SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1914

ZAVRŠNI RAD IZ PREDMETA INTERAKTIVNA RAČUNALNA GRAFIKA

Postupci animacije ljudskih likova

Petar Mrazović

Zagreb, Lipanj 2011.

Sadržaj

1. Uvod

Razvojem tehničkih znanosti, pa time i računarstva sve češće se susrećemo s vizualnim efektima koji nam oduzimaju dah. Prirodno se pitati na koji način su ostvarene te animacije, specijalni efekti i virtualna stvarnost. Sve su to produkti računalne grafike i animacije, područja vizualne računarske znanosti koje se razvijaju nezamislivom brzinom i nalaze široku primjenu u raznim područjima od znanosti, inženjerstva i medicine do vojnih primjena ili umjetnosti.

Slika 1.1. Primjeri primjene računalne grafike i animacije

Računalna grafika je relativno novo područje računarske znanosti. Počinje se razvijati 60-tih godina prošlog stoljeća s ciljem stvaranja slika ili uklapanja i mijenjanja slikovnih i prostornih podataka koji su uzeti iz stvarnosti pomoću računala. Ostvariti računalnu predodžbu vizualne stvarnosti u ono vrijeme se činilo vrlo teškim i neizvedivim zadatkom, no današnji filmovi, video igre ili razne realizacije proširene stvarnosti nam ukazuju na brzinu kojom računalna grafika napreduje.

Računalna animacija, veliko i važno područje računalne grafike, započinje s razvojem u isto vrijeme kad i računalna grafika. Sama animacija kao postupak stvaranja iluzije pokreta poznata je puno prije pojave modernih računala, ali s njima je doživjela procvat. Računalna animacija našla je široku primjenu u znanosti, inženjerstvu i medicini, ali ponajviše u „industriji“ zabave i umjetnosti (filmovi i video igre). Ovako široka primjena je posljedica činjenice da svaku veličinu koja se može mijenjati u vremenu možemo animirati.

Čest zadatak animacije je prikazati živa bića u pokretu, a posebno ljudske likove. Upravo je to i središnja tema ovog završnog rada. Ovaj rad daje uvid u važna područja računarske znanosti - računalnu grafiku i animaciju, pregled razvoja animacije u širem smislu, te opis i primjenu postupaka animiranja ljudskog tijela. Rad također prati proces programskog ostvarenja tehnološke demonstracije animiranog ljudskog tijela koja je rezultat završnog zadatka.

2.1. Povijest animacije

Riječ animacija dolazi od latinske riječi *anima* koja predstavlja živuću silu u svakom biću. Dakle, animirati znači doslovno oživjeti, tj. dati život. Animaciju možemo najpreciznije definirati kao brzi prikaz slika dvodimenzionalnih ili trodimenzionalnih objekata s ciljem dobivanja iluzije pokreta. Zapravo se radi o optičkoj iluziji koja nastaje zahvaljujući medicinskom fenomenu tromosti oka. Utisak slike koju vidimo ostaje u mrežnici oka približno 1/25 sekunde.

Prvi pokušaji animacije datiraju još iz starijeg kamenog doba, kada su ljudi u pećinama prikazivali životinje s različitim položajima nogu pokušavajući ostvariti dojam kretanja. No, intenzivniji razvoj animacije započinje tek početkom 19. stoljeća kada nastaju prve igračke i slikovne knjige koje izazivaju iluziju pokreta. Razvojem filmske industrije bilježi se i veliki napredak animacije. Postaje prisutna u svim segmentima umjetnosti i zabave. Pojavom računala, a time i računalne animacije, pronalazi se primjena u širim područjima kao što su medicina, inženjerstvo ili vojne primjene. Shema 1. na stranici 8 slikovito prikazuje razvoj animacije u razdoblju od 1800. godine do danas.

Slika 2.1. Thaumatrop (lijevo) i fenakistoskop (desno)

Početkom 19. stoljeća nastaju prvi jednostavni uređaji i igračke koje su prikazivale slike u pokretu. Njihovom pojavom počinje intenzivniji razvoj animacije. 1824. godine britanski fizičar John Ayrton Paris izumio je *thaumatrop*, napravu koja se često smatra pretečom

filma. Riječ je o okruglom kartonu razapetom između dvije uzice. Najčešće je na jednoj strani bila nacrtana krletka, a na poleđini ptica, na mjestu koje odgovara sredini krletke. Kada bi se karton zavrtio dovoljno brzo gledatelj bi dobio dojam kao da se ptica nalazi u krletci.

6 godina kasnije pojavila se i druga naprava koja je brzim prikazom slika izazivala iluziju pokreta - *fenakistoskop*. Fenakistoskop je okrugli karton pričvršćen na dršku u središtu kartona. Na rubovima kartona nalazile su se slike nekog pokreta u određenom trenutku. Kada bi se karton zavrtio oko svog središta, slike na rubu bi stvorile dojam pokreta. 60-tih godina 19. stoljeća počinju se pojavljivati i prve slikovne knjižice.

1877. godine Charles-Émile Reynaud izumio je *praksinoskop*, uređaj koji se sastojao od manjeg valjka s ogledalima, okruženog crtežima s unutarnje strane većeg valjka. Okretanjem slika u ogledalu bi se pojavio fiksni odraz koji je reflektirao pokretnu sliku. Praksinoskop je vrlo brzo krenuo u komercijalnu prodaju, a Charles-Émile Reynaud je nastavio njegov razvoj. 1888. godine nastaje *Theatre Optique*, veliki praksinoskop namijenjen javnom prikazivanju animacija. 4 godine kasnije dogodila se i slavna premijera u Parizu gdje su prikazani filmovi u trajanju od 12 do 15 minuta.

