na koncu, odabire se *output* i format te je gotova animacija spremna za lansiranje (eng. *launch*).

Važno je imati na umu da 3D softveri ne animiraju umjesto animatora objekte na sceni, već oni samo djeluju po naredbama koje je animator postavio. Dakle, animator jest taj koji animira, dok je računalo samo *medium* koji će dobivene naredbe i izvršiti. Baš kao što je i olovka medij kojim se stvaraju crteži na papiru. Kod niskobudžetnih i kratkih animacija računalo može znatno skratiti vrijeme potrebno za animiranje te smanjiti produkcijske troškove, međutim, ako se radi o dugometražnoj animaciji tada nam je svakako potreban tim ljudi. Samim time i troškovi rastu ukoliko želimo u nekom razumljivom vremenskom roku završiti projekt.

Računalo danas može učinit mnogo toga umjesto čovjeka:
Može izračunati **među-pokrete** (eng. *inbetween*-ove). Ako se objekta *O* kreće iz točke *A* u točku *B,* dovoljno je da se postavi u početnu i kranju točku, a kretnju između te dvije točke će izračunati računalo. Tako da nije potrebno "ručno" animirati baš svaki trenutak nekog pokreta. Time se znatno štedi na vremenu.
Zahvaljujući **čestičnoj animaciji** (eng. *particle animation*) lako se mogu iscenirati velike gomile ljudi, drveća ili bilo kakve mase ljudi, životinja, biljaka ili sličnog, a da nije potrebno svaki objekt zasebno izmodelirati i animirati.

Također računalo još može; stvoriti gotovo savršene **3D iluzije**, imitirati gotovo sve postojeće **materijale**, simulirati razne vrste kompliciranog **osvjetljenja**, **integrirati *live action*** s virtualnim slikama to jest kombinirati pokrete i izvedbe živog glumca s animiranim likovima (eng. *motion capture*).

Premda računalo znatno olakšava i ubrzava mnoge radnje animator je taj koji udahnjuje život animiranim stvarima jer u suprotnom pokreti i razne kretnje mogu ispasti mehaničke i neuvjerljive. Animator je taj koji točno mora postaviti parametre i *keyframeove* u animaciji da bi pokreti bili što uvjerljiviji i fluidniji. Sve što je slučajno i kaotično u prirodi računalu je teže uvjerljivo interpretirati. Kaos, odjeća i ostale organske teksture poput kože ili zemlje su iznimno "teško" izvedivi u CGI-ju jer moraju izgledati prilično uvjerljivo s obzirom da gledatelj (čovjek) očekuje određenu razinu realnosti zato što je 3D iluzija prilično uvjerljiva, pa iz tih razloga se i očekuje razumljiva razina realističnosti. Ljudsko oko/um je mnogo osjetljivije na pokrete i likove koji su humanizirani s obzirom da te kretnje čovjek vidi i čini svaki dan, te se s njima možemo i poistovjetiti, pa je tako i tolerancija za umjetno, odnosno mehaničko (kretanje), vrlo niska. Možemo zaključiti da razina vizualne reprezentacije, koja uvelike ovisi o autoru i cjelokupnoj ideji, ne mora nužno biti stopostotna dokle god način na koji je animirana odgovara do neke mjere (karikiranje jest ipak ono što animirane filmove čini posebnim) realnim očekivanim pokretima. Može se reći da tu leži ključ dobrog animiranja. Ujedno to je i dio kojem se posvećuje najviše pažnje i vremena. Usavršavanje kretnji poput *walk cyclea* (ciklusa hodanja) su jedne od najzahtjevnijih zadataka (vremenski i tehnički) koje jedan animator mora savladati. Sve u svemu računalna animacija za *full scale* projekte gotovo je jednako vremenski i produkcijski zahtjevna kao i klasično animiranje. Proces stvaranja jedne animacije od planiranja, modeliranja, animiranja pa do pozadinskih procesa koji se paralelno odvijaju da bi se neki lik softverski mogao animirati te rendera koji mogu trajati danima ili tjednima jest veoma kompliciran i dugotrajan. Kako bi se mogao razumjeti s tehničke strane način na koji nastaju animacije potrebno je opširnije znanje programskih jezika, programiranja i računalne grafike, no to nije nešto što animator treba izvrsno znati da bi napravio izvrsnu animaciju. Animiranje je danas timski rad koji je iznimno

kompleksan, dugotrajan i zahtjevan, te iz godine u godinu sve više i više inovativan. Kontinuirano se informirati o novitetima i *upgradeovima* postalo je ključno kako bi se pratili novi trendovi i zahtjevi tržišta, ali isto tako i poboljšali i modernizirali dosadašnji načini animacije, koja se još uvijek nije prestala razvijati. Animacija na koncu jest oblik umjetnosti i kao i svaka umjetnost zahtijeva puno vremena, strpljenja, talenta, strasti i prije svega mnogo vježbe.

Faze, u točno određenom redosljedu, koje se trebaju proći prilikom stvaranja jedne animacije, podijeljene su ovdje u 4 kategorije; planiranje, konstrukciju, animiranje te dotjerivanje i eksportiranje, jesu sljedeće:

Planiranje:

- Storyboarding, Background, Charcater design (Model pack)
- Animatik

Konstrukcija:

- Modeliranje
- Rigging
- Teksturiranje i osvjetljenje
- Previsualization

Animiranje:

- Animacija
- Pre-render

Dotjerivanje i eksportiranje:

- Compositing
- Specijalni efekti
- Sound design

- montaža
- **Export**

U sljedećim poglavljima pobliže se prošlo kroz većinu faza, kako bi shvatilo kako i što je potrebno učiniti za izradu jedne animacije od početka do kraja. Sljedeći gore navedene korake napravljena je jedna animacija u *open source* softveru, čije je *workflow*[23] pobliže opisan u poglavlju 7. poglavlju. Također, faza *animation script* tj. scenarij animacije nije navedena jer se smatralo da je već napisan tj. da već postoji. Neke od navedenih faza mogu biti kraće ili duže ili mogu biti preskočene, a neke čak i dodane (nisu sve navedene) no, sve to ovisi o veličini i vrsti projekta. Ovdje smo naveli samo one osnovnije i neizbježne za jednu uobičajenu produkciju.

3.1. Podjela, stilovi i vrste animacije

Animacija je područje grafike koje postoji već dugo vremena. U pravilu način na koji se radi nije se mnogo promijenio, samo su se razvili napredniji alati i razvile nove i raznovrsnije tehnike. U ovom poglavlju nije se ulazilo u to kako je animacija nastala i kakav je bio njen razvoj kroz povijest, već se samo ukratko dotaklo nekih od ostalih tehnika i vrsti animiranja. Fokus je, dakle, na 3D načinu animiranja. No, ipak se ukratko raščlanilo vrste i stilove animacije kako bi se bolje shvatilo gdje smjestiti kategoriju 3D animacije.

Važno je reći da se animacijom smatra bilo kakva manipulacija sekvencama slika "*frame by frame*". Sve animacije, uglavnom, možemo smjestiti u ove 3 kategorije; 2D, 3D ili *stop-motion*. Međutim granice između ove 3 kategorije sve se češće se slabije razabiru. Termin 2D animacije odnosi se na animacije stvorene korištenjem dvodimenzionalnih crteža. Klasično ili ručno nacrtane animacije su najjednostavniji primjer takve animacije. Iako se danas se 2D animacija radi isključivo pomoću računala, koja nam znatno pojednostavljuju

[23] Workflow, eng. Izvor: Oxford dictionary je sekvenca industrijskog, administrativnog ili drugog procesa kroz koje neko djelo prolazi od svog početka pa sve do kraja.

animiranje 2D objekata na sceni tj. "virtualnom papiru". 2D računalna animacija se još naziva i vektorskom i *renderira* se u dvodimenzionalnom prostoru, dok 3D animacija se renderira u virtualnom 3D prostoru, pomoću poligona, virtualnih kamera i tako dalje. Kako i 2D, tako i 3D animiranje se postiže *frame by frame* tehnikom ili matematičkom interpolacijom između dva *keyframea*. I to se ne razlikuje ni kod svih ostalih vrsta i stilova animacija. Međutim, prvotne faze, prije samog animiranja, su veoma različite. Iako oba načina mogu u nekoj mjeri imitirati *stil* jedan drugoga, softveri za stvaranje 2D i 3D animacije su potpuno različiti. Pomoću 3D softvera mogu se izraditi animacije koje izgledaju kao da su napravljene 2D softverom, te do neke mjere 2D softveri mogu učiniti isto, iako znatno lošije i teže. No, to i dalje ne mijenja činjenicu da je ta animaciju u suštini 2D odnosno 3D. Neki od softvera koji se koriste za 2D animaciju su; *Adobe Flash i Director/Shockwave*. Neki od 3D softvera su; *3D studio max, Maya, Cinema4D, Blender* itd.

Može se reći da kada se nekoj animaciji pripisuje prefiks 3D tada to znači da 3D slike od kojih je animacija ukomponirana su u stvari CGI slike što bi značilo da su računalno generirane slike koje stvaraju iluziju 3D prostora s velikom preciznošću. Specijalni efekti u filmskoj industriji, također, spadaju u tu kategoriju. No, u konačnici renderiziranjem svaka 3D animacija, iako stvorena od 3D modela animiranih u 3D virtualnom svijetu, postane sekvenca plošnih, dvodimenzionalnih slika projiciranih na nekom ekranu. Treba naglasiti da stereoskopija je kombinacija dviju neznatno različitih slika koje stvaraju iluziju prostora, baš kao i ljudske oči u stvarnom svijetu, nije vrsta 3D animacije već način projekcije, npr. *flipbook*. Sve što je snimano uživo slika po sliku tj. *frame by frame* ispred fotografskog objektiva naziva se *stop-motion* ili *stop- frame* animacijom. Takve animacije mogu izgledati vrlo trodimenzionalno, na isti način kao što neki video izgleda trodimenzionalno, ali termin 3D, kao što je već rečeno, je isključivo rezerviran za računalnu CGI animaciju. Primjerice, *pixilation*[24] je oblik *stop-motion* animacije u kojoj se animiraju živi glumci umjesto npr. glinenih figurica.[37]

[24] Naziv pixilation dolazi iz engleskog jezika od rijeci *pixies* (mitsko biće, patuljci, vilenjaci) a ne od rijeci *pixel*. Zbog neobičnog načina na koji se likovi pomiču i kreću u animaciji.

Danas se rijetko čuje da se neki animirani film naziva crtanim filmom (eng. *cartoon*) ili obratno. Čini se da je crtani film postao termini koji se odnosi na starije crtane filmove koji su se radili ručno dok još nisu postojali specijalizirani računalni softveri za crtanje i animiranje. Iako crtani (eng. *cartoon*) filmovi jesu ujedno i animirani filmovi, ako se vodi osnovnom definicijom animacije, ipak taj termin se sve rjeđe upotrebljava kad se govori o crtanim filmovima. Može se reći da mnoge vrste animacije (određeni načini kreiranja animacije ponekad stvaraju i svoje nove stilove koji postaju specifični samo za tu vrstu animacije) dobe ime po načinu na koji su izrađene, svrsi ili čak po softveru u kojem su izrađene. Neki od ostali termina i imena koja se koriste za različite vrste odnosno ponekad i stilove animacije su; *Web* animacija (*GIF*-ovi ili *Flash*), *Jab* animacija, *animatik* (koristi se za animiranje *storyboard*-a), *VJ* animacija, *Whiteboard* animacija, *live-action*, 2D *motion-graphics* animacija (hrv. 2D pokretna grafika), *screencast*, *typography*, eksperimentalna animacija i tako dalje. Vrste animacije mogu se probati podijeliti i na sljedeći način, po realističnosti, pa se razlikuje ovih par stilova; **polu-realistična** animacija, gdje se stvari ne žele prikazati skroz realističnima ali ih se želi prikazati vjerodostojnim i uvjerljivim, **realistična** animacija, gdje su pokreti veoma realistični jer se stvaraju pomoću *motion capture* tehnike[25], **hiper-realistična** animacija, gdje animacija nadilazi realizam, te se likovi pomiču na načine koji nisu mogući u stvarnom životu, emocija koju likovi predstavljaju, također, će biti drugačija od stvarnosti, i *charcater design* će biti jedinstven, **crtana** ili ''žustra'' animacija, koja nam daje osjećaj klasične animacije (pa čak i ako se radi o 3D likovima), gdje se ne pokušava prikriti stiliziranost likova, likovi su animirani ovisno o karakteru koji predstavljaju, **stilizirana** animacija, koja pretjerivanjem kod nekih realističnih scena daje faktoru zabavnosti na vrijednosti, i to vrlo često pomoću posebnih tehnika koje omogućuju alati za animiranje, te **limitirana animacija**, u kojoj se sa što manje slika pokušava dočarati pokret odnosno iluzija kretanja. [5]

[25] Gdje se snimaju pokreti stvarnih glumaca, koji se kasnije obrađuju i koriste kao referenca za animiranje likova.

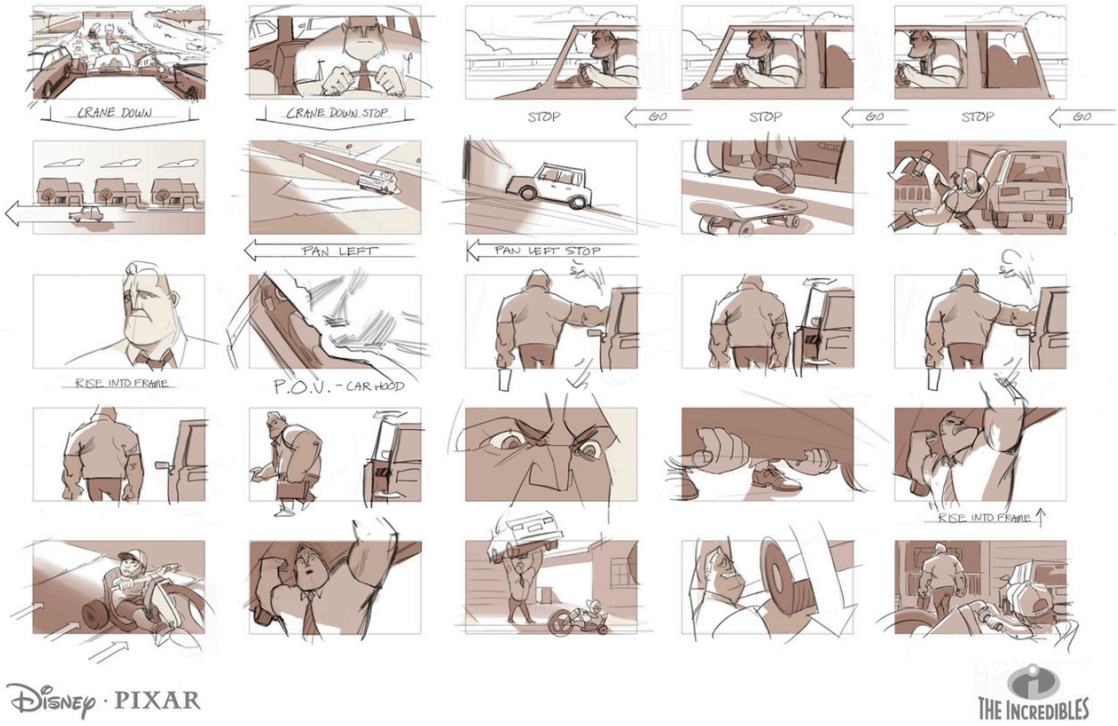
3.2. *Pre-animation*; planiranje animacije

Prije nego se započne s kreiranjem 3D animacije u bilo kakvom softveru važno je zamisliti i napisati priču (ako još ne postoji) te ju dobro razraditi i uskladiti s tehničkim mogućnostima softvera. Što je bolje definirana na početku procesa to će čitavi *workflow* teći uz što manje pogrešaka i izmjena. Tako će i sama faza pripreme za animaciju biti znatno kraća s obzirom da je upravo taj dio jedan od najzahtjevnijih što tehnički pa tako i vremenski pa je stoga iznimno bitno da se tu radi što manje pogrešaka i izmjena odnosno da se prije početka samog procesa što jasnije definiraju sadržaji, objekti i pokreti buduće animacije. Pa stoga, nakon što je ideja razrađena , priča osmišljena te scenarij po mogućnosti napisan, napraviti će se *storyboard*, i to što tehnički preciznije. U ovom dijelu nije toliko bitno koja se tehnika koristi te koliko su crteži foto-realistični ili vjerodostojni likovima iz priče, već koliko jasno prikazuju pozicije kamere, objekata te pokrete koji će se kasnije animirati. Iznimno je važno obratiti pažnju na promjene kadra, kako ne bi bilo oštrih rezova te kako bi cjelokupna animacija bila što fluidnija ili oku ugodnija i uvjerljivija, i do neke mjere realistična. 3D animacija pruža upravo tu mogućnost - da se iz realističnog i polu-realističnog odskoči u super-realizam te kreiramo svijet po vlastitoj ideji. Iz tih razloga za promjenu kadra ne moraju se nužno koristiti uobičajeni načini prijelaza iz scene u scenu koji se inače koriste u klasičnom filmu. Upravo u ovakvim detaljima krije se plastičnost CGI animacije koju nam omogućava svijet 3D-a.

3.2.1. *Storyboarding* i animatik

Storyboard jest panel ili paneli (ploče, eng. *board*) na kojim su prikazane sekvence crteža/skice koje prikazuju i opisuju značajne promjene akcije i scene unutar planiranog filma, televizijskog *show-a*, animacije ili televizijske reklame itd. Pa tako, za potrebe *storyboarding*-a, mogu se koristi razni softveri (digitalni mediji) koji olakšavaju organizaciju rada i nude alate koji omogućavaju manipulaciju objektima (crtežima) na način da se mogu ponovno upotrebljavati i modificirati ovisno o potrebama na sceni. Pa tako neki od najpoznatijih softvera su: *Toon Boom, Celtx* (koji ima i svoju besplatnu verziju), stranica *www.storyboardthat.com* i tako dalje, ili jednostavno, ovisno o kompleksnosti animacije i budžetu, *storyboard* se može napraviti prostoručno s jednostavnim alatima, poput olovke. Jedino što jest bitno u cijelom procesu jest da je *storyboard* jasan/čitljiv, precizan i fluidan. Dok sama tehnička i umjetnička dotjeranost crteža, opširno detaljiziranje te koloritet u ovoj su fazi manje bitni. Dakako, ako se želi istaknuti određeni objekt na sceni tada je poželjno da je i u *storyboard*-u detaljnije razrađen kako bi mu se i u kasnijim fazama posvetila jednaka pažnja. Ukoliko iste osobe neće kasnije izrađivati i animirati te objekte i likove jako je bitno da sljedeća osoba u procesu ima što jasniju sliku kako taj objekt ili lik te cjelokupna scena trebaju izgledati. Važno je, također, naglasiti da u nekoj mjeri *storyboard*-om određujemo dužinu animacije odnosno njeno trajanje. Osim obveznog numeriranja scena i označavanja prelazaka iz kadrova, poželjno je napisati i očekivano trajanje određenih scena. Kod *storyboard*-a za stripove ovako nešto nije nužno. Ukoliko u animaciji ima dijaloga ili monologa, ispod svakog panela koji predstavlja scenu može se napisati tekst odnosno dijalog koji će se odviti u sceni. Svako dodatno opisivanje scene i zvukova, iako moguće, nije poželjno stavljati u *storyboard*. *Storyboard* bi trebao moći samo slikom odnosno vizualno što bolje dočarati ugođaj scene, pa tako i cijelog filma. Na slici 1 prikazan je dio jednog uspješno izvedenog *storyboard*-a. Isto tako, ako je projekt iznimno velik i zahtjevan često se koristi više od jednog *storyboardista* koji po uputama režisera te po scenariju rade *storyboard*. Zašto se toliko pažnje posvećuje upravo ovom dijelu u procesu stvaranje jednog

filmskog ili animiranog uratka? Zato što upravo ovdje po prvi puta napisani tekst ili zamišljena ideja poprimi vizualne tj. grafičke oblike, odnosno riječi po prvi puta postaju slike, što znači da *storyboardisti* na neki način imaju moć ili slobodu definiranja budućih likova, detalja i scenografije, pa čak i ako se rade po izravnim naputcima autora. Iz tih razloga, osobe koje se bave kreiranjem *storyboard*-a imaju iznimno zahtjevnu i važnu ulogu u procesu stvaranja čitavog projekta. Stoga, u animaciji *sotryboarding* s pravom jest umjetnička grana za sebe.



Slika 1: *primjer storyboard-a za velike produkcije*
(izvor: http://41.media.tumblr.com/tumblr_m8lbg5dbe81r31ck3o1_1280.jpg)

Storyboarding se može podijeliti na sljedeće faze odnosno dijelove koje je potrebno izraditi i/ili definirati:

1. Utvrđivanje vremenskog toka; utvrđivanja parametra kada i gdje je priča smještena, određivanja redoslijeda događaja u priči, kronološki/linearno ili će biti *flashback*-ova, putovanja kroz vrijeme, (prošlost, budućnost itd.) napraviti listu kojim će se

redom, glavni događaji, pojavljivati na ekranu odnosno kojim će redom biti ispričani.

2. Identificiranje ključnih scena u priči; *storyboard* mora gledaocu predočiti suštinu priče koja će biti kasnije animirana/snimljena. Nije nužno prikazati baš svaki najsitniji detalj već ukazati na ključnije momente. Važno je prikazati prekretnice u priči tj. *plot-twist*-ove, važnije promijene te promijene scenografije. Ako animacija traje 30 sekundi tada *storyboard* ne bi trebao imati više od 15 *framea*. U prosjeku 2 sekunde po *frameu*.

3. Koliko detaljno ići; *storyboard* može biti vrlo jednostavan ali i iznimno detaljan. Međutim ako se radi o podužoj animaciji ili filmu tada se treba odlučiti koliko detaljno se želi ići, u suprotnome može postati prezahtjevno i neisplativo. Stoga nije nužno ulaziti u svaki detalj a po potrebi neke se scene mogu prikazati mnogo detaljnije od ostalih. Cilj *storyboard*-a je pružiti vizualnu jasnoću kako bi baš svima bilo jasno o čemu se tu radi. Storyboard nije umjetničko djelo već praktičan pregled cjelokupne slike i priče a ne skup individualnih prikaza. Tom se idejom treba voditi kada se odlučuje o razni detalja na sceni. Dakle, dobar *storyboard* treba biti razumljiv svima koji ga gledaju, jer on je vodič, referenca i smjer za animiranje i modeliranje.
4. Sastaviti opis što svaka ćelija treba prikazivati. Ova faza posvećena je podrobnijem analiziranju i promišljanju o svakoj pojedinoj ćeliji, o onome što će se u njoj nalaziti. Zato je dobro napraviti kratak popis bitnijih elemenata. To pomože pri određivanju onoga što uistinu treba u njih nacrtati. Ako se radi o dijalogu važno je opisati emociju likova na sceni, kao i pozadinu scene ako se događa promjena pejzaža ili atmosfere (npr. priroda, grad, noć, dan).

5. Odrediti koji medij koristiti za predložak (*šablonu*). Običan *storyboard template* (hrv.~predložak) može se nacrtati prostoručno podjelom papira na prazne okvire jednake veličine ili se mogu koristiti softveri. Ćelije sa budućim scenama mogu biti postavljene vertikalno ili horizontalno, ali redoslijedom kojim će biti prikazane na ekranu. Veličina ćelija bi trebala biti u istog omjera kao i završni video. Također, poželjno je imati red ili kolonu za *audio* u koju se mogu staviti smjernice za glazbu, dijaloge i ostale zvukove, te ako ima teksta u sceni treba ostaviti i prostor za njega.

6. Skiciranje sličica – Ovo je grubo skiciranje onoga što je već određeno prijašnjim fazama. Poželjno je što više ispravljati i precrtavati, zato što ovdje se moraju postaviti kutovi iz kojih kamera snima, odnosno vrstu kadra (široki, krupni, preko ramena kadar ili pak kadar koji prati akciju), kompozicija (svijetlo, *foreground, background*, palete boja), rekviziti, protagonisti i specijalni efekti.

7. Popratne informacije – numeriranje ćelija, kratke indikacije (ako su potrebne) dijalozi i duljine trajanja scene.

8. Finaliziranje – Nakon što su identificirane ključne točke subjekata i dizajniran je svaki frame, važno je pregledati cijelo djelo i napraviti zadnje promijene ako su potrebne. Svaka ćelija mora prikazivati točno onu akciju koja je za nju zamišljena. Po potrebi opisi i dijalozi mogu biti prilagođeni. Poželjno je da netko *vanjski*, također pregleda *storyboard* kako bi se dao *feedback* i tako se osiguralo da je uistinu sve fluidno i jasno. Ako će to pomoći u prenošenju priče može se dodati i boja.

9. Fino-uštimavanje (eng. *fine-tuning*) nije naj-neophodniji dio, no ponekad može pomoći da bi se bolje shvatila cjelokupna slika, a uključuje: stavljanje likova u perspektive, približavanje ili

udaljavanje likova i objekata od kamere (određivanje dubinske oštrine).

10. Definiranje rezova – svaki prijelaz iz jedne scene u drugu je definiran nekom vrstom prijelaza. Vrlo je značajno kakav će taj prijelaz biti. U ovom trenutku treba razmišljati i o tome kako će te scene zaživjeti u animaciji ili filmu, zašto napraviti rez, na koji način i u koje vrijeme. Ovakvi detalji određuju tempo i ritam cijelog dijela, kao i održavanje kontinuiteta i zadržavanje pažnje gledaoca.

11. Organsko preoblikovanje – Iako je *storyboard* iznimno moćan alat koji pomaže kada se konačno snimaju ili stvaraju scene, vrlo često se dogodi da u toku, dođe do nekih promjena ili prilagodbi iz ovih ili onih razloga. Ponekad, jednostavno u toku rada se uvidi da se neka scena može bolje ili drugačije izvesti ili se pak otkrije potpuno nova scenu koja bi se iznimno dobro uklopila, također moguće je i to da softverska ograničenja onemoguće realiziranje određenih scena. Stoga je važno ostaviti prostora za mogućnost promjene u *storyboard*-u koja bi se mogla dogoditi sama od sebe.

Nakon što je *storyboard* zgotovljen od njega se može napraviti animatik. **Animatik** je animirani *storyboard*. U nekom programu za *editiranje* paneli se ujedine i poslože tako da ima trajanje i tempo koje odgovara onome u filmu ili animaciji za koju su napravljeni. Oni uključuju osnovne zvučne efekte i snimke dijaloga. Poput *storyboard*-a animatik služi za pred-vizualizaciju filma prije nego produkcija započne. Iznimno su bitni za stvaranje animiranih filmova zato što omogućuju da se po prvi puta vidi kako bi budući film ustvari mogao izgledati. Ovdje se po prvi puta dobiva osjećaj tempa i ritma te generalnog napredovanja (animiranog) filma. Nakon animatika poželjno je napraviti posljednje ispravke i biti sigurni da više nama izmjena u priči. Za stvaranje animatika obično se koristi *Adobe After Effects* zato što omogućava jednostavnu manipulaciju crtežima, kamerom, *time-code*-om i eksport formatima.

3.2.2. *Charcter design (Model Pack)*

Osim *storyboard*-a, prije negoli se počne s 3D modeliranjem i animiranjem, treba napraviti *character design* tj. razradu i dizajn likova (glavnih protagonista), kreirati i dizajnirati *props*-e (rekvizite i druge ključne objekte na sceni) te razraditi *background* tj. pozadinske scene. Tek nakon što se to učini može se prijeći na sljedeću fazu.

Kreiranje likova ili *Character designing*, iako se čini jednostavnim jer su likovi iz crtanih filmova uglavnom vrlo jednostavni, upravo je tu jednostavnost jako teško postići. Stoga, tajna uspješnog kreiranja likova jest upravo u postizanju te jednostavnosti. Kada se stvara neki crtani ali i 3D lik pokušavaju se koristiti što jednostavnije linije i oblici. Cilj je stvaranje lako prepoznatljivih karakteristika, te razvijanje osobnosti lika. Bitno je znati što se želi više istaknuti a što prikriti. To uvelike ovisi i o tome za koga je namijenjena ova animacija (dob ili drugi kriteriji). Dakle, naša ciljana publika (eng. *target market*) određuje način na koji razvijamo naše protagoniste. *Character design* kojem je ciljana publika mala djeca, uglavnom se bazira oko jednostavnih oblika i veoma živih boja. Ovisno na kojoj će se medijskoj platformi prikazivati određuje se i razina detalja na samom liku. Nije pogrešno proučavati druge poznate likove jer važno je shvatiti što ih čini uspješnima, te koja je to karakteristika koja najviše privlači. Postoji jako mnogo sličnih likova i bitno je pronaći nešto što će lika učiniti posebnim i prepoznatljivim, koji, naravno, mora biti vizualno privlačan da bi se pridobila pažnja gledatelja. Pa stoga, često i odskakanje od normi i uobičajenog može pomoći u pridobivanju te pažnje. Obično već u prvih nekoliko sekundi promatranja promatraoc presudi zanima li ga ono što gleda ili ne. Stil oko kojeg se gradi naš lik, također određuje njegov karakter; masne, blage, okrugle i jednolike linije mogu značiti da je lik pristupačan, sladak i dobar, dok oštre, iscrtkane i nejednolike linije mogu značiti da je lik nelagodan i nestabilan itd. Kod crtanih filmova uglavnom se ne teži foto-realističnosti već se pokušava pretjerati u nekim karakteristikama (npr. prevelike oči, glava, duge noge) Pretjerane značajke, osim što će naglasiti karakterne osobine glavnog protagonista, pomoći će gledateljima da brže i jasnije identificiraju njegove

ključne osobine. Izbor boja je isto tako vrlo bitan, logično tamnije boje predstavljaju negativne ličnosti, dok svijetle i nježne boje pozitivne ličnosti. Jake boje poput žute, plave i crvene mogu pridonijeti važnosti lika u priči. Predmeti (*props*) poput odjeće, obuće, kišobrana, torbe, koji lik nosi ili ima sa sobom postaju dijelom njegova *character* dizajna odnosno definiranju njegov karakter. Sam, dosad razvijen, izgled lika ne mora nužno biti dovoljan da bi se predstavila njegova ličnost. Stoga ga je bitno prikazati u nekoliko poza i situacija. Pogotovo u situacijama gdje su izražene emocije (tuga, sreća, strah, bol) i reakcije na situacije. Poželjno ga je nacrtati u svim pozama *turn-around*-a[26] (najosnovniji *turn-around* jest; sprijeda, tri-četvrt, profil i leđa). Također, ne treba zanemarivati ni ekspresije lica, pa ako je fokus često na licu našeg lika potrebno je nacrtati i više izraza lica koja su moguća u animaciji. Kako i za *storyboard* tako i za *charcter design* vrijedi isto pravilo, mogu se koristiti razni programi, od komercijalnih (*Adobe Photoshop*, *Correl painter*, *Adobe Illustrator* itd) do *open source* softvera (*Gimp, Inkscape, SWG Edit* itd.) ili se jednostavno mogu koristiti osnovni alati poput olovke i papira. Radovi napravljeni na klasičan način kasnije se mogu digitalizirati skeniranjem. Nije na odmet zatražiti *feedback*[27] od publike tijekom procesa kreiranja dok još lik nije sasvim gotov, kada je preinake mnogo lakše sprovesti nego što bi to bilo kasnije u procesu modeliranja. Planiranje i razmišljanje unaprijed o tome na koji način će se lik ponašati i u kakvom će se okruženju nalaziti odnosno njegovo smještanje u određeni kontekst znatno može utjecati na njegov finalni *character design*. Pa tako, u ovako ranoj fazi, razmišljanjem o **scenografiji** i okolini u kojoj će lik obitavati se definira do neke mjere i atmosfera buduće animacije (npr. tmurna, vedra, brza, spora, smiješna), ali i potvrđuje se njegova vjerodostojnost. Ukoliko se njegova pojava ne uklapa s danim ambijentom, vjerodostojnost njegova karaktera biti će svakako narušena. Može se reći da kod stvaranja *charcter design*-a je poželjno što preciznije razraditi svaki detalj, do te mjere dok on u potpunosti ne zadovoljava sve postavljene kriterije, kao i one osobne, pa tako i one svrsishodne. Ne treba ni zanemariti facijalne ekspresije, jer izražavanje emocije jest element koji

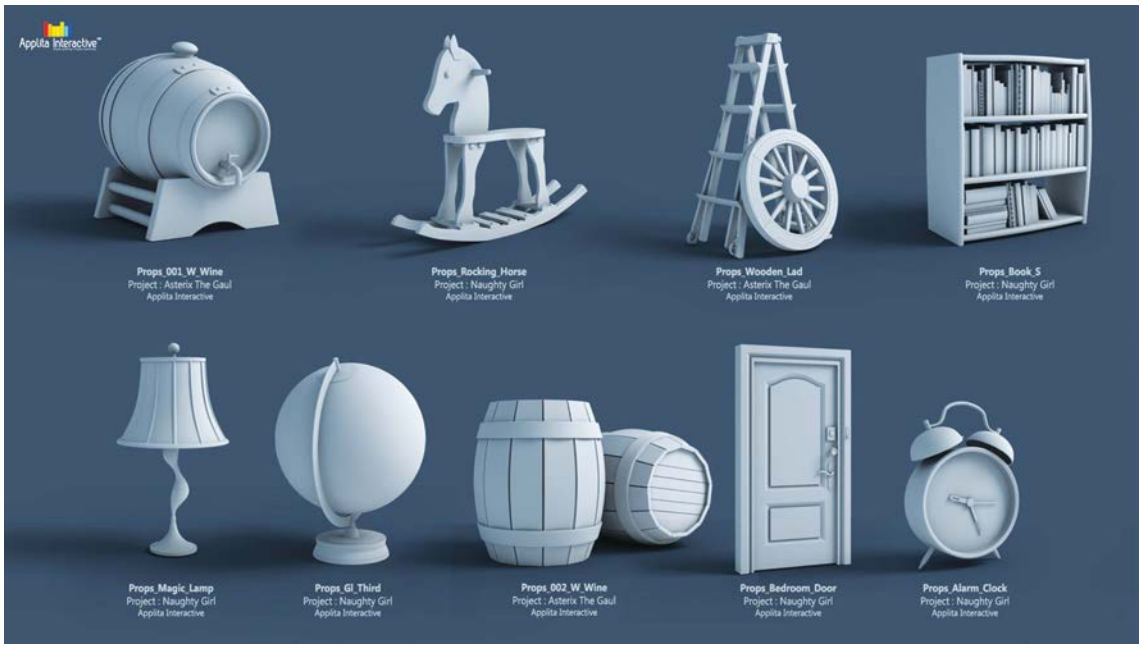
[26] Engleska sintagma za okret od 360 stupnjeva. Okrenuti se.
[27] Eng. Izvor Oxford dictionary; je povratna informacija o reakciji na proizvod, nečije izvršavanje zadatka itd. koji se koristi kao osnova za poboljšanje

najviše utječe na to hoće li nešto biti percipirano kao živo biće ili ne.[15] *Character design,* očito nije nešto što bi se pri kreiranju animacije trebalo napraviti usput ili brzinski već mu treba posvetiti maksimalna pažnja jer upravo ovdje određujemo estetski i karakterni smjer čitave animacije.

3.2.3. *Props* i *Background*

Osim razrade likova važno je i definirati ostale elemente na sceni, koji se u svijetu animacije nazivaju *props* (hrv. rekviziti). Pod rekvizite[28] spadaju svi oni predmeti koji ne čine dio scenografije, no esencijalni su da bi se mogla izvesti neka scena. *Props* zajedno s *character design*-om čine ***Model pack***. ***Model pack*** je srž vizualnoga koncepta u produkciji jedne animacije. To je dokument koji sadrži glavne vodilje vizualnog identiteta koje treba pratiti tijekom cijele produkcije. Ovdje se po prvi puta detaljno definiraju stil i izgled (vizual) animacije, dok se u *storyboard*-u detaljno definiraju priča i postavke kadra. Poželjno je *props*-e razvijati paralelno sa likovima i ostalim objektima scene (scenografijom) tako da se međusobno stilski uklapaju, osim ako animacija ne zahtjeva drugačije ili ako se ne radi o više različitih stilova koji se međusobno kombiniraju. Dakle, objekti poput šešira, automobila, bicikle, naočala, koji ne čine dio scenografije, ali će u nekom trenutku biti korišteni na sceni smatraju se *props*-ima to jest rekvizitima te se moraju dizajnirati. Na slici 2 može se vidjeti nekoliko primjera dizajna rekvizita.

[28] rekviziti (latinski requisita) sve potrepštine koje su, pored dekoracija i garderobe, potrebne da bi se moglo prikazati neko kazališno djelo. Zbog osobitog jezika koji se koristi u svijetu 3D animacije neću uvijek prevađati sve riječi na hrvatski, jer se u praksi vrlo slabo ili uopće ne koriste. Iako riječ *rekvizit* jest dobar prijevod, u ovom kontekstu engleska riječ bolje odgovara radi boljeg sveukupnog shvaćanje teksta.



Slika 2*: Dizajn rekvizita napravljen u 3D-u (izvor: http://studio.envato.com/explore/3d-design-modeling/8469-3d-animation)*

Što se tiče *background*-ova, ilustrirane pozadine su više karakteristične za 2D animacije, dok u 3D animacijama se uglavnom radi o 3D modelima koji ovisno o sceni se mogu renderirati zasebno te upotrebljavati kao pozadinska slika kako bi cijela scena bila “lakša” za konačno renderiranje. Naime što ima manje 3D objekata na sceni, to će renderiranje scene biti brže zato što računalo treba manje vremena da napravi izračune. Zbog toga se često scena renderira u “slojevima”. Slojeve možemo podijeliti na: *foreground*, *background*, likovi zasebno, prirodne pojave (na primjer, magla, dim), pa čak i sjene, i neke refleksije svijetla. Kombinacije mogu biti različite ovisno o projektu. Dok u 2D animaciji kreiranje pozadinske scene može biti zahtjevan posao te gotovo u potpunosti je prepušten ilustratorima i njihovim vještinama, u 3D-u se radi o 3D modelima. Poželjno je prije postaviti kameru i kutove snimanja kako bi se izbjeglo nepotrebno modeliranje dijelova objekata koji se neće vidjeti na sceni. Ovisno o stilu i kompleksnosti animacije, ovisi i vrijeme kreiranja scenografije.

3.3. Konstrukcija; Modeliranje, *Rigging*, Teksturiranje i osvjetljenje, *previsualization*

Kategorija koju je nazvana konstrukcija se sastoji od 3 osnovna koraka a to su: modeliranje 3D objekata, njihovo *rigg*-anje te lansiranje testnih rendera. U ovu fazu ulazi i namještanje svijetla i kamera, te teksturiranje i/ili koloriranje 3D modela. Ovo je ujedno i prva faza koja se radi u 3D softveru. Upravo zbog produkcijske kompleksnosti 3D-a važno je da ono što se napravi u ovoj fazi prođe kroz što manje izmjena i bude izvedeno na što jednostavniji, tehnički "čišći" način, kako bi se optimiziralo vrijeme produkcije čitave animacije, ali i ostavilo najviše prostora za animiranje koje je ujedno najduže i znalački najzahtjevnije. Iza dobrih modela kao i iza dobrog animiranja stoji mnogo iskustva i vježbe. S više prakse 3D modeli postaju bolji a animacije uvjerljivije. U nastavku smo prošli kroz ove tri faze i njihove popratni radnje.

3.3.1. Modeliranje

3D modeliranje je proces kreiranja matematičke reprezentacije trodimenzionalne površine nekog objekta korištenjem specijalnih 3D softvera. Produkt tog procesa jest 3D model. Takav model na koncu, za potrebe animiranog filma, je prikazan kao dvodimenzionalna slika, koja je stvorena procesom renderiranja. Modeliranje je uglavnom manualno ili polu-manualno. Postoje već gotovi izmodelirani objekti koji se mogu koristiti ili adaptirati, no može se krenuti modelirati i iz "ničega". Modeliranje u teoriji može se paragonirati s kiparstvom. Od nekakvog jednostavnog geometrijskog oblika počinjemo izdvajati brjegove ili dubiti dolove sve u svrhu oblikovanja željenog oblika. Razlika između stvarnog modeliranja i 3D modeliranja jest ta što u 3D-u se barata sa geometrijskim površinama, obično trokutima ili pravokutnicima. 3D model se sastoji od površina tj. ploha, vrhova tj. točaka i rubova tj. linija, koji su međusobno povezani u virtualnom 3D prostoru. S obzirom da su 3D modeli

ustvari kolekcija podataka odnosno točaka i drugih informacija, oni se stvaraju algoritamski ili skeniranjem. U slučaju animacije to je gotovo uvijek algoritamski (što izvodi računalo). Ovo je faza gdje će se stvoriti 3D likovi, rekvizite i čitava okolinu koja će činiti animaciju. Ti 3D modeli rade se na bazi *character design*-a koji je prije ustanovljen. Postoje više vrsta 3D modeliranja, ali u animaciji prevladava poligonalno i NURBS (eng. *Non-Uniformal Rational Bezier Splines*) modeliranje. Modeli mogu biti čvrsti (eng. *solids*) ili "*rubni*" (eng. *shell* ili *boundary*). Čvrsti modeli definiraju volumen objekta kojeg predstavljaju dok *boundary* predstavljaju samo površinu odnosno "*ljusku*". Prvi se više koriste u inženjerske svrhe dok drugi se više prakticiraju u računalnoj grafici. S obzirom da površine nisu konačne, potrebna je diskretna digitalna aproksimacija. Najrašireniji način reprezentiranja konačnih površina je pomoću poligonalnog modeliranja, te po potrebi još se može primijeniti i *subdivizija površine* (eng. *subdivision surface*), ako se želi postići još glađe i definiranije površine. Kod poligonalnog modeliranja apstraktni objekti su svedeni na jednostavne oblike poput trokuta i pravokutnika, koji se u tom slučaju nazivaju *mreže*, (eng. *meshes*). *Meshes* su mreže međusobno povezanih trokuta ili pravokutnika koji u konačnici sačinjavaju objekt. Mreže trokuta, za razliku od pravokutnih, pokazale su se jednostavnijima za renderiranje zbog tehnike renderiranja koja se u međuvremenu razvila (tehnika eng. *scanline rendering*)[29]. Taj proces transformacije reprezentacije objekta zove se "*popločavanje*" odnosno engleski *tessellation*. *Tessellation*, odnosno popločavanje odnosno *"mozaik-slaganje"* je postupak popločavanja površine korištenjem jednog ili više geometrijskih oblika, bez preklapanja ili praznina. Termin se upotrebljava i u matematici i u arhitekturi ali u računalnoj grafici se taj termin koristi kada se opisuje upravljanje podacima o poligonima koji predstavljaju objekte na sceni kako bi ih podijelili u odgovarajuće strukture za proces renderiranja. Npr. u *real time* renderiranju, podaci su posloženi ("popločeni") u trokute (*Open GL i Direx X*)[30]. Glavna prednost slaganja za *real time* grafiku jest to da dopušta da se dodaju i

[29] *Scanline rendering* je algoritam za određivanje vidljivih površina u 3D grafici. Radi na principu red po red, a ne poligon po poligon ili piksel po piksel principu.