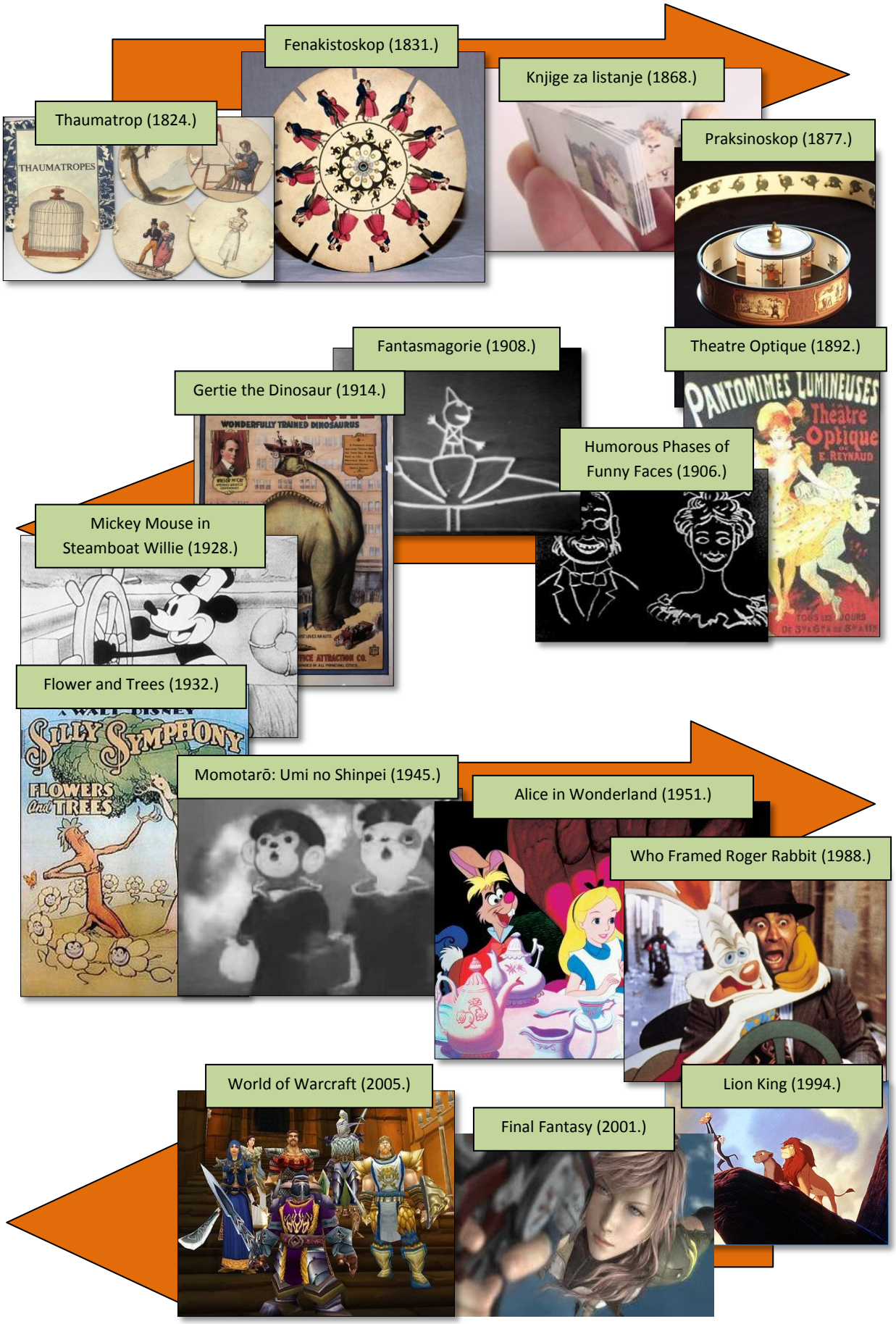
Slika 3.2. Émile Cohl i John S. Blackton

Prvi pravi animirani film pojavio se 1906. godine. Riječ je o filmu *Humorous Phases of Funny Faces* američkog filmaša Jamesa Stuarta Blacktona. Blackton je prvi pravi animator koji predstavio koncepte tehnike stop animacije i animacije ručnih crteža (eng. *hand-*

drawn animation). 1908. godine francuski umjetnik Émile Cohl napravio je animirani film *Fantasmagorie* koji je prikazivao jednostavne ljudske oblike u interakciji s raznim predmetima (boce, cvijeće i sl.). Spomenuta dva animirana film ostvarila su velik uspjeh i postavila temelje animiranog filma. Slijedeći uspjeh Blacktona i Cohla, američki strip autor Winsor McCay 1911. godine objavljuje, nakon 4 godine rada i oko 4 000 crteža koje je sam izveo, kratki animirani film *Little Nemo in Slumberland*. Slijedi ga njegov najpopularniji animirani film *Gertie the Dinosaur* 1914. godine. Kreativni vrhunac McCayeve karijere predstavlja animirani film *The Sinking of the Lusitania* (1918.), s oko 25 000 crteža. Ovim kratkim animiranima filmovi započinje razvoj danas vrlo velike industrije animiranog filma.