[30] OpenGl i DirectX su multiplatformski aplikacijski programski *interface*-i za render 2D i/ili 3D vektorske grafike.

oduzimaju detalji iz 3D mreže bazirajući se na postavljenim kontrolnim parametrima. Na ovaj način originalni podaci, odnosno *mreže*, sa svakom promjenom ne gube dodatno na kvaliteti.[34]

Dakle, može se reći da su 3 najpopularnija načina za reprezentaciju modela **poligonalno** modeliranje, **krivuljno** modeliranje i **digitalno** kiparenje. **Poligonalno modeliranje** jest kada je model sastavljen od poligonske mreže (eng. *polygon mesh*). Točke u 3D prostoru su povezane linijama kako bi formirale mrežu. Takve mreže su fleksibilne i lakše su za renderiranje. Međutim, s obzirom da su poligoni planarni, zakrivljene površine mogu se jedino prikazati s još većim brojem poligona.

Krivuljno modeliranje jest kada su površine definirane krivuljama koje su pod utjecajem ponderiranih[31] kontrolnih točaka. Krivulja slijedi točke. Ako se težina povećava za neku točku (ponderacija) tada će se krivina povući bliže toj točki. Tipovi krivina su; NURBS (*non-uniform rational B-spline*), krivine (*spline)*, zakrpe (eng. *patches*) i geometrijske primitive (eng. *geometric* primitives).

Digitalno kiparstvo (3) (eng. *digital sculpting*) je relativno nova metoda modeliranja. Postoje 3 vrste: *premještanje* (*displacement*) **(1)**, koja koristi gusti model koji je uglavnom nastao subdivizijom površine (*subdivision surface*-om) poligonske kontrolne mreže (*mesh*-a), te sprema nove lokacije za verteks[32] pozicije korištenjem 32bitne mape slike koja sprema popravljene lokacije; *Volumetrijske krivine* **(2)**, koje su bazirane okvirno oko voksela, funkcioniraju vrlo slično kao i premještanje samo što nema problema s rastezanjem poligona kada ih nema dovoljno u regiji da bi se postigla deformacija; *dinamičko popločavanje* (eng. *dynamic tessellation*) **(3)**, je pak slično vokselima, no površina se stvara koristeći trokute kako bi se zadržala glatka površina i dozvolili finiji detalji. Nova mreža obično će prebaciti originalnu visoko-razlučivu

[31] Ponderacija (lat. ponderatio) jest mjerenje, odmjeravanje; ravnoteža, ravnomjerna i odgovarajuća raspodjela tereta; u kiparstvu: ravnomjerna podjela težine tijela na pojedine udove; slik. ravnoteža u položaju i kretnji jedne figure.

[32] Verteks (eng. vertex), vrh plošnog popločenja ili tessellation-a jest točka gdje se 3 ili više ploha/ploča susreću. Uglavnom, ali ne nužno uvijek, pločice tessellation-a tj. "popločenja" su poligoni, te vrhovi (eng. vertices) tessellation-a su također i i vrhovi svojih pločica. Generalno, popločavanje može biti shvaćeno kao vrsta topološkog kompleksa stanica, kao što to mogu i plohe polihedrona ili politopa; vrhovi drugih vrsta kompleksa poput simplicijalnog kompleksa (eng. *simplicial complex*) su svoje nul-dimenzionalne plohe. Jednostavno u geometriji verteks (verticies, tj. točke ili vrhovi) jest posebna točka koja opisuje uglove ili intersekcije geometrijskih oblika.

informaciju o mreži u *displacement* podatke ili normalnu mapu podataka ako se koristi motor za igre (eng. *game engine*). Sve u svemu u fazi modeliranja oblikujemo individualne objekte koje ćemo kasnije koristiti u sceni. Dakle, iz nekoliko najpopularnijih načina reprezentacije 3D modela, može se zaključiti i 3 vrste tehnika modeliranja; tehnika stvaranja konstrukcijski čvrste geometrije, tehnika implicitne površine, i tehnika subdivizije površine. [8]

Isto tako vrste modeliranja se mogu podijeliti s obzirom na korišteni softver. 3D softveri se mogu u grubo svrstati na one koji služe za stvaranje 3D modela u umjetničke svrhe i onih koje služe u praktične svrhe odnosno za potrebe inženjeringa. Pa tako možemo razlikovati i dvije vrste modeliranja s obzirom na korišteni softver; a to su parametrično modeliranje (eng. *parametric modeling*) i eksplicitno modeliranje (eng. *explicit modeling*).[35] To su dvije posebne metode definiranja 3D modeliranja. Parametrično modeliranje je uglavnom korišteno kod inženjerski orijentiranih projekata zbog toga što sadrži mnogo preciznije informacije o dimenzijama i odnosima te može uključivati i povijest dizajniranja tj. modeliranja. Eksplicitno modeliranje je tipično korišteno od strane umjetnika i industrijskih dizajnera jer je mnogo fleksibilnije i dopušta provedbu promjena u toku rada koje nužno nisu povezane s nekom drugom definiranom točkom geometrije. Prilikom modeliranja važno je uvijek imati na umu jednu stvar; a to je održavati stvari što je moguće jednostavnijima. Kad god je to moguće uvijek treba odabrati najjednostavniji mogući način modeliranja sa što manje poligona. Što je model kompliciraniji i što su radnje zahtjevnije to će računalu trebati više vremena za procesiranje istog. Također treba pripaziti i na to da mreža nema nikakvih anomalija ili nepravilnih spajanja jer nam to kasnije (prilikom animiranja) može donijeti poteškoće.

3.3.2. Teksturiranje i osvjetljenje

U fazi konstrukcije modeliranje nije jedino što se mora napraviti da bi se zgotovio model. Moraju se još postaviti materijali odnosno boje i teksture. Proces dodavanja tekstura na model naziva se "*texture mapping*". To su zapravo 2D slike koje se dodjeljuju određenom segmentu 3D modela. Kao

što 3D model ima koordinate (XYZ) kroz koje se kreće u 3D prostoru, tako i 2D tekstura ima svoje koordinate (UV koordinate) koje određuju na koji dio površine modela se tekstura postavlja. Vrsta tekstura ima na pretek pa se tako modelu dodjeljuju tzv. *shader*-i odnosno materijali koji imaju različita svojstva poput; difuzije (eng. *diffuse* teksture), *bump*-a (eng. *bump* teksture), mapiranja normala[33] (eng. *normal maps*), pomijeranja (eng. *displacement* teksture), šuma (eng. *noise teksture)* itd.

Osvjetljenje je također vrlo bitan i nikako zanemariv element u CGI animaciji. Svijetlost je vrlo često presudna za kvalitetu i realističnost 3D scene. Bitno je napomenuti da svjetlost u 3D svijetu je samo nepotpuna simulacija svjetlosti u stvarnom svijetu koja je zasad poprilično nedostižna zbog kompleksnih kalkulacija koje bi se trebale izračunavati pri svakom koraku renderiranja scene. Zbog toga za dobro osvjetljavanje koristi se više svjetlosnih izvora. Ima različitih tipova svjetla poput; usmjeravajućeg svijetla, točkastog svijetla, te simulacija sunca itd. Svima se mogu odrediti temperature, odnosno jačinu i toplinu te obojenje svijetla.

Nakon, što se završi s modeliranjem svih objekata i likova i njihovim teksturiranjem odnosno dodavanjem materijala modelima te se postavila rasvjeta na sceni, može se prijeći u slijedeću fazu, *rigging* fazu, a zatim i fazu animiranja. Ova faza može biti odrađena i nakon animiranja, no ako ju se napravi sada tada je poželjno dezaktivirati neke od njenih elemenata prilikom animiranja kako se ne bi dodatno opteretilo računalo, pogotovo, ako modeli imaju, kosu, krzna, trave i druge čestične modele i sl.

[33] *Normal mapping* hrv. mapiranje normala, u 3D računalnoj grafici je tehnika kojom se stvara iluzija svijetlosti na *bump*-ovim (hrv. "udarcima") i *dent*-ovim (hrv. udubljenjima), tj. implementacija *bump mapping*-a. Koristi se kako bi se dodalo detalja bez dodavanja dodatnih poligona. Uobičajeno korištenje ove tehnike jest da se uvelike poboljša izgled i detaljiziranost nisko-poligonalnog (eng. *low polygon*) modela generiranjem mape normala iz visoko poligonalnog modela ili visinske mape (eng. *hight map*)

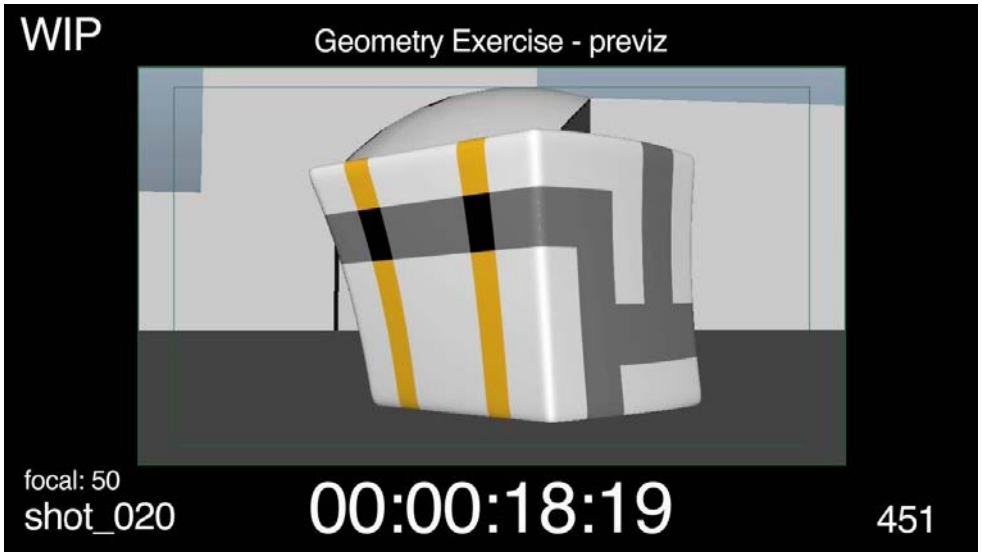
3.3.3. *Rigging*

Rigging je jedna od ključnih faza u animaciji jer ukoliko nije dobro izvedena kasnije prilikom animiranja dolazi do značajnih problema koji mogu poprilično unazadovati čitavu produkciju. *Rigging* je proces stvaranja kosti i zglobova za 3D modele koji će kasnije omogućiti animatorima da ih postavljaju u poze i pomiču odnosno manipuliraju tim 3D modelom kako god žele. Tehnike i tok rada kreiranja *rig*-a je manje više jednak u svim 3D aplikacijama, no dobrim poznavanjem alata s kojima radimo maksimiziramo njihovu učinkovitost. *Rig* je nešto što služi isključivo u svrhu animiranja i ne postoji van 3D softvera. To je set alata koji animator koristi da bi pokrenuo 3D modele te nisu vidljivi u konačnom renderu. *Rig* ili "*armatura*" ovisno o softveru ima i drugačije nazive jest dakle, stvaranje digitalnog kostura. Iako možda na prvu ne zvuči logično, no armatura se gradi isključivo nakon što smo već izmodelirali 3D model i napravili njegovu konačnu verziju. Tek tada se prema tom modelu gradi njegov kostur, a ne obratno. To ne znači da svaka stvarna kost mora imati i svoju 3D kost već su stvari znatno pojednostavljene. Također, ne mora nužno svaka kost pomicati neki dio tijela (tj. *mesh*) 3D modela. Neke kosti služe isključivo kao kormila ili poluge za pomicanje jedne ili više kosti radi lakšeg animiranja i same jednostavnosti korištenja digitalnog kostura. *Rig* se postavlja u centar tijela tj. 3D modela, baš kao i prave kosti, te prati njegov oblik. Isto tako, kosti se mogu koristiti za pomicanje mišića lica, kapaka, očiju, obrva itd. Mogućnosti korištenja *rig*-a su mnoge jer su najbolji i najefikasniji način za pomicanje objekata bez narušavanja originalne strukture *mesh*-a, a izvođenje kompliciranih pokreta, poput hoda, leta, trka i sl. bez njega u 3D-u su praktički neizvodivi. Za neke softvere, poput Blendera, mogu se pronaći već gotovi kosturi koje su izradili 3D kreatori te ih stavili na raspolaganje ostalim 3D umjetnicima, koje je potrebno samo malo prilagoditi da bi odgovarali nekim drugim sličnim modelima. Osim što korištenje gotovog *rig*-a može ponešto skratiti vrijeme *rig*-anja može i poslužiti kao primjer kako da se napravi dobar *rig.* Jednom kada se napravi dobar *rig*, taj se *rig* može koristit beskonačno mnogo puta, dakako, dokle god je on primjenjiv na te likove. *Kosturi* za životinje nisu primjenjivi na ljude i obratno.

No, tehnika i principi izrade jesu. Kod stvaranja *rig*-a bitno je da je što jednostavniji u pokretljivosti kostiju i pomičnosti zglobova. *Rig*-ovi u 3D-u su još uvijek iznimno kompleksni i sastoje se od mnogo *meta*-kostiju i *sub*-elemenata. Zbog toga je iznimno važno poznavanje alata s kojima se barata kako bi se što bolje usmjerile te kosti i zglobovi da kasnije pri animiranju, kada ih se bude konačno povezalo s modelom, omoguće realne i što prirodnije kretnje. Članci i kosti *rig*-a se ne bi trebali pomicati u svim mogućim smjerovima već bi trebali pratiti prirodne smjerove kretnji. Ako se radi o čovjeku tada kost mora pratiti maksimalne i minimalne gibove stvarnih kostiju čovjeka. Vrlo je važno da su kosti međusobno dobro sinkronizirane te da su ovisnosti i težine dobro raspoređene. Kod *rigging*-a nekog lika obično se počinje od postolja (ili trupa) oko kojeg će se dalje graditi čitav kostur, a kost postolja će ujedno biti i glavna kost koja će pomicati sve kosti odjednom. Osnovna kost ne utječe na *mesh* ali je vrlo praktična prilikom animacije hoda tj. pomicanja čitavog modela u svim smjerovima. Od glavne kosti dalje se grade kosti trupa, a desna ili lijeva strana tijela koje mogu biti reflektirane i na taj način se može osigurati da su obje strane kostura jednake. Npr. za kosti trupa kod čovjeka dovoljne su 3 kosti, dok za kosti šake je potrebno napraviti gotovo sve kosti koje uistinu čovjek ima da bi se postigla optimalna pokretljivost šake. Animiranje se znatno olakšava ako se osim kostiju postave i kosti *"kormila"*, odnosno kosti koje pomiču više kostiju i zglobova odjednom, umjesto da se pomiče svaka kost zasebno. Na koncu, kada su sve kosti postavljene na svoje pozicije, kostur se mora povezati sa 3D modelom, i to tako da svaka kost zna na koji dio tijela treba utjecati odnosno što pomicati. Ovaj dio ponekad može se pokazati vrlo zahtjevnim jer se raspolaže težinom i masom tijela, ali i gibljivošću pojedinih udova. Težine se mogu rasporediti manualno ili to može učinit softver, međutim iako on to i učini, vrlo vjerojatno će biti potrebne korekcije. Važno je za napomenuti da u ovoj fazi *mesh* (3D model) mora biti gotovo bez greške jer ukoliko ima kakvih mana to će se amplificirati prilikom aplikacije *rig*-a i određivanja težine te konačno i pomicanja udova.

3.3.4. Predvizualizacija *(previsualization)*

Proces *pred-vizualizacije* (eng*. previsualization*) je stvaranje ogoljene verzije filma prije nego ga se ustvari napravi. Napravljena potpuno u 3D programu, *pred-vizualizacija* može biti animirana vrlo bazično, tek toliko da se prenese smisao i tok priče. Na neki način, *pred-vizualizacija* je u biti 3D animatik. Potrebno je postaviti kutove snimanja i ostale pokrete objektiva (ako već nisu učinjeni). *Previsualization (*kratica *previz* dalje u tekstu*)* daje priliku režiserima da vide kako će pojedine scene po prvi puta izgledati napravljene u 3D-u, pa se tako mogu, po potrebi, prije animiranja napraviti još neke korekcije kadrova ili pokreta u *storyboard*-u. To je posljednja što se čini prije nego se u potpunosti upusti u animiranje. Može se uštedjeti jako puno truda i vremena ako se na vrijeme primijeti da neke scene neće dobro funkcionirati ili da bi bolje funkcionirale na drugačiji način. Ono što je poželjno uvrstiti u *previz* jest: naslov, naznaku da je to WIP (eng. *work in progress*, hrv. rad u procesu stvaranje tj. ne-finalna verzija), *timecode* (hrv. vremenska traka), brojanje *frameova*, (zato što sam *timecode* ponekad nije dovoljan odnosno dovoljno precizan), broj scene i postavke kamere (dubinska oštrina i/ili žarišna duljina).[25]



Slika 3: *Screenshot s postavkama za pred-vizualizaciju (izvor: Animation for beginers, Morr Meroz, 2014)*

Također, prije animacije, poželjno je sprovesti, onoliko koliko se smatra da je to dovoljno, **testnih rendera** kako bi se osiguralo da dotična scena izgleda točno onako kako je i predviđeno, te da boje i teksture odgovaraju onim zamišljenim. Na kraju ne preostaje ništa drugo nego početi animirati tj. udahnuti život 3D modelima.

3.4. Animiranje

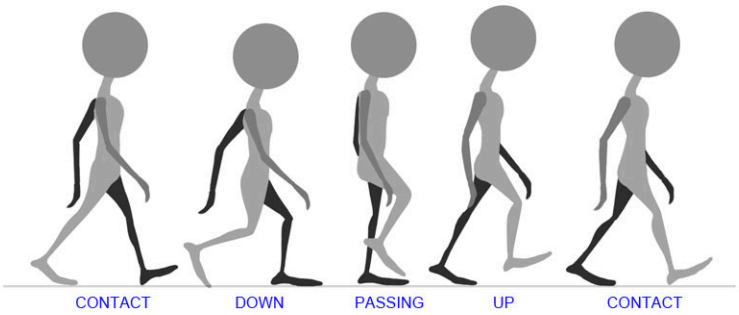
Animiranje je najduža od svih faza u procesu stvaranja jedne 3D animacije. Za dobro animiranje potrebno je dobro poznavanje 3D softvera i njegovih mogućnosti ali i dobro poznavanje okoline. Poznavanje načina na koji se kreću, ljudi, način na koji se kreću životinje, koji izrazi lica se prave dok se izražava neka emocija, kako izgleda udarac ili sudar, kako kiša pada ili puše vjetar itd. Umjetnost animiranja jest u tome da promatranjem i pažljivim poučavanjem se savladaju i što vjernije prenesu stvarni pokreti i kretnje u animaciju. Jedan od najteže savladivih i najzahtjevnijih kretnji za animiranje jest ljudski hod odnosno ljudski ciklični hod (eng. *human walk cycle)*, zatim slijede životinjski hod, trk, skok i mnogi drugi. Ljudi su vizualno osjetljiviji na greške posebice onih pokreta koji su im bliskiji tj. na one pokrete koji se viđaju ili obavljaju svaki dan. Stoga vrlo lako i brzo se uočavaju i najmanje nepravilnosti. Neprirodne kretnje, trzaje ili preskoke makar trajali i manje od sekunde primjećuju se iznimno brzo i precizno. Cilj čitavog animiranja jest stvoriti fluidne, meke i lukovite kretnje. Naši pokreti ali i pokreti sveg ostalog u prirodi nisu četvrtasti tj. "robotizirani" već su polukružni i organski. Primjerice, dok se hoda blago se maše rukama i u zraku se crtaju krugovi i polukrugovi (lukovi). Dobar animator trebao bi znati pronaći najbolji omjer kod pomicanja zglobova a da kretnje ne budu ni previše opuštene ili gumene, ni previše četvrtaste i krute. Dakako, mnogo toga ovisi i o animiranom liku i o tome kakav karakter mu se želi pripisati. Pokreti i način kretanja nekog lika govore jako puno o njegovoj osobnosti, čak i više nego u stvarnom životu zato što su u animaciji emocije i kretnje znatno pojačane a i pažnja gledaoca je mnogo usmjerenija. Vjerodostojnost likova je ključna.

Specifičan način hoda čini lika prepoznatljivim, pa čak i onda kada mu se promijeni vizual. Kretnje trebaju biti uravnotežene tj. bez pokreta zglobova koji u prirodi nisu fizički izvedivi, desna i lijeva strana (ako se radi o hodu) trebaju biti manje ili više jednako animirane, te treba paziti na težište tijela i kako ono utječe na kretnju. Također, treba se pokušati održati kontinuitet pokreta bez naglih promjena ili zaustavljanja i/ili pokretanja. Hod i pokreti se kroz animaciju mijenjaju ali stil u pravilu bi trebao ostajati isti. Animiranje treba podijeliti na sekvence redom kojim će se animirati. Svaka sekvenca se sastoji od nekoliko scena i svaka scena se sastoji od snimaka (kadrova, eng. *shot*).[34] Kadar je definiran kao snimka, djelomična ili u cijelosti, koja se koristi u finalnoj montažnoj verziji filma. To je dio filma između dvije montaže, ''dio vremena''. U igranim filmovima (eng. *live-action)* snimkom se smatra trenutak od kada se počinje snimati pa sve dok se kamera ne zastavi. Zatim u montažnom procesu montažer reže te snimke na vremenski okvir koji mu je potreban.[13] Međutim, u animaciji se animiraju samo ti finalni vremenski okviri za finalnu montažu. U igranom filmu neku scena se može snimati iz 3 različita kuta, dok se u animaciji animira samo onoliko *frameova* koliko je potrebno za svaki kut snimanja. Ovaj način rada nije veoma fleksibilan, no s obzirom da animacija zahtjeva mnogo vremena, na ovaj se način pokušava što više skratiti vrijeme animiranja. Kadrovi su obično veoma kratki, tako da animator uglavnom neće animirati kadrove duže od nekoliko sekundi. Kretnje kamere i kutovi, te *rigovi* bi trebali biti već namješteni prije nego se počne animirati. U te svrhe služe *previz* i testni *pre-renderi*, kako bi prije samog animiranja sve bilo spremno. S obzirom da je animiranje jedna od vremenski najzahtjevnijih faza važno je pokušati vremenski optimizirati i dobro odraditi sve faze jer svaka pogreška koja nije korigirana na

[34] Kadar, jedinica film. izlaganja, dio filma u kojemu se bez ikakvih promatračkih prekida prati prizorno zbivanje. Snimka ostvarena uključenjem kamere i njezinim isključenjem naziva se snim. kadar, a kad je se umontira u filmsko djelo govori se o montažerskom kadru. Kadrovi se u filmskom djelu međusobno razdvajaju montažnim prijelazom. Kadar je određen onim što se u njemu vidi (sadržaj kadra, odnosno prizor u vidnome polju, u izrezu; → mizanscena; → prizor), načinom na koji se to vidi, tzv. parametrima kadra: → izrez, → plan, → rakurs, → pokret kamere), te svojim trajanjem (→ trajanje kadra). Određivanje, tijekom snimanja, što će se i kako vidjeti u kadru naziva se kadriranje, a razdjeljivanje scene, odn. prikaza prizora na uzastopne kadrove naziva se raskadriranje. Kadar se može sastojati i od jedne sličice (kadar-sličica), ali se obično sastoji od više njih, povezanih tako da tipično izazivaju dojam prividnoga kretanja. H. Turković

vrijeme sa svakom sljedećom fazom postaje sve veća, a samim time i svako rješavanje problema produžuje i vrijeme produkcije.

Stvari koje mogu pomoći u fazi animiranja; dok se animira ili tik prije nego počne animirati su; **snimanje referenci** (eng. *shoot reference*), iako se može biti prilično sigurno u to kako se želi da neka scena izgleda uvidjeti će se da ju je u praksi mnogo teže izvesti. Zbog toga je poželjno napraviti živu snimku osobe ili nekog trećeg (životinje, prirode i prirodnih pojava) koje će se uzeti kao reference. Takve snimke osim što služe za detaljnu analizu pokreta i kretnji, mogu se direktno ubaciti u 3D softvere te doslovce po njima graditi animaciju. Iako se može pretpostaviti da se zna kako pas trči ili čovjek hoda, vrlo često samo ideja o tome, nije dovoljno da bi se uspješno prenijela kretnja u 3D svijet. Pokret se sastoji od mnogo gotovo nezamijetnih *među-kretnji* koje pri običnom promatranju često promaknu, a upravo implementacija takvih detalja čini animaciju boljom. Stoga nije na odmet koristiti referentne snimke i gledati ih u *slow-motion*-u kako se ne bi previdjele neke kretnje koje su ključne za uspjeh čitavog pokreta. Također, ako u animaciju ima dijaloga, tada je gotovo obvezno napraviti snimke govora, koje će se direktno koristiti prilikom animacije govora. Reference se mogu uzeti i iz drugih izvora, nije nužno osobno ih snimati. Kvaliteta takvih snimki je manje bitna, dokle god se iz njih može iščitati potrebna kretnja. Jedna od stvari koja također može pomoći jest stvaranje minijature kadra (eng. *thumbnail shot*), ako to crtačke vještine animatora omogućavaju, tako što će se nacrtati glavne poze u koje se mora postaviti likove da bi se napravila zamišljena kretnja. Zatim će se te nacrtane poze koristiti kao ključne poze za postavljanje osnovnih *keyframeova*. Nakon što su se postavili glavni *keyframeova*, trebaju se i doraditi među koraci (oni između 2 *framea*) te se i uskladiti čitava kretnja. Do neke mjere 3D softver može umjesto animatora napraviti pokrete koji nastaju između ključnih poza, međutim gotovo uvijek ih treba doraditi, i više ili manje promijeniti kako bi pokret bio fluidan i bez "skokova". Primjer ključnih poza pri animiranju ljudskog hoda je prikazano na ilustraciji 1. Ilustracija prikazuje 5 ključnih poza koje se ciklično ponavljaju i tako stvaraju simulaciju ljudskog hoda.



Ilustracija 1*: 5 ključnih poza kod ljudskog hoda*
(izvor: https://jimi3d.wordpress.com/tag/walk-cycle/)

Može se zaključiti da je za uspješno animiranje potrebno mnogo promatranja i proučavanja stvarnih pokreta kao i dobro poznavanje alata s kojima se animira Bitno je imati jasnu ideju o tome kakvu osobnost se liku želi pripisati. 3D softveri su tu da nam znatno olakšaju animiranje, no ipak bez "ljudskog dodira" tj. postavljanja parametra i eventualnih popravaka, animacija ne bi bila moguća. Tokom animiranja dobro je napraviti testne *pre-rendere* i to nekolicinu i visoko-razlučivih, koji mogu pomoći da se na vrijeme uoče greške i greškice te da se eventualne nepravilnosti isprave na vrijeme. Animiranje je vremenski i znalački jedna od najzahtjevnijih ako ne i najzahtjevnija faza. Dobro animiranje jest ključ uspjeha neke animacije. Iako ono samo bi bilo znatno otežano da prije nisu učinjene iscrpne pripreme.

3.4.1. *Pre-render*

Pre-renderirano znači da se renderiraju visoko detaljizirani objekti s elaboriranim svijetlom i visoko-detaljiziranim teksturama, često se još zovu i *cut-scene* (hrv. slobodan prijevod; rez- scene). *Real time* je ono što se događa i renderira trenutno i to je uglavnom *game*-ersko okruženje, hardver povlači okolinu, uključujući svijetla, u hodu. Što je hardver jači to će okolina biti detaljnije prikazana u pristojnom *framerate*-u. Dok, *Pre-render* je renderiran ranije i spremljen je u neki izlazni format (npr. *avi, mpeg, quicktime, mov, png,*

jpg, targa itd.). Ono što se praktički gleda jest film. Međutim, *pre-render* nužno ne mora, ujedno, uvijek značiti da su modeli i teksture ‘visoko-razlučive (detaljizirane). Točnije je reći da štogod se do sada gledalo to više nije moguće vidjeti na računalu kao 3D objekt već kao 2D sliku ili seriju 2D slika (slijed ili video). Kad god se neki objekt gleda u nekom 3D aplikacijskom sustavu, taj pogled jest *real time*. U trenutku kada se renderiranjem napravi sekvenca nepokretnih slika koja se kasnije poslože u video ono što se gleda jesu *pre-renderirani* 3D objekti, koji su sada 2D slike. Dakle, pošto je *pre-render* 2D, pomicanje u *real timeu* bilo bi ograničeno na zoomiranje i horizontalno-vertikalno pomicanje. Pomicanje, dakle, jest moguće ali ono što se vidi nije objekt sam već njegova 2D reprezentacija. U *real time* 3D okruženju s obzirom da je hardver sposoban renderirati 2D slike u hodu, možemo pomicati pogled (kameru) u bilo kojem smjeru. Dakle, *pre-render* slike se koriste za stvaranje finalnih ali i testnih rendera. Razlika između jednih i drugih je jednostavno u razlučivosti slike. Veća razlučivost jednako dulje vrijeme renderiranja. Također, ponekad se može koristiti i *real time* reprezentacija unutar 3D softvera za *preview* (hrv. slob. prijevod "pred-pregled") scene, no njena učinkovitost u svrhu pregledavanja scene ovisi o tome kolika je razina razlučivosti dovoljna za dotične potrebe, ali i tome kolike su mogućnosti računala da to izprocesira. Po završetku ove faze produkcija može prijeći u sljedeću tj. faza koja je nazvana završno dotjerivanje, a uključuje; montažu, *compositing*, vizualne efekte i *sound design*. Ako se žele preskočiti neki od sljedećih koraka tj. smatra se da *compositing* i/ili specijalni efekti nisu potrebni, u tom se slučaju mogu odmah izbaciti finalni *pre*-renderi ili se pak može eksportirati animaciju u konačni *output* (hrv. slob. Prijevod izlazni format). Međutim, to gotovo nikada nije praksa kod ozbiljnih produkcija, dok za probne ili "kućne uratke" može biti jedno od mogućih rješenja.

3.5. Dotjerivanje: *Compositing*, specijalni efekti, *sound design i* Montaža

Iako proces produkcije može završiti direktnim eksportom videa ili sekvenci slika iz 3D softvera u kojem je animacija napravljena, to uglavnom nikad nije slučaj. ***Compositing*** *(hrv. compositing)*, iako zadnji u nizu produkcije, ne treba nikako biti podcjenjivan. *Compositing* vizualno može znatno promijeniti izgled animacije; ima moć uniformacije čitave animacije, te može odrediti atmosferu i ton animacije. Može biti izveden direktno u softveru ili se za *compositing* može koristiti neki vanjski softver (što je češće slučaj). Primarna funkcija *compositinga* je korekcija boje, dubinske oštrine *i lens flare*-a i sl. odnosno ono što se radi jest uređivanje fotografija, samo što, kada se govori o 3D animaciji tada se ne radi o stvarnim već o CGI fotografijama. Ono što u *compositing*-u se još može učiniti jest ujediniti slojeve scene koja je prije *raslojena* kako bi je se lakše renderiralo (ako je to bilo potrebno). Na ovaj način se ima nešto veća moć manipulacije nad objektima u sceni tokom faze *compositing*-a. Raslojavanje je također iznimno korisno u slučajevima kada je potrebno neke elemente ponovno renderirati, da li zbog novo nastalih promjena ili greška, te u tom slučaju se ne moraju renderirati čitave scene ispočetka već samo taj element. Time se znatno krati vrijeme ponovnog renderiranja. Dakako, ovako nešto je moguće samo ako postavke scene to omogućavaju, odnosno ako ta promjena ne utječe na ostale objekte na način da ugrožava kompaktnost i logičnost dotične scene. Zadatak *compositing artist*-a nije samo stvaranje finalne gotove animacije već i otkrivanje pogrešaka i razvijanje mogućih strategija *compositing*-a. Razvijanje odgovarajućeg *compositinga* za pojedinačnu animaciju rezultira sveukupnim uravnoteženim izgledom, odnosno svi dijelovi animacije, i kada promatrani odvojeno trebaju biti prepoznati kao dio jedne veće cjeline. To bi značilo da svaka pojedina sekvenca treba imati jednako uređenu fotografiju kako bi se mogla povezati s ostalim scenama s kojima zajedno tvori jedan film.

S obzirom na postojanje dviju različitih grupa digitalnih *compositing* softvera razlikujemo dva načina *compositing workflow*-a. A to su: *node*-baziran (eng.

node-based compositing, (hrv. slobodni prijevod; čvorno-baziran) *compositing* i *layer*-baziran (eng. *layer based compositing* hrv. slobodan prijevod slojevito-baziran) *compositing*. *Node-based compositing* softveri su namijenjeni za visoko-kvalitetne i visoko-budžetne produkcije te su samim time i skuplji. Oni, se suočavaju sa kompleksnim *compositing* zadacima tako što povezuju nekoliko jednostavnih operacijskih slika u finalnu *output* verziju. Na svaku od tih operacija se referira kao čvor tj. *node* i zajedno tvore shematsko čvor-stablo (eng. *node tree*) koje prikazuje *workflow* rada. Na kraju *node* stabla nalazi se *output*-ni *node* koji prikazuje finalnu sliku. Ovakav način *compositinga* pruža nam veću fleksibilnost i veću mogućnost manipulacije pojedinim elementima jer su CGI slike pretvorene u operacijske čvorove. Drugi način *compositinga* je slojeviti *compositing*. Za razliku od *node*-baziranog *compositing*-a, *layer*-bazirani *compositing* radi na potpuno drugačijem principu. Svaki element je slagan jedan iznad drugog tj. slojevito. Dakle, slojevi i specijalni efekti se slažu jedan preko drugog, dok kod *node* sistema svi su istovremenu umreženi do zadnjeg *output*-nog *node*-a.[18]

Node-based compositing softveri su *Nuke* i *Apple Shake* dok poznati *layer-based compositing* softver je *Adobe After Effects*. *Blender* kao i mnogi 3D softveri imaju svoj interni *node-based compositing workflow*.

Neki od ostalih softvera za *compositing* su *i: Flame 3D Visual (VFX)* i *Motion 5*.

Kompoziter radi s direktorom svijetla i FX umjetnicima, kako bi zajedno stvorili finalni proizvod. Kod malih produkcija ove osobe su u biti jedna osoba, i u tom slučaju se treba uzeti u obzir sve ove parametre prilikom procesa *compositing* -a. Cilj ove suradnje jest prije svega postizanje estetski izbalansiranog izgleda. Sve u svemu, *compositing* omogućava da se naprave neke korekcije koje su se previdjele u prijašnjim fazama ili pak nisu uopće mogle biti izvedene.

Vizualni efekti (eng. *visual Effects*) u animaciji često znače kombinacija 3D i 2D tehnika za postizanje željenog efekta. U slučaju 3D animacije uglavnom se misli na dodavanje 2D i svjetlosnih efekata u *editing* programu prilikom *compositing*-a. Može se reći da animacija vizualnih efekata je umjetnost korištenja 3D modela, animacije likova, *rigging*-a, svijetla i ostalih tehnika kako bi se stvorila animacija s određenim vizualnim efektom, ne samo za svrhe CGI

animiranih filmova ali i filma, televizije, reklama, igara i drugih aplikacija. Animacija i vizualni efekti u svijetu računalne grafike najčešće se i koriste zajedno.[17]

S obzirom da je renderiranje dugotrajan proces i odvija se u fazama, ono što se čini u **montaži** jest to da se s vremenom zamjenjuju finalni renderi s onima koje su stvoreni u *previz*-u, pa tako sve do kraja. U međuvremenu, ono što se radi kako bi se maksimalno optimiziralo vrijeme, jest *sound design* (ako je već moguć) i dodavanje specijalnih efekata (po potrebi) te compositing trenutne sekvence (*shot*-a). Pravo editiranje se u biti već dogodilo u *previz*-u i *compositing*-u dok su se uređivale i dodavale novonastale scene. Za razliku od igranog filma gdje se montaža odnosno editiranje događa odmah nakon snimljene snimke (*shot*-a), u animacije ono se događa kontinuirano tokom cijelog procesa; od animatika pa sve do zadnjeg rendera. Najveći dio editiranja i montaže ipak jest obavljeno u *After Effects*-u tako da za manje projekte *After Effects* može poslužiti za eksportiranje finalnog output-a, u suprotnom se mogu koristiti uobičajeni alate za filmsku montažu poput *Final cut*-a ili *Adobe Premier*-a. Ono što se u montaži u biti radi jest pripremanje animacije za finalni eksport.

Sound design, za razliku od *soundtrack*-a znači ukomponiravanje svih zvukova koji se koriste u sceni, što paralelno što jedan nakon drugoga. Radi li se o dijalogu ili monologu (ljudskom govoru), šumu vjetra, zavijanju pasa, razbijanju stakla, pozadinskoj glazbi (posebno pripremljenoj ili već nekoj postojećoj) itd. njihovo montiranje, editiranje i u nekim slučajevima i stvaranje spada pod **sound design**. Zvuk u *editing* programu ima svoju traku te prati tok animacije. Poželjno je imati različite trake za glazbu, dijaloge/monologe te zvukove iz prirode i sl. U principu jednako je kao i filmsko editiranje samo što se ovdje bavi zvukom a ne slikom. Kako izlaze nove scene tako ih se umeće u *timeline* i istovremeno se umeću njihove pripadajuće zvukovne snimke. **Sound design** je kreativan proces, glazba i zvukovi znatno utječu na ritam animacije. Ponekad treba posvetiti malo više vremena traženju točno odgovarajućeg zvuka ili pak razmišljati van-standardski ako se nekim zvukom želi obilježiti neki važni trenutak, pogotovo ako ne postoji u prirodi nešto slično. Kod velikih produkcija

za *sound design* zadužen je čitav *team* ljudi koji se bave isključivo ovim dijelom produkcije. Ponekad, pripreme počinju puno prije nego se završi s animiranjem kako bi *sound design* bio gotov u isto vrijeme kada i finalni renderi. Stvaranje *sound design*-a, ako nije prezahtjevno može biti napravljeno i u *editing* programu zajedno s renderiranim snimkama ili pak profesionalnije u nekom programu za *editiranje* zvuka, te kasnije ubaci u program za montažu, u traku predviđenu za zvuk. Kod manjih produkcija to je rjeđe potrebno, s obzirom da se ne radi o velikoj količini zvukova, može ih se jednostavno kontrolirati i u običnom *editing*/montažnom programu (npr. *Adobe After Effects ili Premier*).

3.5.1. Eksport

U ovom pod-poglavlju samo će se ukratko pokušati objasniti neke od eksport mogućnosti za finalni video zapis. Ne postoji jedinstveni obrazac već svaki softver ima svoje načine i formate, pa tako eksportiranje videa uvelike ovisi o softveru za montažu ali i mediju na kojem će biti prezentiran. Prilikom eksportiranja u finalni format mora se pripaziti na neke od osnovnih postavka, a to su: **format** (npr. ekstenzije; .mp4, .wma), ***codec*** (kratica za koder/dekoder npr. H.264 za MP4), ***framerate*** (npr. u filmu je uglavnom 24 fps[35], još su uobičajeni 25 fps i 30 fps) i **audio postavke** (standardna kvaliteta je 44.1 kHz i 128 kbps.), ***data rate*** (što je manji broj manja je i kvaliteta ali i manja veličina datoteke) i ***deinterlacing*** (bitno za *online* videe), ***aspect ratio*** (hrv. slobodni prijevod; omjer aspekta, 4:3 i 16:9 su dva najpoznatija omjera), **rezolucija** (mjeri se u pikselima, standardni (4:3) video je tipično 640 x 480 px (širina x visina), a *widescreen* (16:9) video je često 853 x 480 px, dok je HD video (16:9) 1280 x 720 px – pola veličine (također 16:9) bi bilo 640 x 360 px) i ***keyframing*** (uglavnom automatsko, veći broj povećava i veličinu datoteke). [21] Važno je prvo eksportirati video u visokoj rezoluciji jer iz više rezolucije se može lako i bez većih problema preći u nižu. Isto tako, važno je da tokom svih ''eksporta'' i

[35] [35] FPS je kratica za *frames per second*, *hrv. frame* po sekundi. Npr. 25 frameova se izvrti u 1 sekundi.

rendera se ima isti *framerate*, kako kasnije ne bi došlo do problema s usklađivanjem *timecode*-a ili eventualnim usporavanjima ili blokiranjima *slika*. Dok su renderi iz 3D softvera gotovo uvijek u nekom slikovnom formatu (*jpg, png, targa* i sl.), finalni eksport format će uvijek biti u video obliku.

4. Zašto animirati u *open sourceu*; *Blender*

Blender nastaje 1988. po ideji Tona Roosendaala[36], koji je tada radio u jednom nizozemskom studiju za animaciju, kao *art director* i *software developer*, gdje polako uviđa da dosadašnji 3D alati koje koristi trebaju biti kompletno promijeni tj. napisani ispočetka. Pa tako 1995. počinje pisati kod za 3D softver koji će danas biti poznat kao *Blender*[37]. Osniva tvrtku posvećenu razvoju 3D alata i softvera *Blender* koja nažalost zbog loše prodaje i ekonomske situacije staje s projektima pa tako i s razvojem Blendera. S obzirom da ponovno pokretanje tvrtke s velikim timom developera nije bilo moguće, 2002. Roosendaal pokreće *non-profit* Blender fondaciju. Jedan od ciljeva fondacije jest bio pronaći način kako nastaviti s razvojem i promocijom *Blendera* kao *open source* projekta. Od kada je pušten u optjecaj po uvjetima generalne javne licence, Blenderov razvoj proizlazi od tima vrlo posvećenih volontera diljem svijeta, koji su vođeni originalnim kreatorom. Upravo, zbog otvorenog pristupa koji je omogućio dnevne interakcije, te primanje povratnih informacija od strane korisnika ali i od developera, Blender ima izvanrednu mogućnost vrlo brzog unaprjeđivanja i razvoja. Nove i poboljšane verzije izlaze mnogo češće nego što bi to bio slučaj kod komercijalnih proizvoda, pa tako unaprijeđene verzije izlaze i po nekoliko puta godišnje. No, upravo ta otvorena komunikacija koja je Blenderu omogućila tako brz eksponencijalni razvoj pokazala se u neku ruku dvosjeklim mačem, zato što se javljaju problemi održavanja i organizacije upravo takve komunikacije. Blender je stvoren od stotinjak aktivnih volontera iz čitavog svijeta; od studija/tvrtaka, individualnih umjetnika, profesionalaca, hobista, znanstvenika i studenata, *VFX* eksperta i animatora itd. a ono što ujedinjuje svih njih jest želja za jednim pristupačnim i potpuno slobodnim/ otvorenim izvorom u

[36] Osnivaču i kreatoru Blendera Ton Roosedaalu sveučilište Leeds Metroplitan dodijelilo mu je počasni doktorat iz tehnologije za njegovu kontribuciju kreativnoj tehnologiji.