Slika 4.3. Winsor McCay: *Gertie the Dinosaur, Little Nemo in Slumberland*

Pojavom računala moderniziraju se postupci animacije. Računalna animacija nalazi sve širu primjenu, i to ne samo u filmskoj industriji. Vrlo brzo razvijaju se alati za jednostavnu animaciju koji su dostupni svima. Tehnologije animacije i dalje intenzivno napreduju čime se razvija snažna filmska i industrija video igara. U nastavku je shemom 1. prikazan razvoj animacije u razdoblju od 1800. godine do danas.



Shema 1. Razvoj animacije

2.2. Animacijske tehnike

2.2.1. Tradicionalna animacija

Slika 5.4. Kamera za snimanje crteža na celuloidnoj foliji

Tradicionalna animacija je najčešće korišteni postupak animacije prošlog stoljeća. Tradicionalno animirani filmovi sastoje se od niza fotografija ili ručno crtanih sličica, pri čemu se svaka slika vrlo malo razlikuje od prethodne da bi se njihovim brzim prikazivanjem ostvarila iluzija pokreta. Sličice tradicionalno animiranog filma nastaju kao skice za crtačkim pultom, a potom se prenose na tzv. celuloidne folije i snimaju specijalnim kamerama (slika 2.5.). Iz tog razloga tradicionalna animacija je poznata i pod nazivom *celuloidna animacija*.

Postupci tradicionalne animacije danas se u praksi rijetko koriste. Pojavom računala oni se moderniziraju, pa se danas najčešće crteži skeniraju ili crtaju izravno na računalnim sustavima, te se animiraju pomoću razvijenih programskih alata. Najpoznatiji primjer dugometražnog tradicionalno animiranog filma je *Pinocchio* produkcije Walt Disney iz 1940. godine. Krajem 20. stoljeća postupci tradicionalne animacije potpomognuti su računalnim sustavima, pa tako velik uspjeh bilježe filmovi *Kralj Lavova* (SAD, 1994.) i *Avanture male Chihiro* (Japan, 2001.), dobitnik oskara za najbolji animirani film.

Razlikujemo i nekoliko stilova tradicionalne animacije: potpuna animacija, djelomična animacija i rotoskopija.

Potpuna animacija (eng. *full animation*) je pojam koji se odnosi na produkciju visokokvalitetnih tradicionalno animiranih filmova koje koriste detaljne crteže i uvjerljivo prikazuju kretanja. Primjeri potpune animacije su spomenuti *Kralj lavova*, te filmovi *Ljepotica i zvijer* (SAD, 1991.) i *Aladdin* (SAD, 1992.).

Slika 6.5. Potpuna animacija (*Aladdin, Ljepotica i zvijer, Kralj lavova*)

Djelomična animacija (eng. *limited animation*) podrazumijeva uporabu manje detaljnih crteža, čime su i ostvareni pokreti manje uvjerljivi. Djelomična animacija je često korištena kao metoda stiliziranog umjetničkog izričaja, pa tako velik broj *anime* filmova Japanske produkcije pripadaju upravo ovom stilu. Najznačajnija ostvarenja djelomične animacije su djela Hayaoa Miyazakija, najvećeg japanskog redatelja animiranih filmova: *Princeza Mononoko* (1997.), *Avanture male Chihiro*, *Pokretni dvorac* (2004.) i drugi.

Slika 7.6. Djelomična animacija (*Avanture male Chihiro, Princeza Mononoko*)

Rotoskopija je tehnika koju je patentirao Max Fleischer, poznati američki animator (*Betty Boop*, *Popeye*), 1917. godine. Radi se o animaciji koja je vođena pokretima prethodno snimljenog stvarnog objekta.

2.2.2. Stop animacija

Stop animacija (eng. *stop-motion animation*) je tehnika kojom se iluzija pokreta ostvaruje fizičkim manipuliranjem stvarnih objekata te njihovim fotografiranjem. Razlikujemo više podvrsta stop animacije, obično po vrsti materijala koji se koristi.

Glinena animacija (eng. *clay animation, claymation*) koristi glinu, plastelin ili neki drugi sličan prilagodljiv materijal kako bi se ostvarila iluzija pokreta. Korištene figure najčešće

imaju žičani kostur kojim se olakšava pozicioniranje. Poznatiji primjeri glinene animacije su animirani filmovi *Wallace i Gromit* (Velika Britanija, 1989.), te *Pobuna u kokošinjcu* (SAD, 2000.).

Slika 8.7. Glinena animacija (*Wallace i Gromit, Pobuna u kokošinjcu*)

Animacija izrezivanjem (eng. *cutout animation*) koristi dvodimenzionalne komade materijala poput papira ili tkanine. Najpoznatiji primjeri ovakve animacije su animirane scene iz poznate britanske humoristične serije *Leteći cirkus Monty Pythona*. Tvorac ovih animacija je poznati britanski glumac, redatelj, scenarist i animator Terry Gilliam.

Podvrsta animacije izrezivanjem je **siluetna animacija** (eng. *silhouette animation*) gdje se likovi dodatno osvjetljavaju pozadinskim osvjetljenjem, te su vidljivi samo kao siluete.

Lutkarska animacija (eng. *puppet animation*) uključuje lutkarske figure koje se stavljaju u interakciju s konstruiranim okolišem. Lutke najčešće imaju žičani kostur radi lakšeg pozicioniranja. Najpoznatiji lutkama animirani filmovi su djela američkog redatelja Tima Burtona: *Božićna pustolovina* (1993.) i *Mrtva nevjesta* (2005.)