[37] Logo i ime brenda zaštićeni su *copyright-om* (ne čine dio GNU GPL licence), te mogu biti korišteni samo od strane Blender Fondacije na njihovim proizvodima, internetskim stranicama i publikacijama. Takav pristup je učestao i kod drugih *open source* proizvoda ne toliko zbog moguće zarade na brendu, koliko zbog toga da se originalni proizvod razlikuje od svojih varijacija i modificiranih verzija, kojima se dopušta postojanje, razvijanje i daljnje modificiranje, ali ne i korištenje imena i loga brenda Blender.

stvaranju 3D kreacija[38]. Blender fondacija podržava i potiče te ciljeve te zapošljava vrlo mali tim, no uvelike ovisi o *online* zajednici kako bi postigla te ciljeve. Zašto je ta *online* zajednica toliko bitna? Kako je Blender nastao iz ideje dijeljenja i zajedničke kontribucije tako se ta ideja proširila i na nus-proizvode odnosno 3D modele i animacije koji iz njega nastaju. Mnogi 3D *artist*-i svoja dijela *upload*-aju *online* i daju ih na korištenje i proučavanje svima koji to žele. Tako da svi koji to požele mogu učiti iz tuđih 3D modela što nije vrlo Uobičajeno za zajednicu nekog *closed-source* softvera. Upravo iz tih razloga Blender se može smatrati jednim od najpodobnijih 3D alata za početnike. Međutim, to ne ograničava nikoga da ga koristi za mnogo zahtjevnije produkcije. Blender fondacija izbacila je nekoliko animiranih filmova napravljenih u Blenderu (npr. *Sintel*) koji dokazuju da svakim novim *upgrade*-om uvelike dostiže kvalitetu komercijalnih i znatno skupljih softvera. Također, s obzirom da je zajednica veoma *responsive*[39], za gotovo svaki problem lako će se pronaći odgovor ili ga se može postaviti, te u vrlo kratkom roku netko će se zasigurno odgovoriti. Kao što je već rečeno, Blender alati, sa svakim *upgrade*-om postaju bolji pa tako i *rig* koji je ključan za uspješno animiranje, tekućine i UV mape su odlično razvijene, a s obzirom da je to softver otvorenog koda, odličan je za korištenje kodova u svrhu 3D-a ali i omogućava da se sam softver prilagodi potrebama korisnika (naravno ako vještine i znanje to dopuštaju). Upravo kako bi se istražila i testirala podobnost Blendera za stvaranje jedne CGI animacije, u eksperimentalnom dijelu rada (poglavlje 7.) napravljena je jedna kratka i jednostavna animacija, koja je u nekoj mjeri pokušala pokriti što više osnovnih Blender alata koji su potrebni za re-kreiranje određenih scena. *Workflow* njene produkcije kroz slike i opise opisan je u poglavlju 7.

[38] Slobodan prijevod*; free/open source 3D creation pipeline*, englesko objašnjenje: Pipeline; Computing: A linear sequence of specialized modules used for pipelining (Oxford dictionary). Prijevod. Linearna sekvenca specijaliziranih modula koji se koriste za sprovođenje (u proizvodnom traku). U ovom slučaju alati te popratni alati potrebni za kreiranje 3D proizvoda

[39] Eng. doslovni prijevod – uzvratno, osjetljivo, kontekstualno objašnjenje: fleksibilan servis koji je osjetljiv na promjenjive društvene obrasce

5. Usporedba 3D softvera

U tablicama 1,2 i 3 izlistani su neki, ali ne svi 3D softveri koji spadaju u istu kategoriju softvera kao Blender. Korištenje 3D softvera koji odgovara osobnim potrebama može znatno utjecati na kvalitetu i vrijeme produkcije neke animacije ili 3D grafike, ali nije presudno. Važnije je dobro poznavanje tog softvera kako bi se iz njega mogao izvući maksimum. 3D softveri, kako je već spomenuto, mogu se ugrubo svrstati u one koje služe umjetničkoj svrsi i u one za potrebe inženjeringa. Pa tako Blender, 3Ds Max, Autodesk Maya, Newtek Lightwave i Maxon Cinema 4D spadaju u kategoriju umjetnički orijentiranih softvera na bazi eksplicitnog modeliranja. 3Ds Max, Maya i Lightwave smatraju se uistinu profesionalnim alatima u svijetu 3D animacije. Oni uključuju brze motore za renderiranje (eng. *render engine*), koji podržavaju iznimno napredne značajke poput realističnih refleksija i refrakcija, radioziteta (eng. *radiosity*), kaustičnosti (eng. *caustics*) i render *nodsa*. Obično se koriste u filmskoj industriji za visokobudžetne projekte. Usporedbe radi postavljeni su u kolone te su ispod svakog navedene neke od njihovih osnovnih značajki i mogućih prednosti, koje bi mogle utjecati na odluku o odabiru određenog softvera. Rečeno je i koju poziciju zauzimaju u usporedbi s ostalima. Usporedbe su nastale isključivo iz teoretskog proučavanja i analiziranja softvera prema specifikacijama utvrđenim od izdavača, *feedbacku* korisnika, te radovima koji se mogu pronaći na internetskim stranicama posvećenim CGI-u i 3D modeliranju. Nije sprovedeno praktično i osobno istraživanje svih pojedinih karakteristika, stoga ne treba na ove tablice gledati s izrazitom preciznošću, već one više služe kao indikacije i smjernice pri izboru i usporedbi kvaliteta i učinkovitosti 3D softvera.

Tablica 1: Generalne razlike između softvera

	Autodesk 3Ds max	Autodesk Maya	Blender 3D	Maxon Cinema 4D	Newtek Lightwave
Cijena aproksimativna	5.000 €	2.500 €	0 €	800 €	800 €
kompatibilni OS	Windows	Windows, Mac, Linux	Windows, Mac, Linux	Windows, Mac	Windows, Mac
Ciljano tržište	VIZ, video igre	Video igre, film	VIZ, realtime	motion design	VFX, VIZ, w/LWCAD
Trial verzija	da	da	ne	da	da
jezik	EN, FR	EN	EN, mnogi	EN, mnogi	EN, JP
Popularnost u industriji	vrlo dobra	vrlo dobra	slaba	dobra	vrlo dobra
Vrijeme savladavanja softvera	< 2 mjeseca	< 3 mjeseca	< 3 mjeseca	< 1 mjesec	< 2 mjeseca
Tehnička podrška	loša do dobra	dobra	slaba	dobra	vrlo dobra
Podržavajući sadržaj	dobar	dobar	online zajednice	odličan	vrlo dobar
interface	čist, snažan	fleksibilan, snažan	ne standardan	čist, intuitivan	star, tekstualan
dokumentacija	dobra	odlična	dobra	vrlo dobra	odlična

Tablica 2: Usporedba načina renderiranja

Rendering					
Rendering	interno, Mentalray	interno, Mentalray	Interno, Cycles	Interno	Interno
Kvaliteta	odlična	odlična	dobra	dobra	odlična
Plug in obvezan	Vray	Vray	Yafray/Indigo	Adv. Render	nema
Kvaliteta w/plug in-a	odlična	odlična	vrlo dobra	vrlo dobra	odlična
Textures baker	vrlo dobar	vrlo dobar	dobar	vrlo dobar	odličan

Tablica 3: Usporedba 3D alata

3D alati					
Alati za animaciju	vrlo dobri	odlični	vrlo dobri	dobri	dobri
UV alati	vrlo dobri	odlični	odlični	odlični	dobri
Oslikavanje	nema	vrlo dobro	slabo	odlično	nema
Modeliranje	odlično	vrlo dobro	dobro	vrlo dobro	odlično
Modifiers	odlično	vrlo dobro	dobro	dobro	dobro
NURBS	slabo	vrlo dobro	slabo	nema	nema
Dynamics/Rigid bodies	vrlo dobro	odlično	vrlo dobro	vrlo dobro	vrlo dobro
Soft bodies	vrlo dobro	vrlo dobro	dobro	dobro	dobro
Kosa	vrlo dobra	vrlo dobra	dobra	vrlo dobra	dobro
Tkanina/odjeća	vrlo dobra	vrlo dobra	dobro	vrlo dobro	nema
Čestice	odlične	vrlo dobre	dobre	dobre	vrlo dobre
Tekućine	nema	vrlo dobre	vrlo dobre	nema	dobre
Kompoziting	nema	nema	da	nema	da
Node-based workflow	nema	da	nema	nema	nema
Node-based materijali	plug in-ovi	da	da	nema	odlični
Node-based compositing	nema	nema	da	nema	nema
CG shader/games	odlično	odlično	da	nema	da
Kodiranje	dobro	odlično	vrlo dobro	dobro	dobro
Jezik kodiranja	Maxscript	MEL, Python	Python	expresso coffee	Lscript
C/C++ jezik	dobro	vrlo dobro	vrlo dobro	vrlo dobro	vrlo dobro

Podaci su do neke mjere subjektivni te postoji mogućnost je sa novim verzijama je došlo do promjena ili *upgrade*-a u softverima pa su samim time i neke značajke bolje nego što su ovdje prikazane. Ovi podaci izvedeni su iz detaljne analize 3D softvera koja je prezentirana u članku kojeg je napisao Benoît Saint-Moulin.[4]

Iz ovih tablica može se iščitati da su pojedini softveri, poput *Maye* bolji za filmsko i *VFX* animiranje, dok neki, poput *Cinema 4D*-a, imaju vrlo dobro sučelje koje omogućava brže savladavanje softvera te su tako pogodniji za studente. *3Ds max* i *Lightwave* su odlični za modeliranje, Blender izvrsno funkcionira s programskim kodovima te ima jednako dobro razvijen sistem *UV unwrap-a,* isto kao i *Cinema 4D* i *Maya*. *Maya* i *Blender,* također imaju izvrsne simulacije

tekućine, dok *Lightwave* ima odličan sistem *texture baker*-a. Za zaključiti jest da je softver dobro odabrati ovisno o zahtjevnosti projekta, sposobnosti kreatora ali i financijskim mogućnostima, te s tim predodžbama zakoračiti u svijet 3D-a. Također, za studente pri kupnji softvera postoje razne pogodnosti, pa tako recimo *3Ds max* nudi višegodišnje besplatno korištenje u svrhu edukacije.

6. Posljedice *open sourcea* i utjecaj na *mainstream* proizvodnju

Jako puno inovacija je pokrenuto upravo zahvaljujući *open sourceu*. Uvelike i zbog činjenice da nisu potrebna nikakva dodatna dopuštenja da bi se stvorilo nešto novo i odlično. Proizvodi poput *Arduino* matičnih ploča i 3D printera počeli su potaknuti željenom da se stvore platforme koje bi studentima bile jednostavne za korištenje i koje bi omogućile izgradnju stvari i proizvoda koji uistinu funkcioniraju. Proizvod poput *Arduina* je kompletno *open source*, sve informacije o dizajnu se mogu pronaći na njegovim *web* stranica. Isto vrijedi i za *wordpress.com* (i njegove teme) koji je zaštićen GPL licencom itd. Doduše njihov *trademark* je pod *copyright*-om. No, to je uobičajena praksa kako bi se originalni proizvod razlikovao od svojih mogućih inačica, dok sam proizvod ostaje uvijek slobodan. Ono što je konačni rezultat ovakvog načina poslovanja jest da se dugoročno razvila jedna snažna zajednica *open source* pobornika koji čvrsto vjeruju u *off-* i *on-line* suradnju te u međusobno učenje i podučavanje. Ima iznimno mnogo primjera gdje škole, učenici i studenti izrađuju uspješne projekte koristeći alate koji su *open source*. Na ovaj način se daje prilika školama i ostalim nisko budžetnim organizacijama i institucijama, koje imaju ograničene resurse, da razviju inovacije i naprave ponekad važne tehnološke iskorake. Jedna od prednosti *open source* aplikacijskih sustava je ta što se korisniku dopušta da personalizira i modificira proizvod po svojim potrebama, te tako se u isto vrijeme kreira jedinstveni proizvod na tržištu. Iz tih razloga stvorene su mnogo male proizvodnje i proizvodi bazirani na osobnim potrebama, što se možda gotovo nikad ne bi dogodilo da se koristio *closed-source* program zato što napredak i mogućnost da se istraži (ali i modificira) način na koji radi je uvelike limitirajući ili gotovo nemoguć. Učenjem se ustvari stvara vrijednost. Ovakav način učenja nije diskriminirajući prema onima koji nemaju novaca da bi učili odnosno kupovali skupe softvere koji su im potrebni za učenje. [2]

Ako se moć prebaci na potrošača tj. korisnika koji je proaktivan, strastven, poticajan ali prije svega i obrazovan te mu se daju alati s kojim može baratati, tada nastanu kreacije koje su stvorene iz stvarnih potreba i želja korisnika. Pobornici *open sourcea* složili bi se da nema direktnijeg načina za inovaciju i kreaciju. Danas je inovacija uglavnom prepuštena velikim komercijalnim kompanijama koje nemaju uvijek na umu da zadovolje stvarne potrebe njihovih potrošača. Ideje potrošača su uvijek ispred onih od proizvođača. Ovakvi novi načini organiziranja bez nekakve formalne organizacije, pa tako i *open source* zajednice su dokaz da je takav način interakcije održiv i da je kreativnost potaknuta upravo tom međusobnom suradnjom, interaktivnošću ali prije svega i otvorenošću. Danas se ideje i proizvodi patentiraju i smatra se da osoba koja je patentirala proizvod je apsolutno sigurna kako se taj proizvod najbolje može koristiti te se ne ostavlja mnogo prostora za druge moguće developere da taj proizvod upotrijebe ili preoblikuju na druge načine. S obzirom da je radikalna inovacija nesigurna odnosno komercijalne kompanije neće se angažirati oko kompromitirajućih i nesigurnih ulaganja i inovacija ako u tome nema sigurnog uspjeha odnosno povrata uloženih sredstava, tada nije ni vrlo vjerojatno da će se takva investicija i dogoditi. Ono što je vjerojatnije je da će se dogoditi jest da će se već postojeći proizvod za koji je sigurno da će donijeti dobit adaptira ili unaprijedi. *Open source* ulazi upravo u to sivo područje gdje *closed-source* se ne usuđuje ulaziti. Iz tog razloga eksponencijalno je veća mogućnost za kreativnost, kreaciju i inovaciju.[2]

Closed-source se fokusira na pasivne potrošače, dok potrošači *open source* aplikacija su aktivni. Za *Open source* se može reći je inovacija potaknuta od strane potrošača. Niti jedna tvrtka neće investirati novac u proizvod koji se mora natjecati s vrlo sličnim proizvodima na tržištu koju su uz to već i komercijalno etablirani. Stoga upravo takvi proizvodi su prepušteni *open source* zajednicama. Pitanje koje se postavlja jest može li takav sustav preživjeti samo na volonterima? Ne trebamo li organiziraniji, podržaniji, strukturiraniji i financirani *open source*? Kakve promijene trebamo napraviti u javnoj politici i financiranju da bi to omogućili? Trebamo il Crveni križ za informaciju? Ova pitanja, diskusije

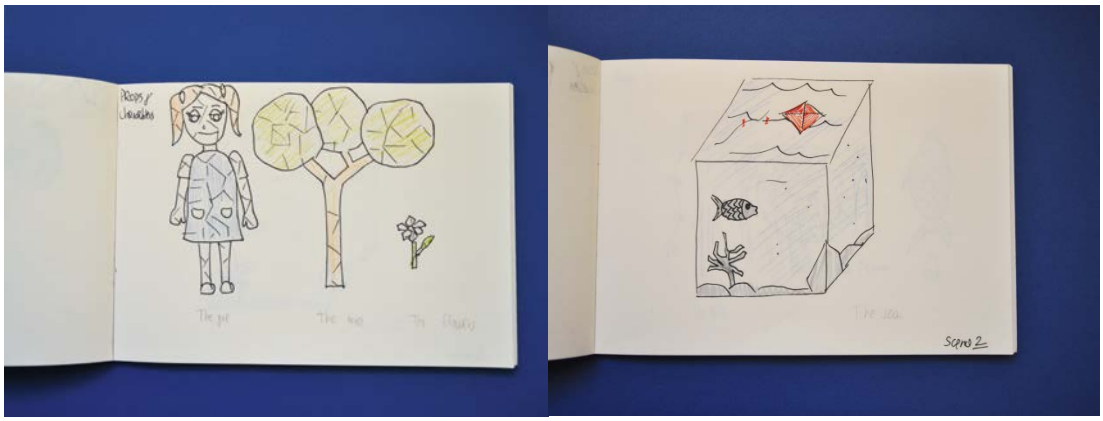
i konstatacije postavlja Charles Leadbeater[40] u svom govoru za TED [2] te smatra da u svakom slučaju *closed-source* se sve više razvija u smjeru *open sourcea*, pa tako možda iz ovog međuprostora izađe nekakav novi organizacijski model. Model koji možda ima potencijala da postane iznimno moćnim. *Open source* multiplicira produktivne resurse zato što korisnike pretvara u proizvođače a potrošače u dizajnere! Pa tako vođena ovim idejama možda i *open source* animacija postane ako već i nije dio jedne velike kreativne kolaboracije.

[40] Charles Leadbeater je britanski autor i bivši savjetnik, bivšeg premijera Ujedinjenog kraljevstva Tonyja Blaira. Čest govornik na TED konferencijama. Istraživač na Londonskom *Think thank Demos*-u. Glavni interesi istraživanja i proučavanja su mu amaterska inovacija i nove poslovne strategije.

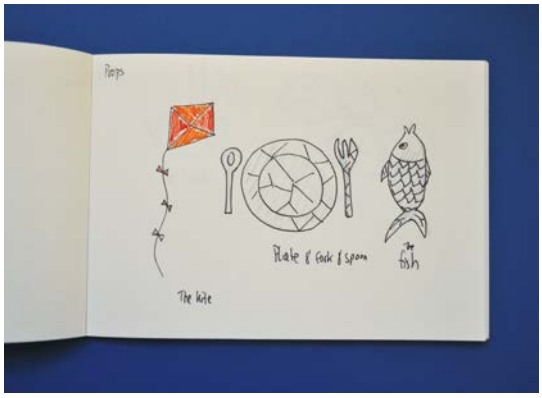
7. Eksperimentalni dio

Ovaj eksperimentalni dio rada imao je za cilj kreirati jednu 3D animaciju u *open source* softveru *Blender*, istovremeno se prošlo kroz sve navedene faze produkcije jedne CGI (3D) animacije, a da su se pritom pokušale što više istražiti mogućnosti i potencijali *open sourcea* i *open source* aplikacija kao takvih. Ova je animacija napravljena u *cycles* renderu pomoću softverskog unutarnjeg motora. Sastoji se od 4798 i *framea*, odnosno u trajanju je od 3 minute i 12 sekundi, s *framerateom* od 25 fps. Veličina izlaznog videa je 1920 px (dužina) puta 1080 px (visina) odnosno HDTV 1080 25. *Compositing* i montaža napravljeni su Adobe After Effects-u. Renderi iz Blendera su sekvence PNG slika.

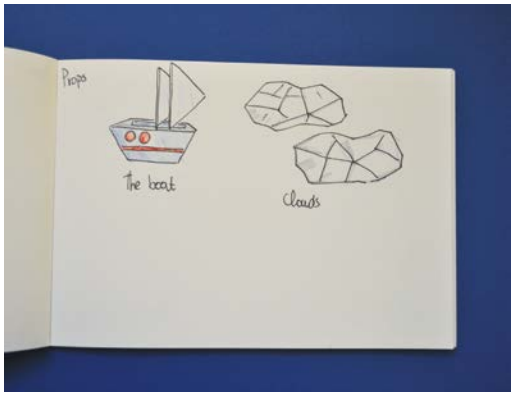
Slika 4: *Concept art & Character design*



Slika 5: *Props (rekviziti) i elementi scene*

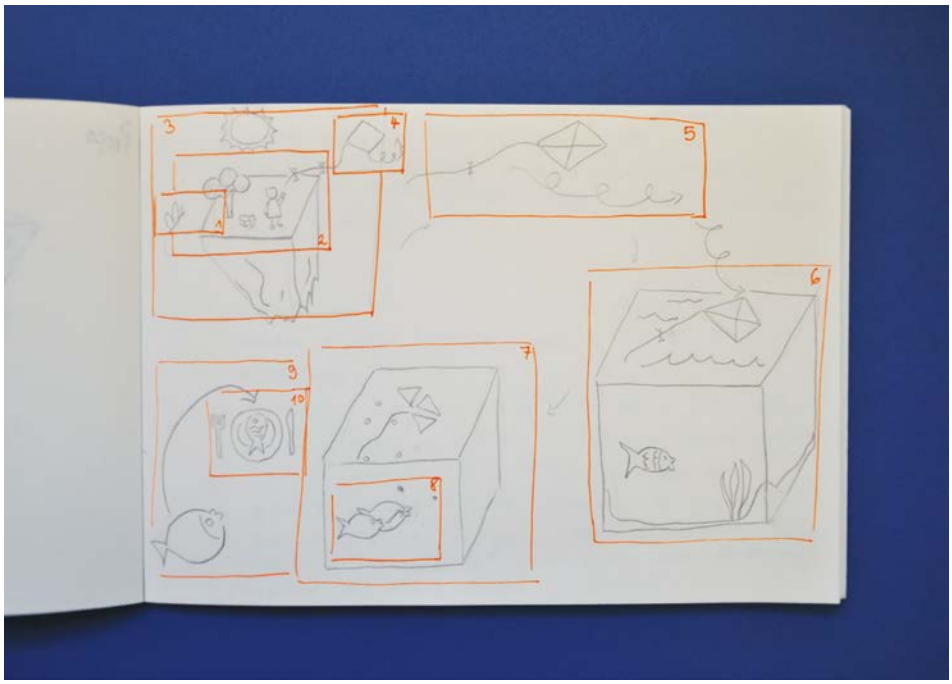


Slika 6: *Props (rekviziti)*



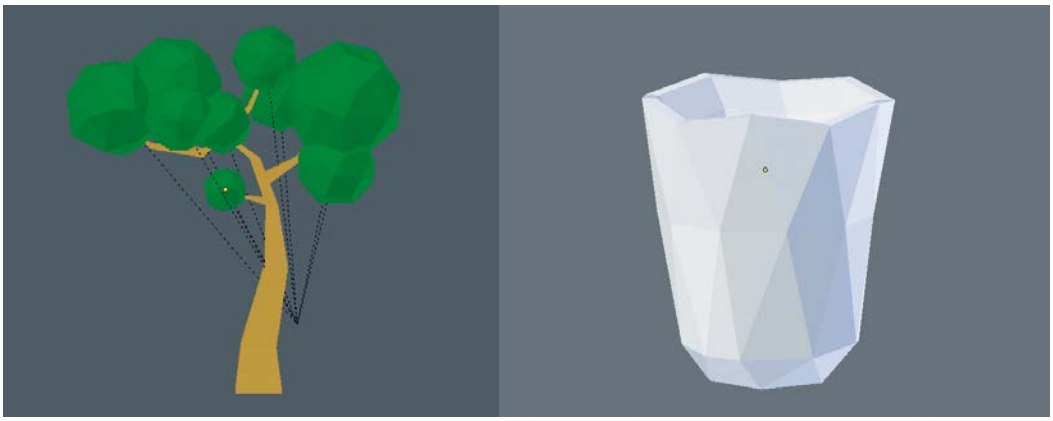
Slika 7: *Props (rekviziti)*

Na slikama 4-7 može se vidjeti razvoj koncepta koje je napravljen prostoručno; olovkom, bojicama i flomasterima na papiru. Slika 9 prikazuje dio *storyboard*-a.



Slika 8: *Segment iz storyboard-a*

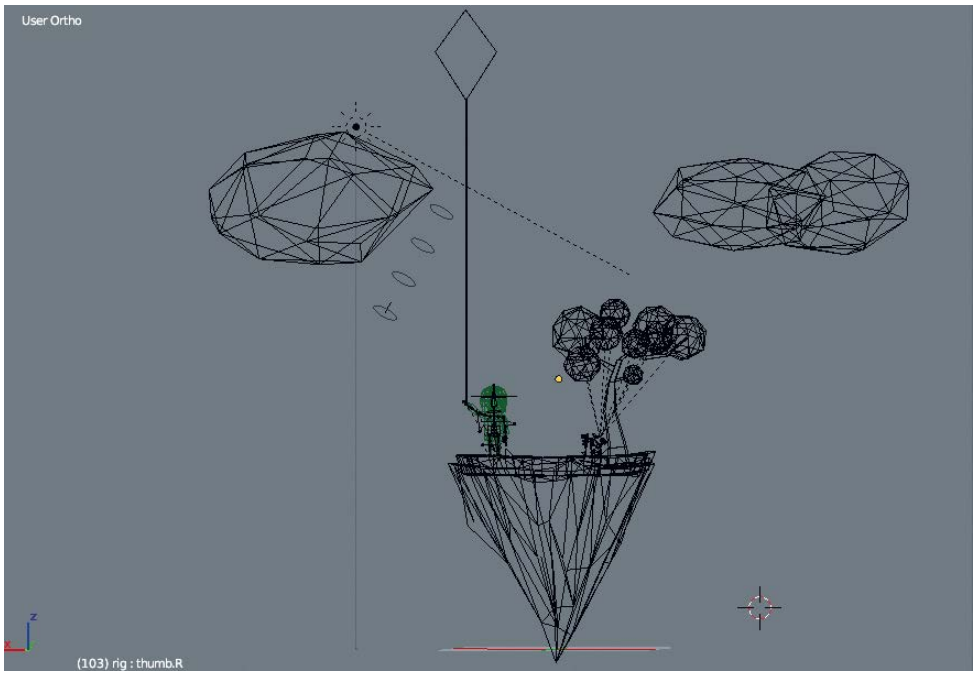
Na slici 8 može se vidjeti dio *storyboard*. Numerirani kvadrati u pojedinoj sceni insinuiraju promjene kadra. Strelicama su prikazane kretnje objekata unutar scene.



Slika 9: *Elementi scenografije u 3D-u*

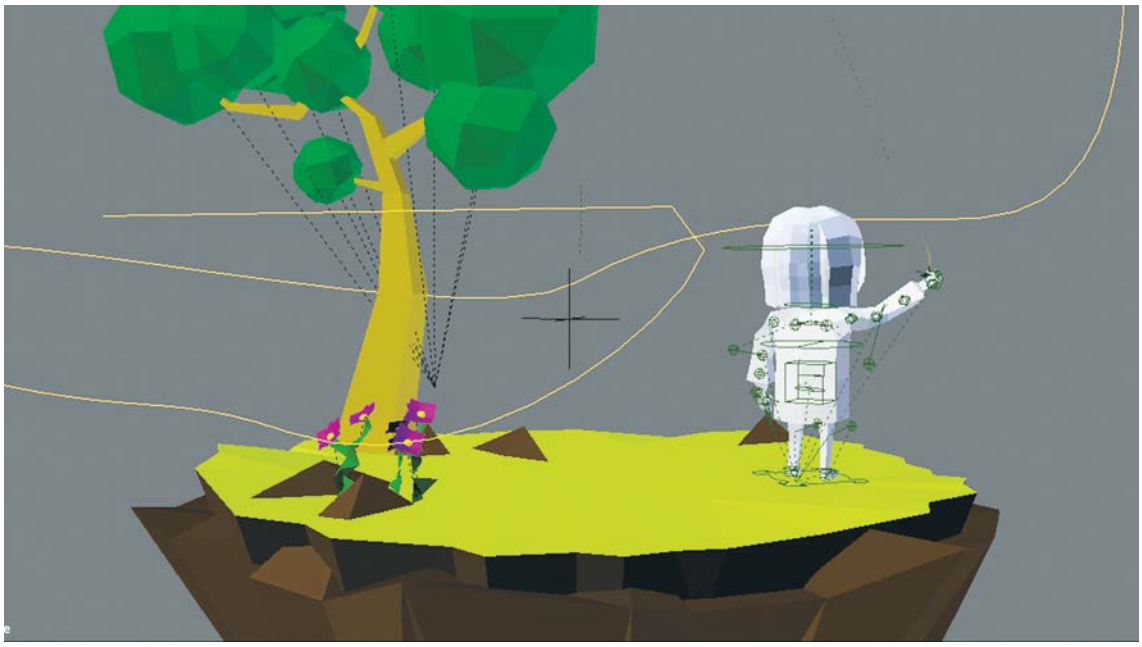
Slika 10: *Rekviziti u 3D-u*

Slika 10 prikazuje 3D modele dvaju rekvizita iz animacije a slika 9 prikazuje neke od elemenata sceni. Paralelno s modeliranjem scena (“*backgrounda*”) i modelima protagonista razvijani su rekviziti i drugi elementi koji čine scenografiju.



Slika 11: *Scena u wireframe modu*

Na slici 11 prikazana je prva scena animacije. Također, na slici 11 vidiljivi su elementi koji u 3D softveru predstavljaju vjetar i sunce, te srodstvo objekata odnosno *parenting* za tu scenu. *Parenting* je naznačen iscrtkanim ravnim linijama koje pokazuju povezanost objekta djeteta (eng*. child*) i objekta roditelja (eng. *parent*). Kao što se može uočiti *mesh* se sastoji od mnogo četvrtastih poligona. U ovoj sceni koristila se simulacija vjetra i *displacement* modifier za *randomiziranje* površine visećeg otoka. Koristio se i *cloth* efekt za potrebe zmaja koji lebdi na vjetru.

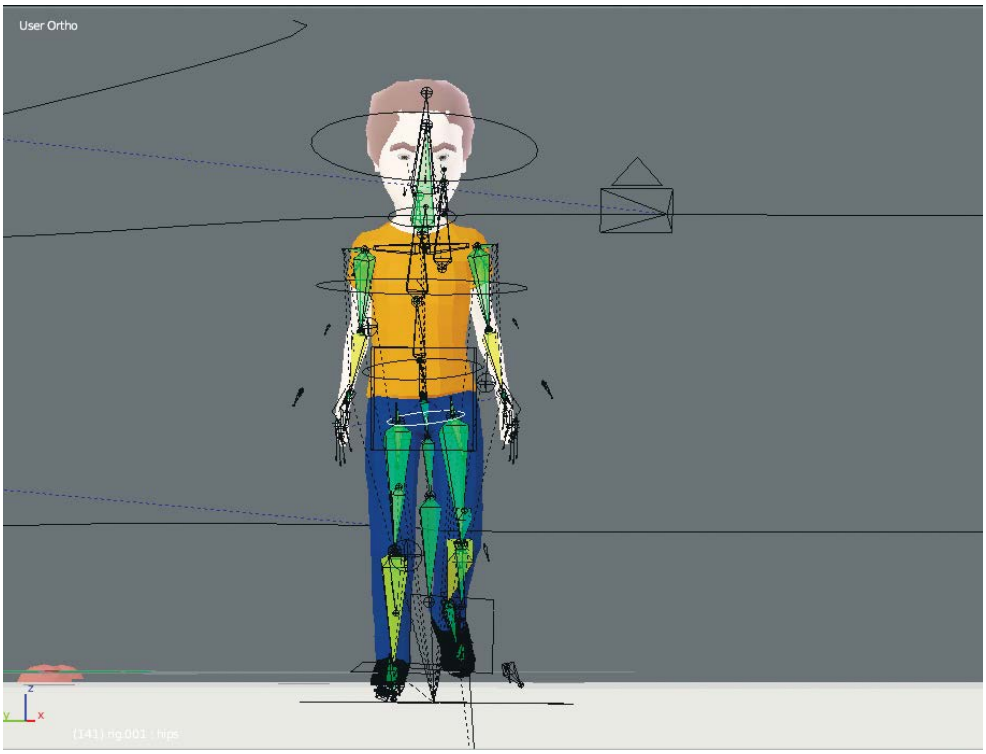


Slika 12: *Segement iz Previsualization-a*

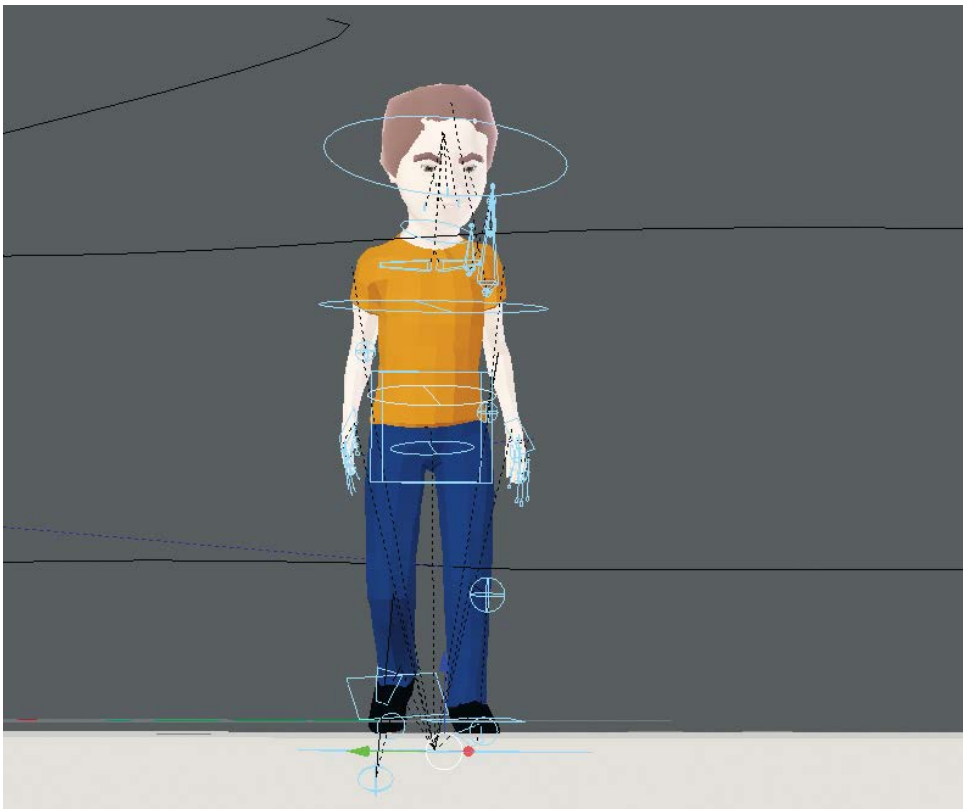
Slika 12 prikazuje dio scene iz *previz*-a. Može se uočiti putanja koja je postavljena pomoću *Bazierove* krivulje koju kamera slijedi prilikom snimanja. Trodimenzionalni križ na sredini scene predstavlja točku fokusa objektiva.

Slika 13: *UV unwrap teksture za model djevojčice*

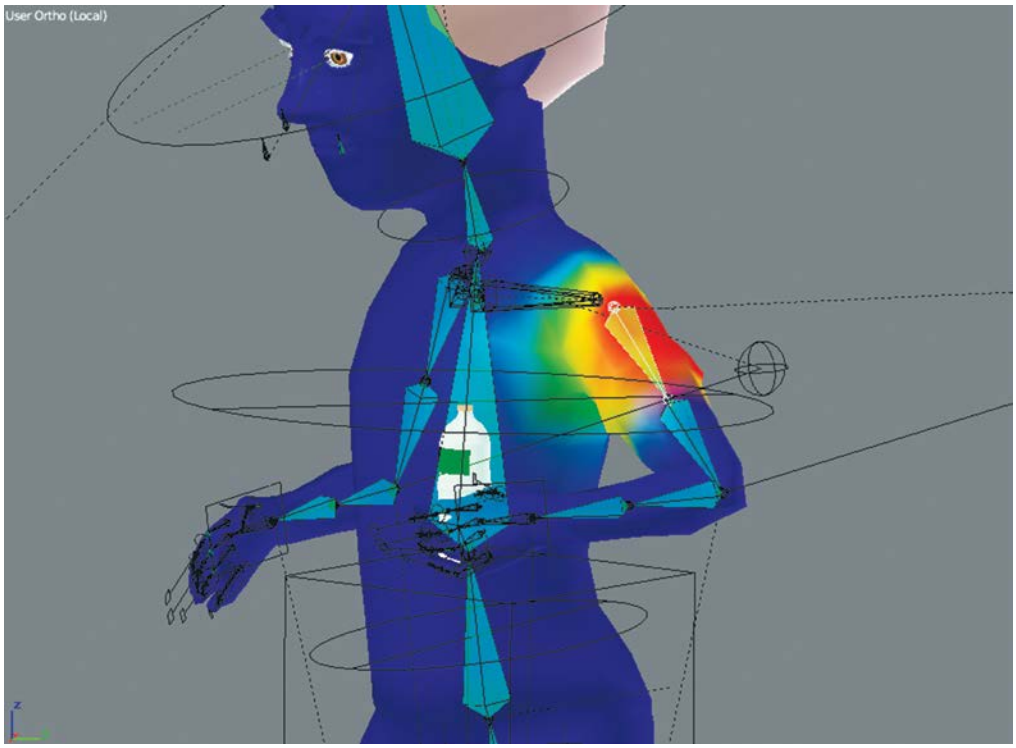
Slika 13 predstavlja *UV Unwrap* teksture za model djevojčice odnosno predstavlja omot koji kada se (pomoću UV koordinatama) postavi/skine na/sa model(a) predstavlja njegov materijal. Napravljen je tako da su “razrezani” dijelovi *mesh-a* kako bi imali više mogućnosti prilikom obojenja. Osim bolje preglednosti takvu teksturu moguće je eksportirati za van-softversko uređivanje. Bojati se može direktno na model ili na plošnu teksturu. Dio bojanja u ovom slučaju izvršen je u vanjskom softveru te je kasnije kao PNG slika ponovno importirana u Blender.



Slika 14: *Primjer rig-a s meta kostima*

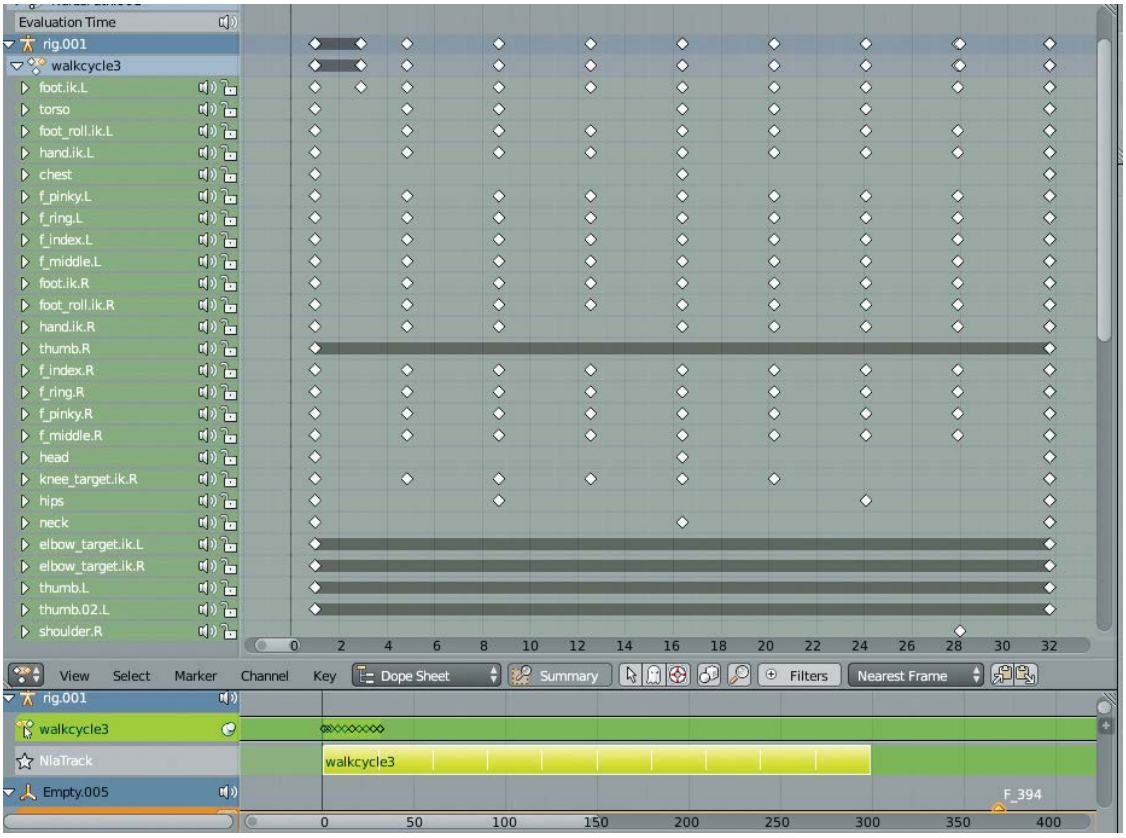


Slika 15: *Primjer rig-a nakon aplikacije rigify-a*



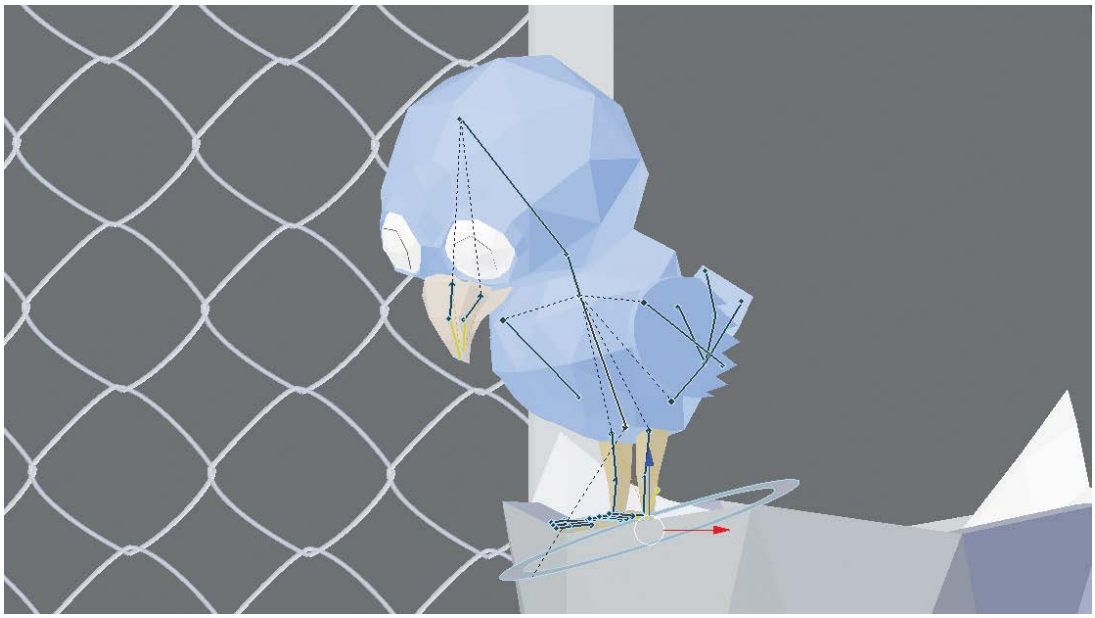
Slika 16: *Raspored težina; utjecaj rig-a na mesh*

Na slikama 14 i 15 mogu se razabrati 2 vrste *rig*-a. U biti se radi o jedno te istom *rig*-u. Prvi je gotovi i čitavi *rig* s *meta*-kostima i drugim *nus*-produktima *rigify*-a (koji nas trenutno ne interesiraju). Drugi *rig* je onaj vidljivi odnosno onaj kojeg koristimo za animiranje. Ovakav *rig* je vizualno čišći i jednostavniji te je samim time lakši za korištenje. Njega ne čine prave kosti već su to samo "poluge" koje pokreću stvarne kosti koje "ne vidimo". Neke od kosti na prvoj slici čovjeka s *rig*-om su naznačene drugačijom fluorescentnom bojom, to samo znači da je ta kost povezana s barem jednom ili više kosti te njenim pokretanjem se utječe i na te kosti. Slika 16 prikazuje raspored težina na tijelu, odnosno kojom jačinom će kosti *rig*-a utjecati na (natezati) *mesh* tj. model koji je *rig*-an. Crvena boja znači jači utjecaj dok plava znači da uopće nema nikakvog utjecaja. Dakle, što je boja toplija to je utjecaj jači, što je boja hladnija to je utjecaj slabiji.