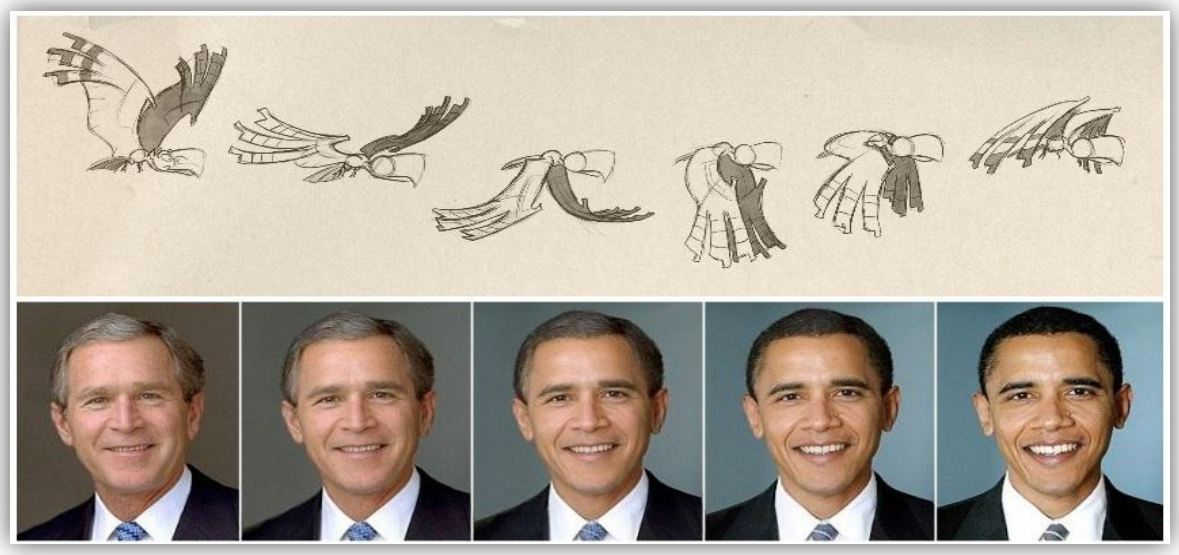
Slika 9.8. Lutkarska animacija (*Mrtva nevjesta, Božićna pustolovina*)

2.2.3. Računalna animacija

Računalna animacija je u osnovi digitalni nasljednik tradicionalne stop animacije, s 3D modelima ili digitalnim 2D ilustracijama. Riječ je modernoj tehnici stvaranja iluzije pokreta pomoću računalne grafike. Računalnu animaciju možemo podijeliti na računalom potpomognutu animaciju i računalno generiranu animaciju. **Računalom potpomognuta animacija** obuhvaća postupke prije opisane tradicionalne animacije koji su modernizirani uporabom računala. Ručno crtani objekti obrađuju se korištenjem raznih programskih alata te se računalno animiraju. **Računalno generirana animacija** ne obuhvaća postupke ručne izrade crtanih objekata koji se animiraju, već se 2D ili 3D objekti izrađuju na računalima gdje se i animiraju.

2.2.3.1. 2D računalna animacija

Dvodimenzionalni slikovni objekti su stvoreni i/ili uređeni na računalu koristeći 2D bitmap ili vektorsku grafiku. Najčešće tehnike dvodimenzionalne računalne grafike su automatizirane računalne verzije tradicionalnih animacijskih postupaka *tweeninga*, *morphinga*, *onion skinninga*, interpolirane rotoskopije i drugih. **Tweening** je postupak stvaranja i dodavanja međusličica između dviju ključnih sličica kako bi se dobila što „glađa“ animacija. **Morphing** je postupak kojim se prijelaz iz jedne sličice u drugu ostvaruje njenim izobličavanjem. **Onion skinning** metodom stapa se više bliskih slika u jednu.



Slika 10.9. *Tweening* i *morphing*

2.2.3.2. 3D računalna animacija

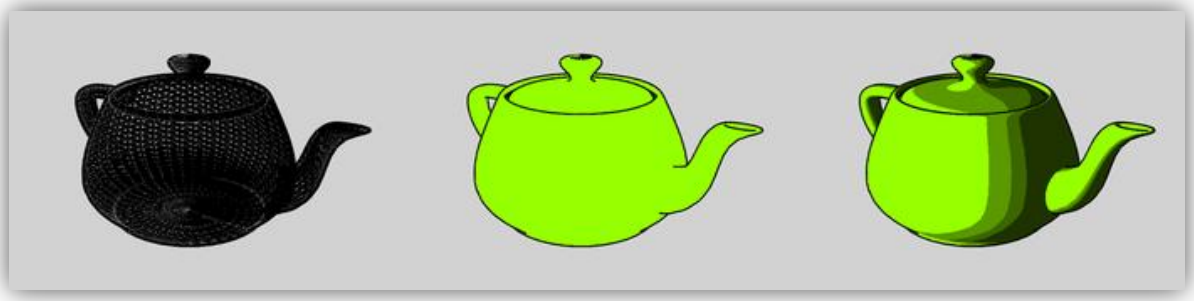
Trodimenzionalna računalna animacija je relativno novo područje računalne grafike i animacije. Postupci 3D računalne animacije su složeni procesi koji često zahtijevaju dobro poznavanje matematičkih osnova u grafici, umijeće trodimenzionalnog modeliranja objekata, te razna znanja o prirodi pokreta i strukturi objekta koji se animira. 3D računalnom animacijom animiraju se pokretni objekti kao što su ljudi, životinje ili roboti, ali i prirodni fenomeni, vatra, voda, kosa, tkanina i sl. Velika prednost 3D animacije je njena realističnost, pa je tako vrhunske 3D animacije teško razlikovati od stvarno snimljenih pokreta što čini ovu vrstu animacije pogodnu za specijalne efekte u filmovima. Danas postoji nekoliko najčešćih tehnika 3D računalne animacije koje su detaljnije objašnjene u nastavku.

Fotorealistična animacija se prvenstveno koristi kada se žele postići animacije koje nalikuju stvarnosti. Posebno velik izazov je prikazati ljude i ljudske pokrete što realističnijima. Animirani film *Final Fantasy: The Spirits Within* iz 2001. godine se često smatra prvim filmom koji pokušava modelirane ljudske likove prikazati što stvarnije. Fotorealistična animacija koristi složene algoritme renderiranja kojima se dobivaju objekti s velikim brojem detalja tako da bi što bolje oponašali stvarni svijet oko nas.

Slika 11.10. Fotorealistična animacija (*Fantasy: The Spirits Within*)

Najpoznatija tehnika **nefotorealistične** animacije je **cel-sjenčanje** (eng. *cel-shading*). Cel-sjenčanje je tehnika koja se najčešće koristi kada se želi oponašati stil stripova, ili drugih rukom crtanih radova. Proces cel-sjenčanja sastoji se od 3 ključna koraka (slika 2.11.).

Započinje s 3D modelom čiji se obrisi i konture linija dodatno naglašavaju. Model se zatim boja osnovnom teksturom, te u zadnjem koraku sjenča.

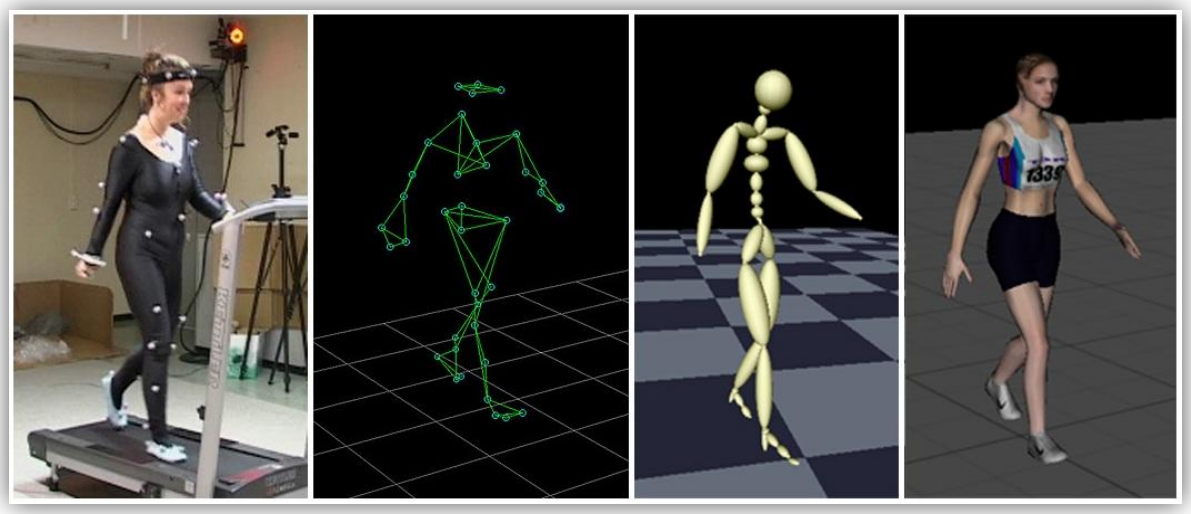


Slika 2.11. Cel-sjenčanje

Animacija izobličavanjem (eng. *morph target animation*) je tehnika koja se najčešće koristi zajedno sa skeletalnom animacijom koja je detaljnije objašnjena u sljedećem poglavlju. Animacija izobličavanjem ostvaruje pokrete transformirajući vrhove modela (eng. *vertex*) čime se dobiva izobličeni izgled mreže vrhova (eng. *mesh*). Pamte se lokacije vrhova u sličici u ključnom trenutku (eng. *keyframe*), a između njih vrhovi se kreću po točno definiranim stazama da bi se dobila što preciznija i „glađa“ animacija. Animacija izobličavanjem često se koristi prilikom prikaza pokreta lica, odnosno emocija (slika 2.12.).

Slika 2.12. Animacija izobličavanjem

Hvatanje pokreta (eng. *motion capture*) je složena tehnika snimanja pokreta iz stvarnog svijeta i njihovog prevođenja u digitalni oblik. Proces zahtjeva posebnu računalnu opremu i kamere, pa je često nepraktičan. Na objekt čiji se pokreti snimaju pozicioniraju se tzv. markeri na specifične točke tijela objekta. Markeri se snimaju s preciznim kamerama, a zatim se računalno obrađuju podaci o pozicijama, brzini i akceleraciji markera čime se



Slika 2.13. Hvatanje pokreta (eng. *motion capture*)

dobiva digitalna reprezentacija pokreta. Markeri mogu biti zvučni, inercijski, magnetski, LED ili bilo koja druga kombinacija spomenutih. Postupak hvatanja pokreta daje nam kao rezultat vrlo preciznu realističnu digitalnu reprezentaciju pokreta, ali zahtjeva skupu računalnu opremu, velik prostor i veći broj stručnjaka za ovakvu vrstu animacije.