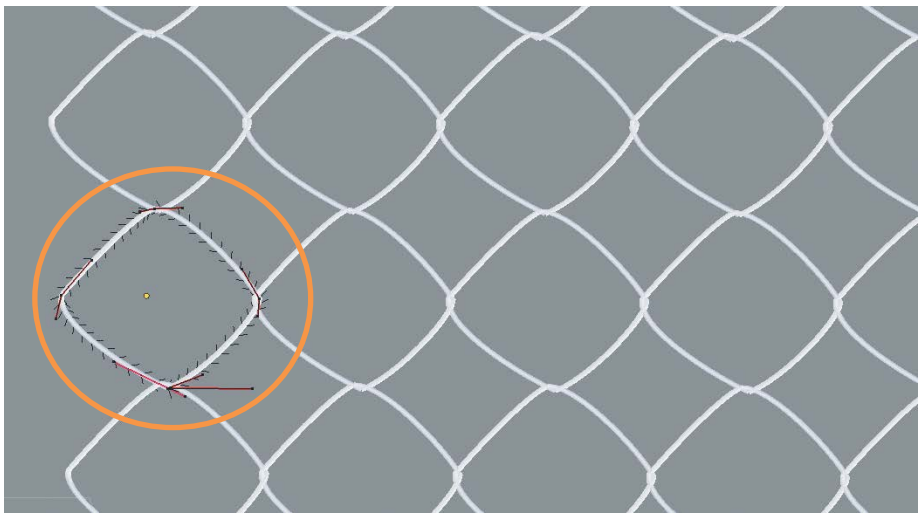
Slika 17: *Postavljanje keyframeova za ciklični hod*

Na slici 17 prikazani su ključni *keyframeovi* koje je potrebno namjestiti ručno da bi se napravio ljudski ciklični hod. U tabeli na lijevo nalaze se nazivi kostiju na koje ti *keyframeovi* utječu.



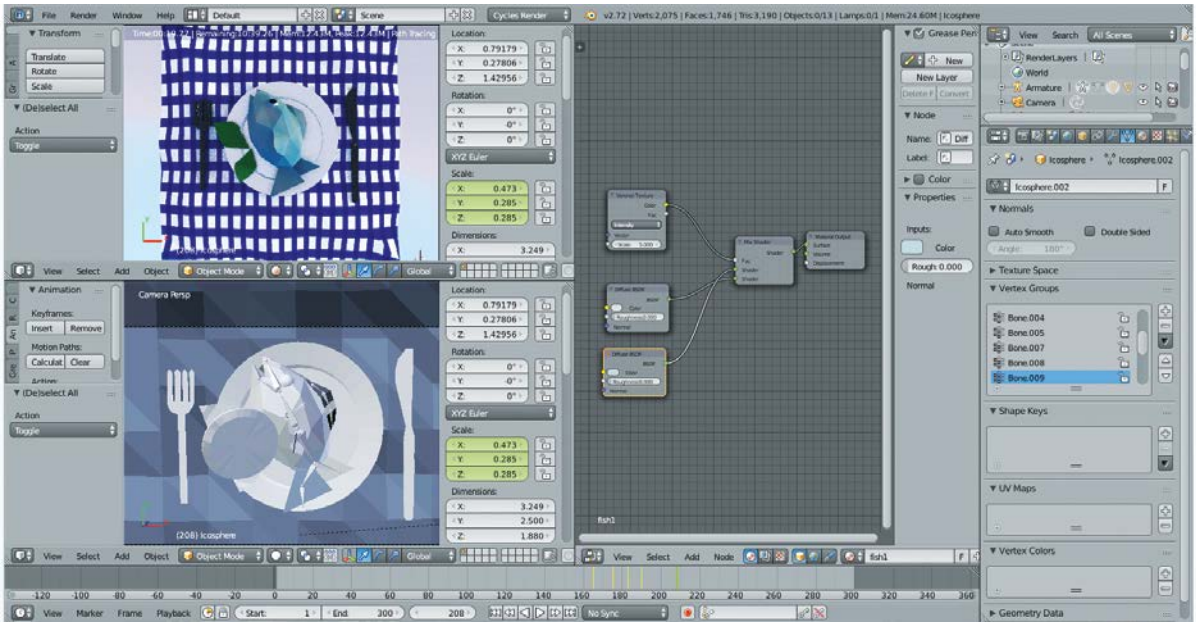
Slika 18: *Rig ptice*

Na slici 18 može se vidjeti kako izgleda konstrukcija *rig*-a za pticu te koje su razlike s obzirom na onaj ljudski. Korišten je drugačiji oblik kostiju isključivo zbog preglednosti i manje kompleksnosti. U ovom slučaju nije se primijenio *rigify* jer to nije bilo potrebno.



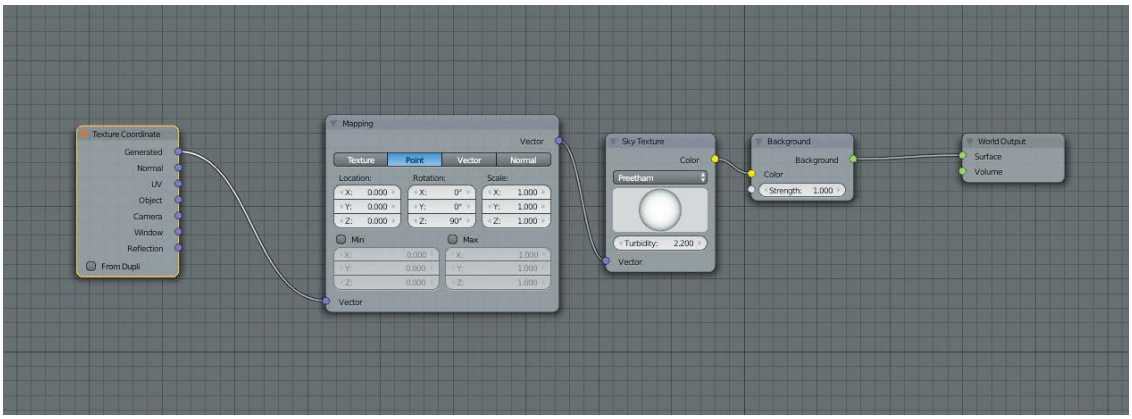
Slika 19: *Žičana ograda-dio scenografije*

Slika 19 prikazuje dio žičane ograde napravljene ponavljanjem jedno te istog oblika, odnosno koristio se *Array modifier*. Osnovni oblik moguće je jednostavno napraviti pomoću *Bazierovih krivulja* (naznačeno narančastom kružnicom), koje se kasnije mogu pretvoriti u *mesh*.



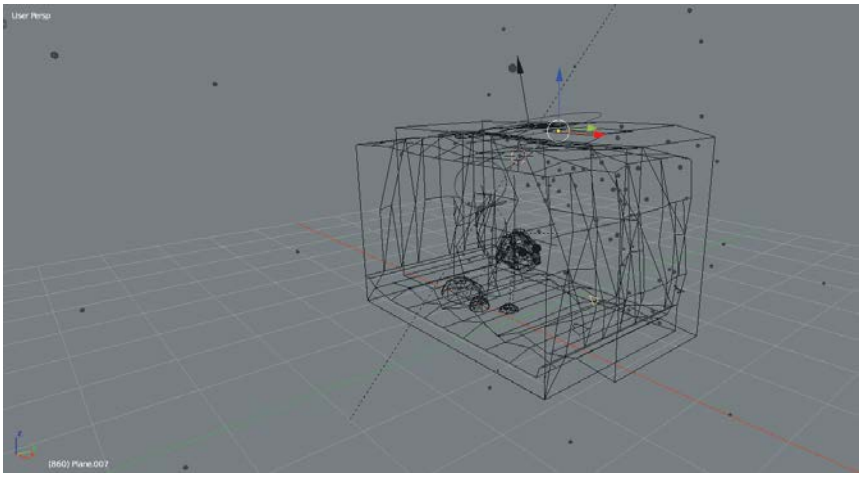
Slika 20: *Interface softvera prilikom node uređivanja*

Slika 20 prikazuje 3 prozora Blenderova sučelja, koji su modularni, što znači da ih se može dodavati i uklanjati te mijenjati po potrebi. Gornji lijevi prozor je render u *real timeu*, donji je u *object modu* i treći (desno) je prikaz *node* strukture za neki od materijala. Blender, isto tako, ima i opciju korištenja već *pred-postavljenih* sučelja koje je moguće izmjenjivati tokom rada.

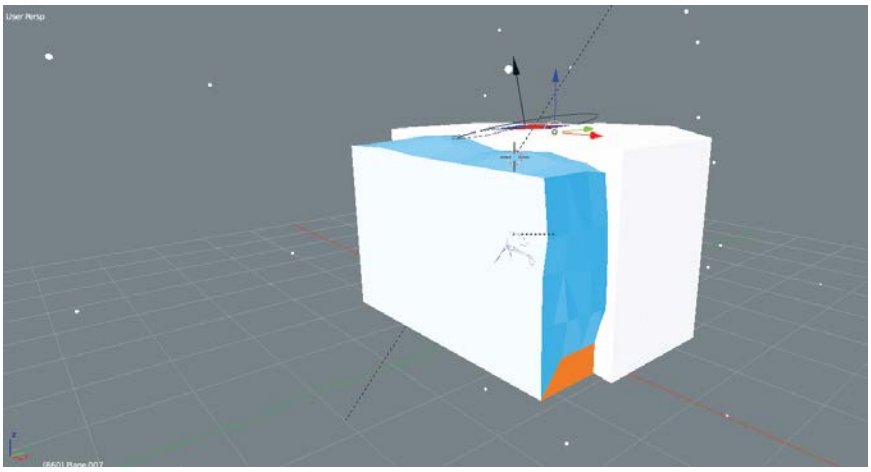


Slika 21: *Node strukture za generiranje teksture neba*

Na slici 21 prikazana je jedna jednostavna *node* struktura koja definira boju neba odnosno *background*-a. Nebo je jednako u svim scenama, jer je jedan od faktora koji utječu na način rasvjete u sceni. Pa zbog uniformnosti animacije nije mijenjano.



Slika 22: *Prikaz ćelije u wireframe prilikom čestičnog raspršivanja*



Slika 23: *Prikaz ćelije object modu prilikom čestičnog raspršivanja*

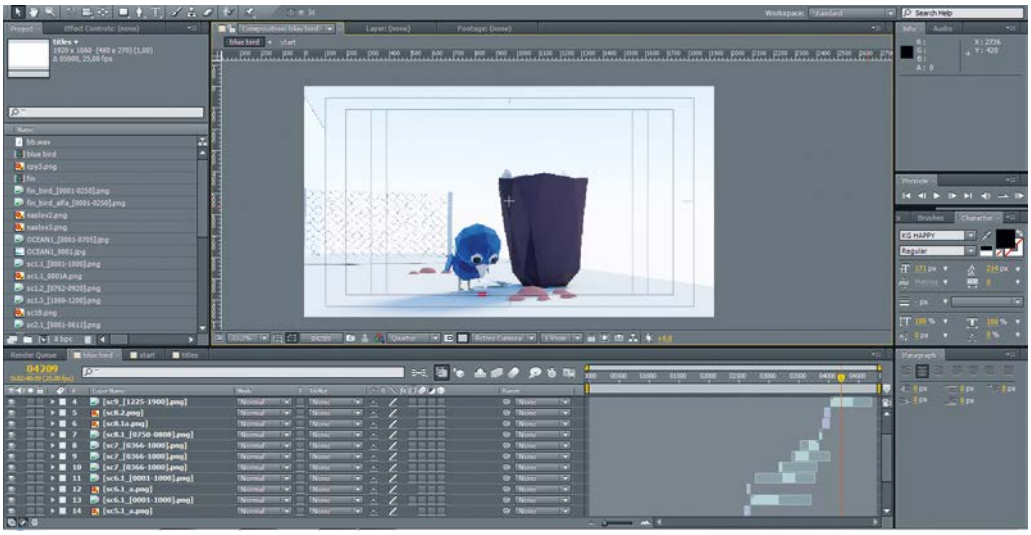
Slike 22 i 23 prikazuju iste elemente na sceni ali u 2 različita *modea*. Slika 22 prikazuje element u *object* (hrv. objektnom) *modeu* (s *material shading-om* (hrv. slobodan prijevod; sjenčanjem na bazi materijala)) a slika 23 element je u *wireframe* (hrv. žičanom) *modeu*. Slike 22 i 23 imaju za cilj prikazati funkciju čestica odnosno kako postaviti neke od fizika u 3D prostoru. Postavljen je prostor u kojem se čestice imaju pravo raspršivati. One se, naime, sudaraju s stranicama kvadra u kojem se nalaze te se odbijaju od njegove površine. To je postavljeno u postavkama za fiziku objekata i prostora. Njihov broj, brzinu, veličinu i način kretanja se postavlja u postavkama za čestice (eng. *Particle settings*).

Slika 24: *Kosa napravljena pomoću čestica*

Particles ili čestice znatno olakšavaju stvaranje količinski iznimno mnogo i po prirodi kaotičnih elemenata. Nakon što se odredi prostor na kojem ili u kojem će se postaviti čestice, njihova količina i ostali parametri koji određuju tip i strukturu čestica, program po zadanim parametrima izračunava čestice. Na slici 24 može se uočiti kosa koja je napravljena metodom čestica (eng. *hair particles*). Pomoću *weight paint*-a (hrv. slob. prijevod bojanje težine) ali i alata za modeliranje kose koje nudi Blender moguće je manipulirati takvim strukturama. Čestice nisu dio *mesh*-a ali postoji mogućnost njihovog pretvaranja u *mesh*. Čestice su iznimno težak element za procesiranje te se preporuča njihovo deaktiviranje prilikom modeliranja kako bi se sistem održao stabilnim.

Slika 25: *Postavljanje svijetla*

Slika 20 prikazuje WIP prilikom namještanja jačine i smjera svjetla, kao i korekcije boje i tona neba. Smjer iz kojeg dolazi svijetlo određuje smjer i način na koji će padati sjene.



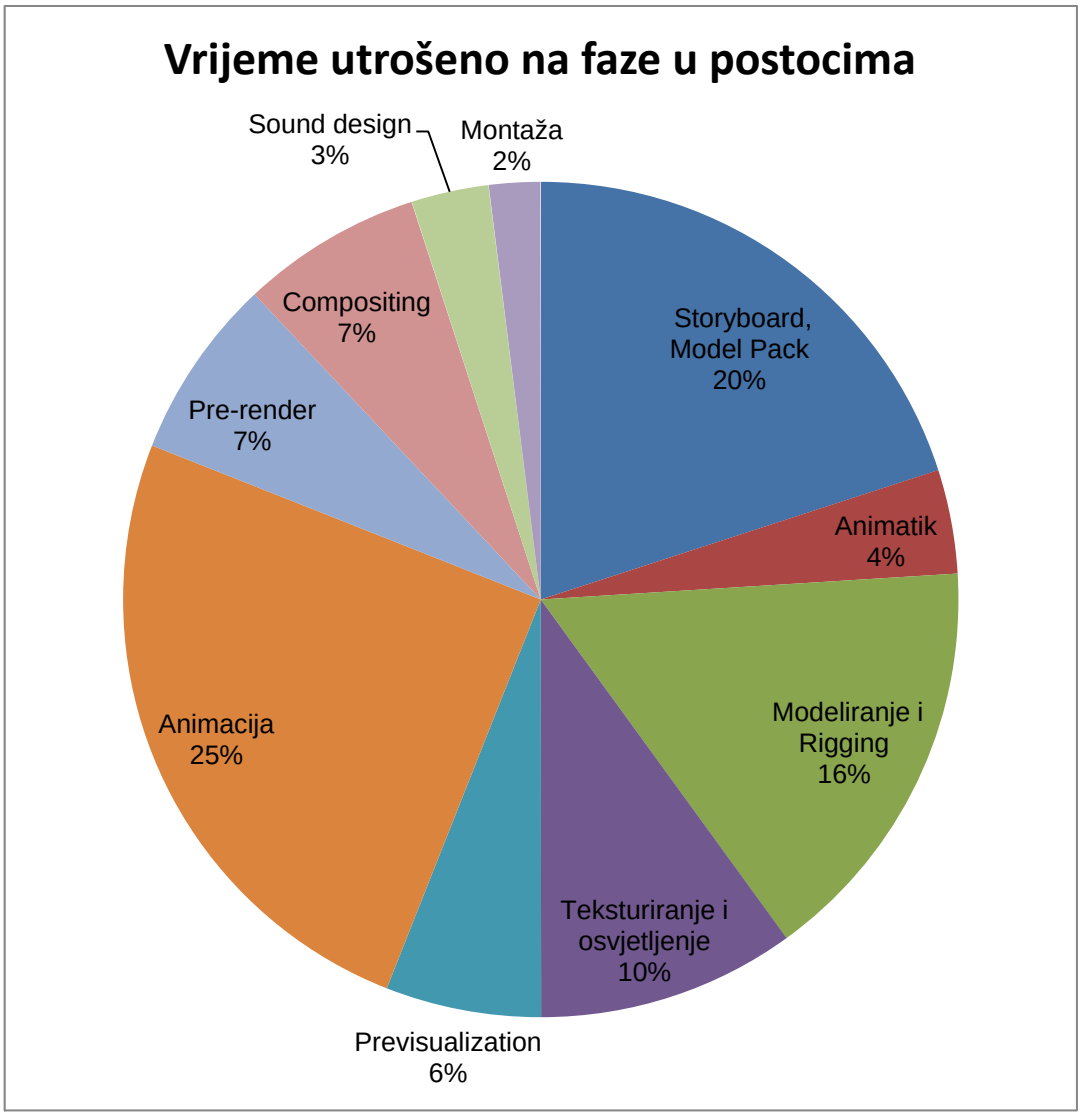
Slika 25: *Compositing i montaža u After Effects-u*

Slika 25 prikazuje snimku ekrana u procesu *compositinga* i montaže animacije u *Adobe After Effects-u.* Ovo je ujedno i zadnja faza te nakon nje projekt je spreman za eksportiranje.

8. Rezultati i diskusije

Za produkciju jedne ovakve animacije, razloženoj u poglavlju 7 i prezentiranoj u prilogu 1, u trajanju od otprilike 3 minute, osobi koja po prvi puta radi 3D animaciju potrebno je oko 3 mjeseca konstantnog rada, pod uvjetom da ima barem osnovno znanje o korištenom softveru. Na kraju, se može zaključiti da ovakav tip animacije, kratka i u *low-poly* tehnici, se može jako dobro i profesionalno, bez većih poteškoća i vremenski optimizirano, izvesti u softveru kao što je Blender. U ovom slučaju ova animacija može konkurirati sličnim animacijama napravljenim u *closed-source* softverima, što načinom izrade i sveukupnom kvalitetom rendera ali više od svega cijenom, zato što softver korišten za ovu animaciju je bio potpuno besplatan. Dakako, važno je imati na umu, da sam rad, trud i vrijeme uloženo kao i računalo koje je potrebno za podržavanje ovakvog softvera i rada u njemu, ipak, nisu besplatni. Također, iako se *compositing* mogao obaviti u Blenderu, pa čak i finalni eksport (izlaz), ipak to nije učinjeno, što zbog brzine što zbog praktičnosti drugih softvera. Tako da uz Blender poželjno je posjedovati još nekakav softver za montažu i/ili *compositing* (u ovom slučaju to je bio *Adobe After Effects*). Ono što je uistinu pomoglo pri stvaranju ove animacije, i nije za zanemariti, jest upravo *online* zajednica Blender korisnika. Postoje razne *web* stranice na kojim se uvijek bez većih poteškoća može pronaći rješenje za moguće probleme i nedoumice. Pri izradi ovog rada kao pomoć i reference služili su radovi i *rig*-ove drugih autora. Pa se stoga da zaključiti da je sam proces učenja i usavršavanja znatno brži nego što bi to bio slučaj s drugim sličnim softverima. Dakako, Blenderovo sučelje je nešto drugačije nego što je to uobičajeno. U početku je potrebno nešto više vremena kako bi se savladalo ali i kako bi se upamtili svi *hotkey*-evi, bez kojih je gotovo nemoguće raditi jer su iznimno praktični. Pozitivna stvar Blenderovog *interface-a* jest što je modularan i može ga se bez poteškoća prilagođavati osobnim potrebama, a to, je pri izradi animacije od velike važnosti. Sve u svemu ovaj program je više nego idealan za početnike i one koji tek ulaze u svijet 3D grafike. *Open source* animacija svakako ima budućnost i svoje

mjesto u CGI animaciji, te nikako ne bi trebala biti podcjenjivana. Prilikom izrade ove animacije nije bilo predviđenih elemenata, koje kasnije, nije bilo moguće tehnički izvesti u datom softveru. Jedina prepreka u tom smislu jesu bile tehničke mogućnosti računala. Stoga prije nego se započne s osmišljavanjem neke animacije, preporuča se da se bude sigurno da je tehnička oprema odgovarajuća. *Open source animacija* načinom izrade ali i vizualno sve se manje razlikuje od animacija nastalih u *closed-source* softverima, jedina velika razlika jest ta što se do nje došlo značajno jeftinije (i u određenom smislu i lakše). Korištenjem Blendera se na indirektan način pridonijelo pokretu *open sourcea* i širenju njegovog imena i ideja.



Graf 2: *Grafički prikaz utrošenog vrijeme po fazama za animaciju "Bule Bird"-a*

9. Zaključak

Sve do sada izvedene teorije o opisu i primjeni *open source* softvera, *free* softvera te pokreta koji ih zastupaju su zaključci autora, a oni su nastali na temelju tekstova i članaka koji su pušteni u opticaj na internetu od različitih fondacija i pobornika *open* i *free* softver pokreta. Međutim, ne podudaraju se potpuno s izvornim tekstovima. Pokušali su se deskriptivno i što objektivnije razjasniti termini i njihove međusobne korelacije. Pošto su tekstovi izdani pod raznim licencama cilj ovog rada je bio da se sažmu ideje i teorije proizašle iz različitih izvora.

Shvaćanje ideologije *open source* i *free softver* pokreta i razumijevanje njihovih popratnih licenci je neophodno ako se želi shvatiti zašto je dobro koristiti *open source* softver, koje se sve mogućnosti otvaraju i zašto bi se na koncu napravila 3D animacija u takvom softveru. U ovom radu objašnjeni su koraci potrebni da bi se uopće krenulo u kreiranje CGI animacije. Sama tehnika neće biti značajno različita ali sama činjenica da se ovdje radi o *open source* animaciji u nekim segmentima može biti prednost. Vrlo često informacija ima mnogo i dolaze iz različitih izvora, a kako je svijet CGI-a svijet koji konstantno evoluira, teško je pronaći ono što uistinu treba kako bi se moglo stručno dalje usavršavati. Animacija u *open sourceu* je tek u svojim začecima, međutim svakim novim tehnološkim napretkom i njene mogućnosti i prednosti su sve jače. Teško je povjerovati da će se u skoro vrijeme *open source* animacija postati jedna od *mainstream* tehnika, ali svakako će sve čvršće i bolje grabiti k tome, te u skoroj budućnosti čvrsto zauzeti svoje mjesto u produkciji CGI animiranih filmovima.