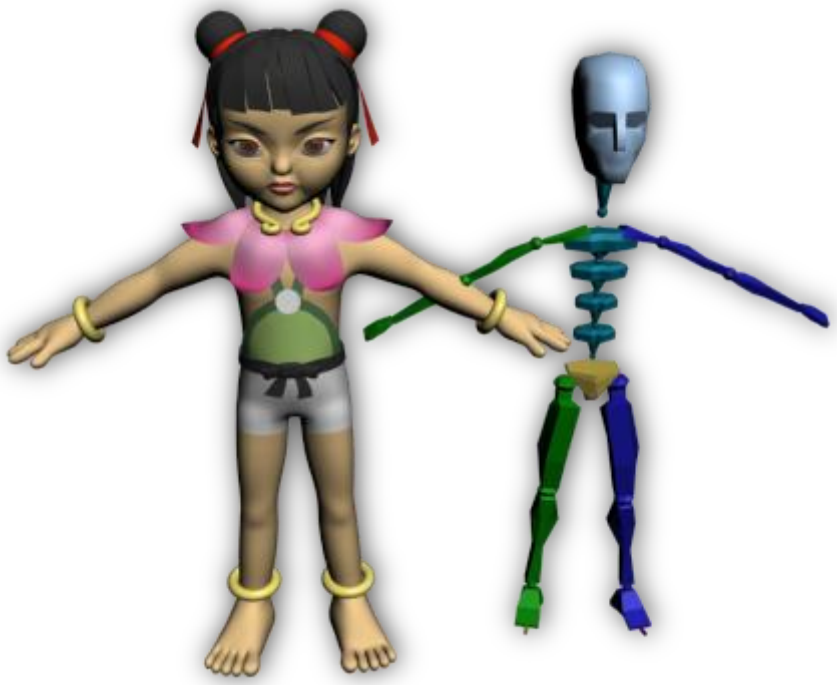
Skeletalna animacija (eng. *skeletal animation*) je najčešće korištena tehnika animiranja kralježnjaka. Ova tehnika korištena je u praktičnom dijelu ovog završnog rada, te je detaljnije objašnjena u sljedećem poglavlju 3: **Skeletalna animacija**.

3. Skeletalna animacija

Skeletalna animacija je tehnika računalne animacije u kojoj je model koji se animira predstavljen u dva glavna dijela:

1) površina koja ocrtava lik – **koža** (eng. *skin, mesh*)

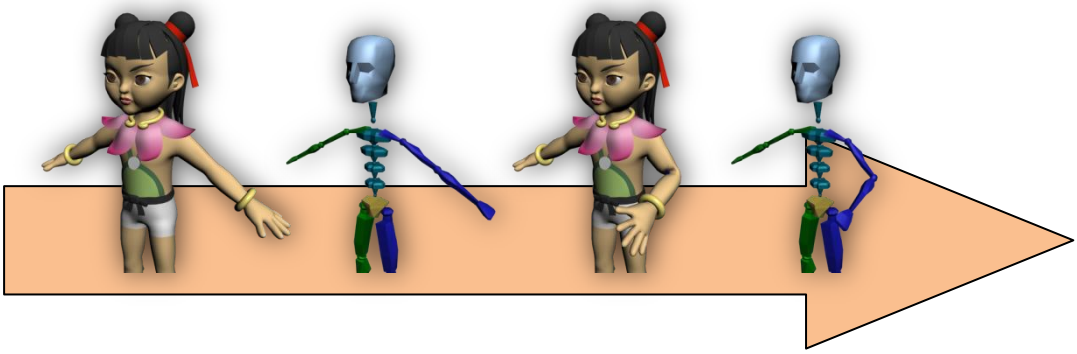
2) hijerarhijski skup povezanih kostiju – **kostur** (eng. *skeleton, rig*)



Slika 3.1. Dijelovi modela animiranog skeletalnom animacijom: koža (eng. *skin*, *mesh*) i kostur (eng. *skeleton*, *rig*)

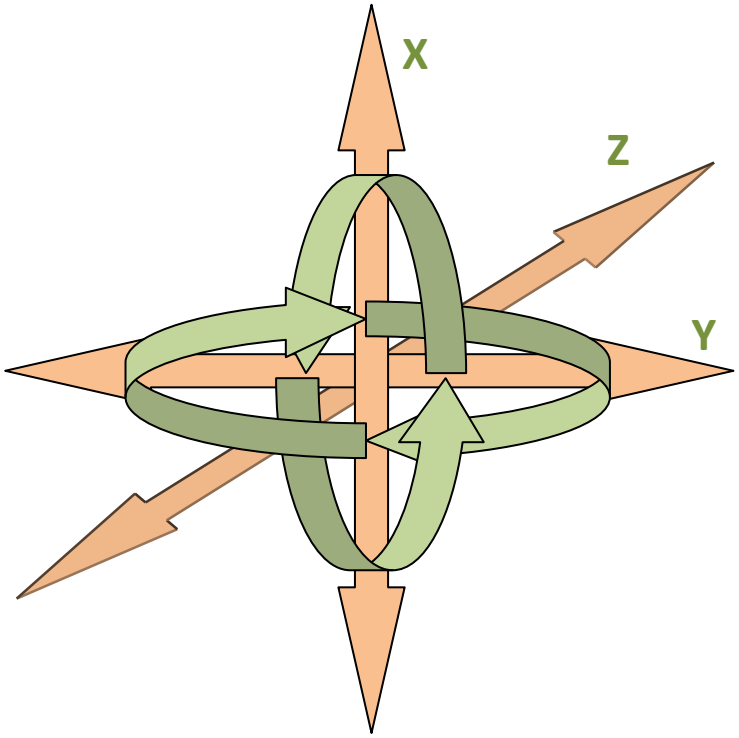
Ova tehnika najčešće se koristi za animiranje modela kralježnjaka kao što su ljudi, životinje, roboti i sl., ali se može koristiti i za kontrolu deformacije bilo kojeg drugog objekta. Ona olakšava često vrlo složene procese animiranja ljudskih i životinjskih likova pružajući korisniku pojednostavljeno korisničko sučelje i time zaobilazi složene algoritme i matematičke transformacije. Svaka kost vezana je za određeni dio kože, tj. mreže vrhova (eng. *vertex*) i svojom trodimenzionalnom transformacijom (položaj, mjerilo, orijentacija) utječe na položaje vrhova, a time i izgled mreže. Dodatno, svaka kost može imati pridruženu roditeljsku kost. Takav skup kostiju je hijerarhijski uređen i kao takav