Pri izradi animacije u sklopu ovog rada najviše se vremena utrošilo na animiranje (25% od ukupnog vremena), potom na modeliranje i *rigging* (20%), 16% vremena utrošilo se na *storyboard* i *character design*, 10% na teksturiranje i reguliranje rasvjete, 7% na *compositing*, 7% i na *pre-render*, 6% na predvizualizaciju te 3% na *sound design* i 2% na montažu. Grafički prikaz grafa 2.

Literatura

1. GNU (2015), The GNU Operating System and the Free Software Movement, www.gnu.org [pristupljeno 15. kolovoza, 2015.]
2. TED (2015) TED: Ideas worth spreading, www.ted.com [pristupljeno 1. kolovoza, 2015.]
3. Blender Foundation, blender.org, Home of the Blender, www.blender.org/foundation [pristupljeno 30. lipnja, 2015.]
4. Benoît Saint-Moulin (2007), 3D softwares comparisons table, raspoloživo na: http://www.tdt3d.com/articles_viewer.php?art_id=99 [pristupljeno 30. kolovoza, 2015.]
5. Vikram Pandya (2013), Styles of Animation, raspoloživo na: http://www.animationsupplement.com/index.php/articles/39-styles-of-animation [pristupljeno 30. kolovoza, 2015.]
6. Netta Canfi (2013), How Does Computer Animation Work?, raspoloživo na: http://www.the-flying-animator.com/how-does-computer-animation-work.html [pristupljeno 30. kolovoza, 2015.]
7. Computer Graphics Wiki – Wikia, http://graphics.wikia.com/wiki/Rendering [pristupljeno 25. lipnja, 2015.]
8. Mark Giambruno (2013), 3D Modeling Basics, raspoloživo na: http://www.peachpit.com/articles/article.aspx?p=30594 [pristupljeno 25. lipnja, 2015.]
9. Catherine Winder, Zahra Dowlatabadi (2001), Producing Animation, pristupljeno preko: www.books.google.it [30. srpnja, 2015.]
10. GNU (2015), Free Software Foundation's Licensing and Compliance Lab, Various Licenses and Comments about Them, raspoloživo na: http://www.gnu.org/licenses/license-list.en.html [pristupljeno 15. kolovoza, 2015.]
11. HAA (2015), Hrvatska akreditacijska agencija (HAA), www.haa.hr [pristupljeno 15. kolovoza, 2015.]
12. David Bushell (2011), Understanding Copyright And Licenses, raspoloživo na: http://www.smashingmagazine.com/2011/06/understanding-copyright-and-licenses/ [pristupljeno 15. kolovoza, 2015.]
13. Filmski leksikon (2003) - Leksikografski zavod Miroslav Krleža, http://film.lzmk.hr [pristupljeno 20. kolovoza, 2015.]
14. Austin Mayden (2013), CG File Formats You Need to Know - Understanding OBJ, FBX, Alembic and More, raspoloživo na: http://blog.digitaltutors.com/cg-file-formats-you-need-to-know-understanding-obj-fbx-alembic-and-more/ [pristupljeno 5. rujna, 2015]
15. Jon Burgerman (2015), 20 top character design tips, raspoloživo na: http://www.creativebloq.com/character-design/tips-5132643, [pristupljeno 5. kolovoza, 2015.]
16. Mike S. Fowler (2003), Animation Layout: Layout Support Material, raspoloživo na: http://www.awn.com/animationworld/animation-layout-layout-support-material [pristupljeno 25. lipnja, 2015.]

17. Sareesh Sudhakaran (2013), What is Compositing?, raspoloživo na: http://wolfcrow.com/blog/what-is-compositing/ [pristupljeno 25. kolovoza, 2015.]
18. Bill Davis (2001), Computer editing: Basic Compositing, raspoloživo na: http://www.videomaker.com/article/8307-computer-editing-basic-compositing [pristupljeno 16. kolovoza, 2015.]
19. G. Wiesen, Uredila: Heather Bailey (2015), What Is 3D Compositing?, rasploživo na: http://www.wisegeek.com/what-is-3d-compositing.htm [pristupljeno 16. kolovoza, 2015.]
20. The Foundry: About digital compositing (2015), What is digital compositing?, raspoloživo na: https://www.thefoundry.co.uk/products/nuke/about-digital-compositing/ [pristupljeno 18. kolovoza, 2015.]
21. Mindy McAdams (2011), Journalists' Toolkit, The best video export options, raspoloživo na: http://www.jtoolkit.com/wp/2011/03/the-best-video-export-options/ [pristupljeno 30. kolovoza, 2015.]
22. Mark Masters (2014), What Makes a Great Character Rig? 12 Things Animator-Friendly Rigs Should Have, raspoloživo na: http://blog.digitaltutors.com/makes-great-character-rig-12-things-animator-friendly-rigs/ [pristupljeno 30. kolovoza, 2015.]
23. Mark Masters (2015), How to Create Your First Character Rig in Blender: Part 1 - Setting up the Armature, raspoloživo na: http://blog.digitaltutors.com/how-to-create-your-first-character-rig-in-blender-part-1/ [pristupljeno 1. rujna, 2015.]
24. Richard Stallman (2015), Why Open Source misses the point of Free Software, raspoloživo na: http://www.gnu.org/philosophy/open-source-misses-the-point.en.html [pristupljeno 25. srpnja, 2015.]
25. Morr Meroz (2014), A step by step guide to becoming an animator; Animation for beginners, pristupljeno preko: http://www.bloopanimation.com/animation-for-beginners-book [30. srpnja, 2015.]
26. CC (2015) Creative Commons, non profit organization, https://creativecommons.org/licenses/ [pristupljeno 15. kolovoza, 2015.]
27. wikiHow (2015)- How to do anything, Create a storyboard, http://www.wikihow.com/Create-a-Storyboard [pristupljeno 15. kolovoza, 2015.]
28. Lance Flavell (2010), Beginning Blender: Open Source 3D Modeling, Animation, and Game Design, www.books.google.it [30. srpnja, 2015.]
29. Halligan, F., (2013), The art of storyboards, Visualising the action of the world's greatest films, East Sussex :210 High street, Lewes, izdavač; The ILEX press Limited
30. Giuli, D.M. (2014), Inquadrature e regia; dallo storyboard alle riprese di un film, izdavač: Gremese New Books s.r.l. Roma
31. Patria Dobbins (2012), 3D rendering in computer graphics, 4735/22 Prakashdeep Bldg, Ansari Road, Darya Ganj, Delhi-110002, Izdavač: White Word Publications

32. Egon Rath (2015), Basic 3D Math: Vectors, Part 1: Vectors, raspoloživo na: http://www.matrix44.net/cms/notes/opengl-3d-graphics/basic-3d-math-vectors [pristupljeno 25. srpnja, 2015.]
33. Hollasch E., (2015), Steve Hollasch, Chapter 4, Wireframe Display of Four Dimensional Objects, raspoloživo na: http://steve.hollasch.net/thesis/chapter4.html [pristupljeno 25. kolovoza, 2015.]
34. Jeff Jonaitis (2000-2004), Basic Modeling Theory and Technique, raspoloživo na: http://www.jjonaitis.com/tuto/tuto1.htm [pristupljeno 1. lipnja, 2015.]
35. Rhinoceros (2015), What are NURBS?, raspoloživo na: https://www.rhino3d.com/nurbs [pristupljeno 15. kolovoza, 2015.]
36. Techopedia (2010-2015), Software Licensing, raspoloživo na: https://www.techopedia.com/definition/2558/software-licensing [pristupljeno 28. kolovoza, 2015.]
37. Alan Rankin (2015), What Is the Difference between 3D and 2D?, raspoloživo na: http://www.wisegeek.com/what-is-the-difference-between-3d-and-2d.htm [pristupljeno 5. rujna, 2015.]
38. Open source initiative (2015), raspoloživo na: http://opensource.org/osd [pristupljeno 25. Srpnja, 2015]
39. Tom Fronczak (2011), 10 Types of 3D Graphics Software Worth Knowing, raspoloživo na: http://www.animationcareerreview.com/articles/10-types-3d-graphics-software-worth-knowing [pristupljeno 30. Kolovoza, 2015]
40. MatterHackers (2015), Finding the right 3D modeling software for you, raspoloživo na: https://www.matterhackers.com/articles/finding-the-right-3d-modeling-software-for-you [30. Kolovoza, 2015]
41. Charles Leadbeater (2015), TED- ideas worth spreading, raspoloživo na: https://www.ted.com/speakers/charles_leadbeater [pristupljeno 15. kolovoza, 2015]

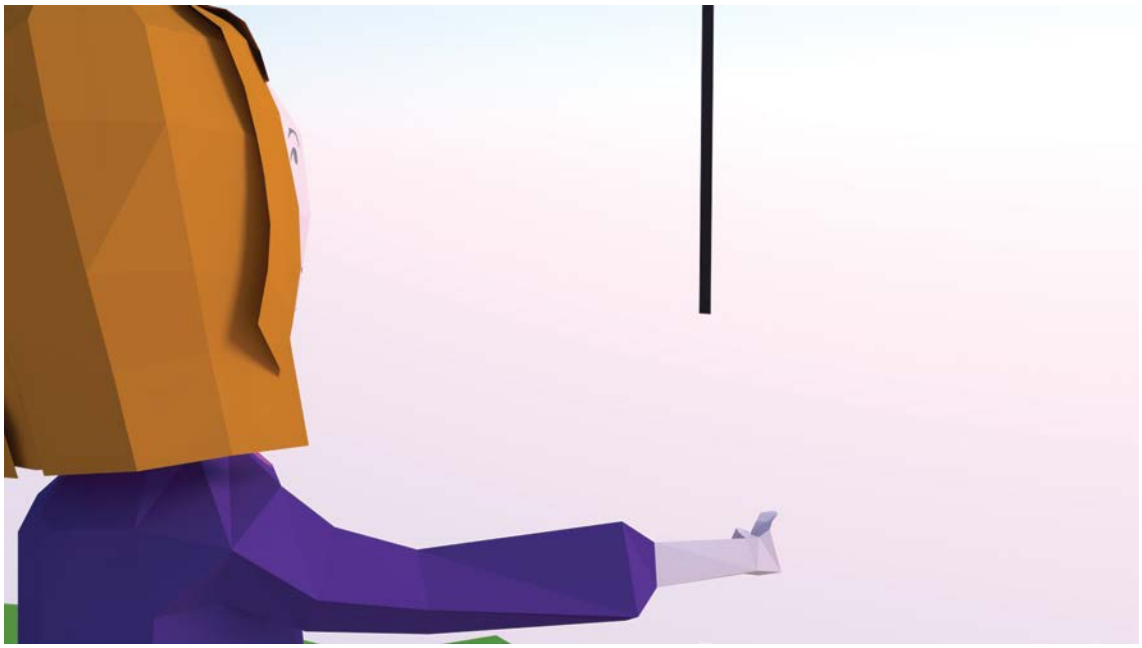
Prilog 1

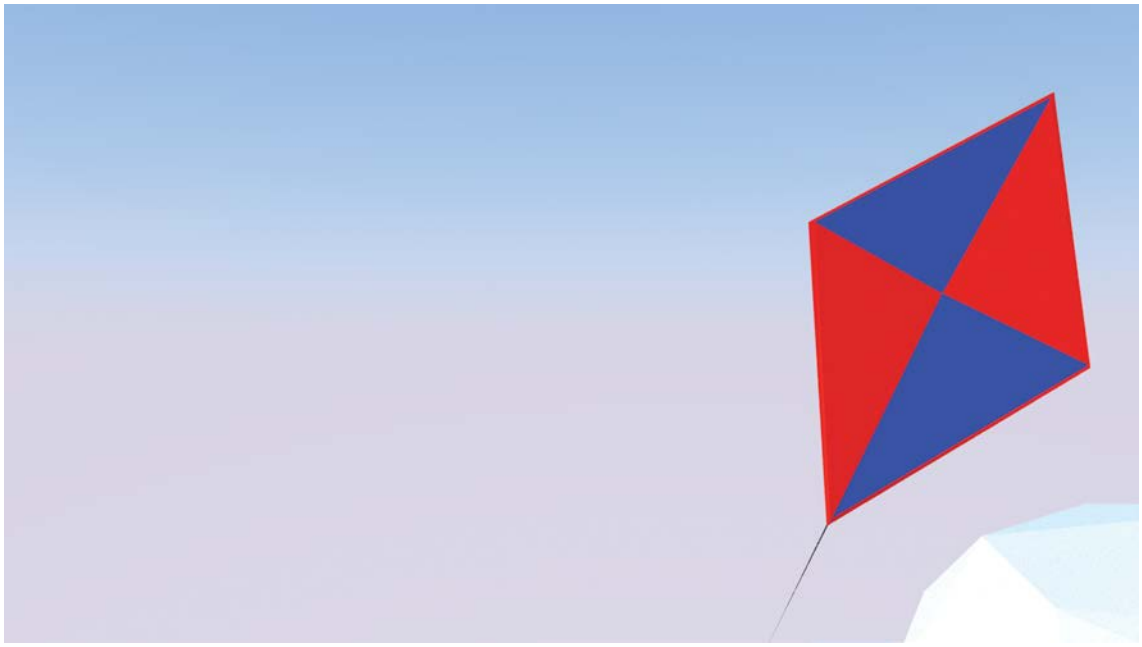
U nastavku slijedi kratki *storyboard* napravljen od CGI slika koje su izvučeni *pre-renderi* iz animacije koja je napravljena u 3D *open source* softveru Blender za eksperimentalne svrhe ovoga rada. Naziv ove kratke animacije je *“Blue Bird”* ili na hrvatskom jeziku *“Plava ptica”.*

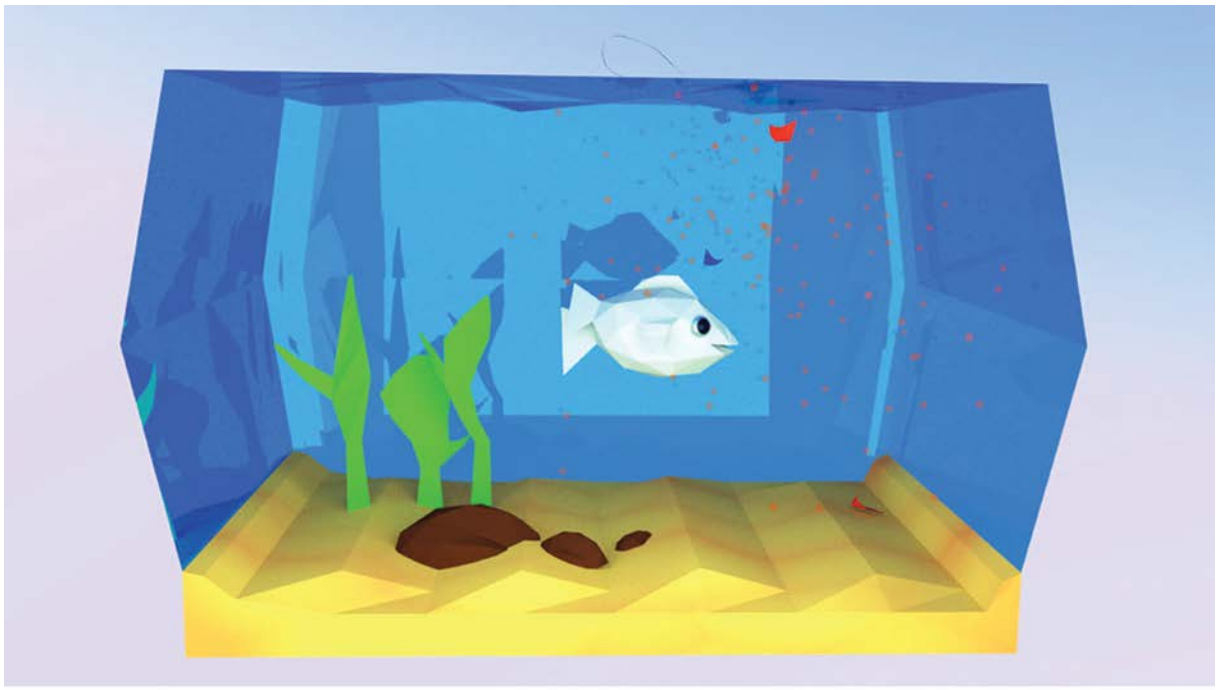
Kratki opis tematike animacije:

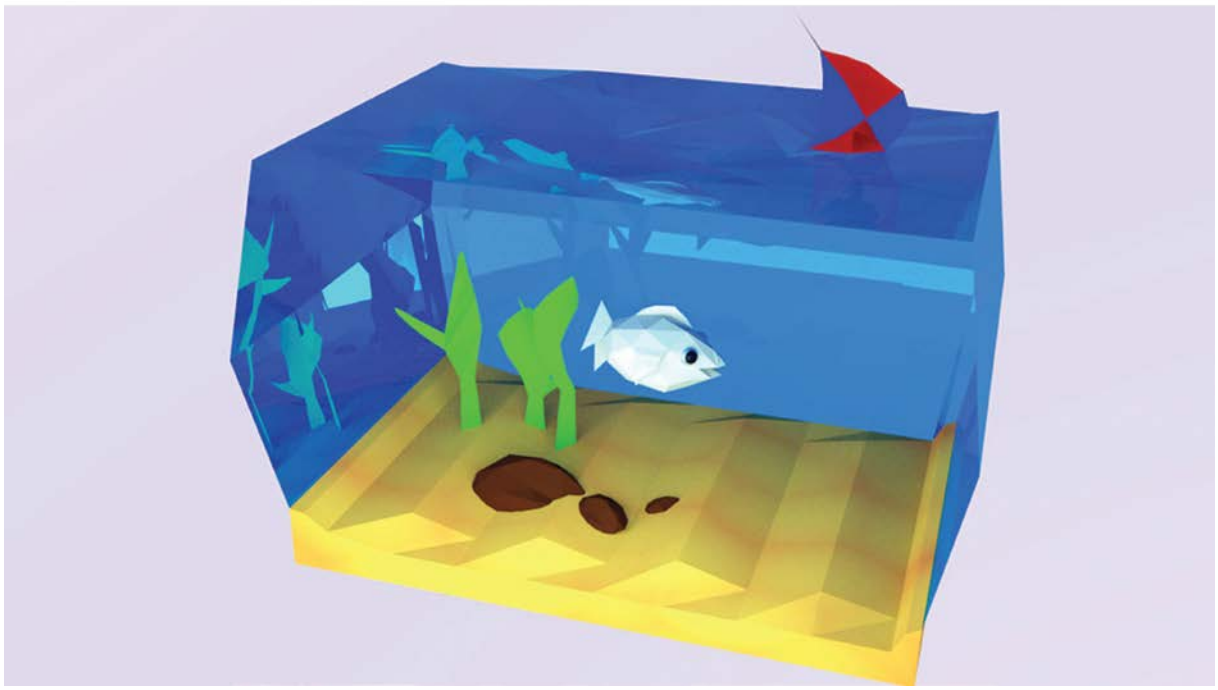
“*Ova kratka animacija pokušava u nekoliko minuta objasniti neke od mogućih posljedice plastike na okoliš, na život ljudi ali i živote drugih živih bića koji zajedno dijele ovaj planet. Govori o tome kako i najmanje zanemarivanje problema plastičnog otpada može imati svoje posljedice. Nečije smeće može postati nečija hrana. Scena otvaranja započinje s jednim poprilično su-realnim krajolikom i hladnom atmosferom. Djevojčica se igra i istodobno uživa u prirodi koja se čini koliko lijepom i opuštajućom toliko i umjetnom i plastičnom, baš kao da podsjeća da je ljepota prirode uvjetna i da bi mogla nestati u trenu, ako ju se zanemaruje. Djevojčica djeluje sanjivo, odsutno, nevino i potpuno je nesvjesna situacije koja će tek uslijediti. Vrlo brzo priča prelazi u nešto ozbiljnije okruženje. Sa svakom novom scenom animacija postaje još ozbiljnija i realističnija kako bi se istakla ozbiljnost situacije, te se polako raspliće prava problematika priče. Ciklus života uspoređen je sa ciklusom tj. kruženjem plastike u okolišu. Animacija je napravljena u low-poly tehnici zbog njene vizualne jednostavnosti i moći da se jednostavno prikažu kompleksne ideje. Time je poruka direktnije i s manje distrakcije prenesena ekspresivnom jednostavnošću geometrijskih oblika, boje i pokreta. Poruka o ekološkoj održivosti, prirodnom kruženju i o gradnji boljeg i samo-svjesnijeg društva, može se na neki način paragonirati s idejom dijeljena, širenja i gradnje sustava zajedno u kolaboraciji koju zastupa pokret open sourcea, pa tako i sam Blender.”*


Blue Bird





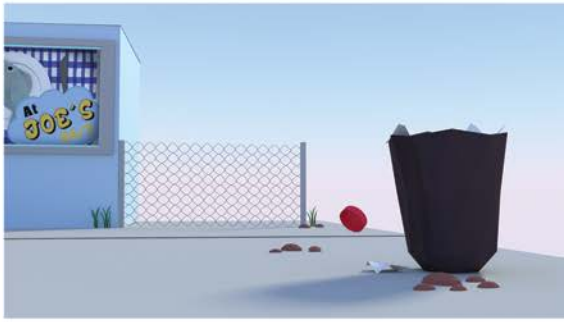





FRESH
FISH
At
JOE'S
24/7


At
JOE'S
24/7


At
JOE'S
24/7


At
JOE'S
24/7


JOE'S
24/7


JOE'S