pojednostavljuje postupke animiranja. Mijenjanjem položaja roditeljske kosti, primjerice nadlaktice, mijenjat će se i položaj kostiju djece, u ovom slučaju kosti podlaktice i šake (slika 3.2.).

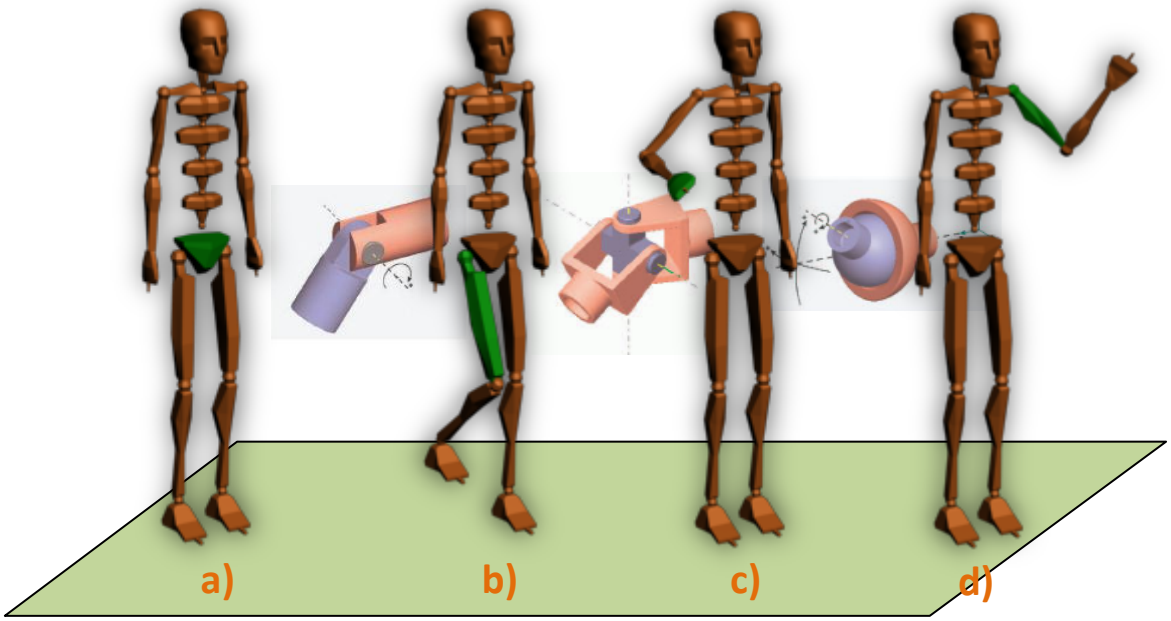


Slika 3.2. Pokreti kostiju lijeve ruke

Kosti modelu pridruženog kostura su međusobno povezane **zglobovima**. Zglob općenito može imati do tri translacijska i tri rotacijska stupnja slobode (eng. *degree of freedom*, *DOF*): rotacija oko X, Y i Z osi, te translacija po X, Y i Z osi (slika 3.3.). U ljudskom tijelu većina zglobova ima jedan ili dva rotacijska stupnja slobode, dok translacijskih zglobova nema. Iznimka je korijenski zglob, kojim se translatira i rotira čitavi kostur.



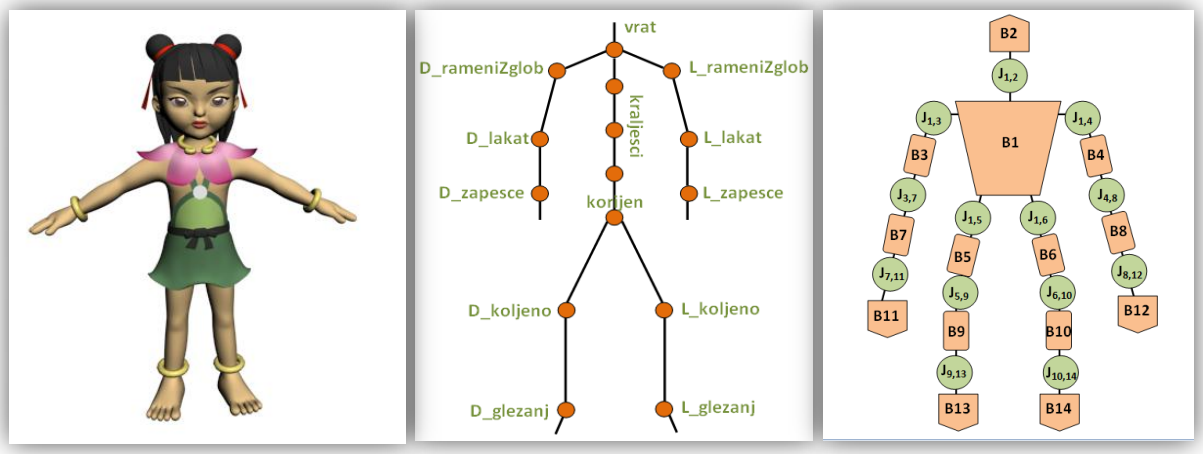
Slika 3.3. Rotacijski i translacijski stupnjevi slobode



Slika 3.4. Rotacijski i translacijski stupnjevi slobode ljudskih zglobova

Na slici 3.4. prikazani su rotacijski i translacijski stupnjevi slobode nekih zglobova ljudskog kostura. Slika a) prikazuje korijenski zglob koji ima sva tri translacijska i rotacijska stupnja slobode. Njegovom translacijom i rotacijom utječe se na čitavi kostur modela. Slika b) prikazuje zglob koljena koji ima samo jedan rotacijski stupanj slobode, a slika c) zglob zapešća s 2 rotacijska stupnja slobode. Slika d) prikazuje rameni zglob, jedan od rijetkih koji ima sva 3 rotacijska stupnja slobode.

U praksi se često modeli koji se animiraju metodom skeletalne animacije prikazuju apstraktnim reprezentacijama modela i stablastim grafom da bi se lakše ostvarili pojedini pokreti i animacija učinila razumljivijom i intuitivnom. Apstraktna reprezentacija modela je ništa drugo nego niz pokretnih zglobova koji su povezani nepokretnim linijama, kostima. Pojedinim zglobovima pridodaje se odgovarajući stupanj rotacijske i translacijske slobode. Ovakva struktura pogodna je za izradu grafa. Graf se hijerarhijski izgrađuje počevši od korijenskog čvora, a to je najčešće zdjelica ili trup. Iz tog segmenta izvode se ostale kosti djece koje ovise o obliku i položaju roditeljskih kostiju. Pojedine kosti djece vežu se na roditeljski čvor preko zgloba kojem se pridodaje odgovarajući stupanj slobode. Primjerice, iz podlaktične kosti izvode se kosti šake koje se povezuju zglobom zapešća, te su ovisne o transformacijama podlaktične kosti. Slika 3.5. na sljedećoj stranici prikazuje proces izrade apstraktne reprezentacije modela i stablastog grafa.



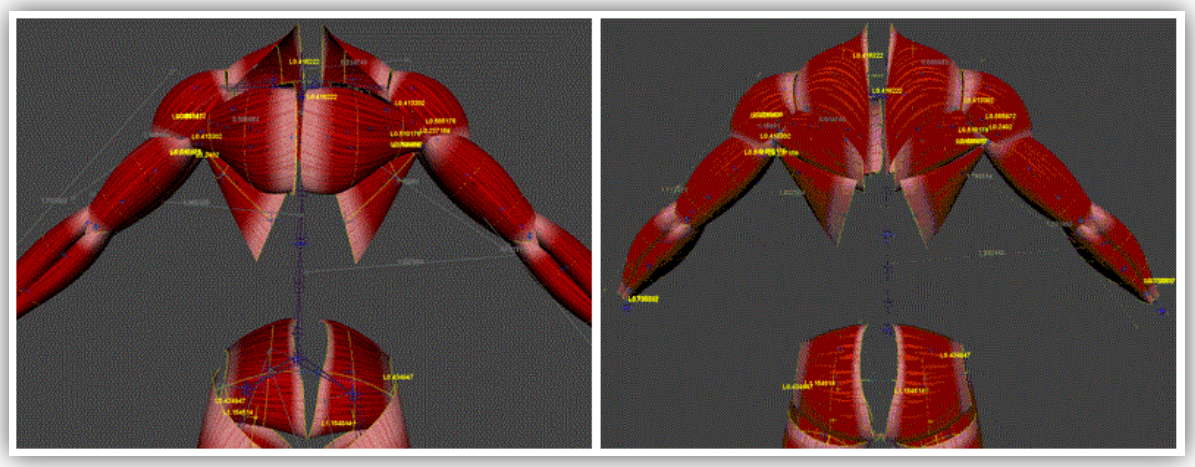
Slika 3.5. Apstraktna reprezentacija modela i graf stabla

Apstraktna reprezentacija modela na slici 3.5. prikazana je u sredini. Narančasti čvorovi predstavljaju zglobove, a linije kosti. Desno na slici 3.5. prikazan je graf. Narančasti četverokuti su kosti koje su označene slovom B (od eng. *bone*) i odgovarajućim brojem. Korijenski čvor je „kost“ označena s B1. Iz nje su izvedene kosti B2 (glava), B3 i B4 (nadlaktice), te B5 i B6 (bedrene kosti). Transformacija roditeljske kosti utječe na položaj i izgled izvedenih kostiju. Kosti su međusobno povezane zelenim krugovima koji predstavljaju zglobove. Svaki zglob označen je slovom J (od eng. *joint*) i odgovarajućim indeksima. Tako primjerice zglob $J_{4,8}$ predstavlja zglob lakta koji povezuje kosti B4 (nadlaktica) i B8 (podlaktica).

Vrlo zahtjevan dio skeletalne animacije je sama priprema modela, a pritom se prvenstveno misli na povezivanje kože (mreže) s kosturom. Ovaj postupak poznat je pod nazivom ***skinning***. Često korisnik sam povezuje vrhove mreže s odgovarajućim kostima, ali danas su dostupni, u sklopu većine alata za 3D modeliranje, heuristički algoritmi koji automatski izvršavaju *skinning*, nakon čega korisnik ručno radi korekcije. Na slici 3.6. prikazan je rezultat automatskog *skinninga* iza kojeg su potrebne dodatne korekcije. Naime, neki vrhovi mreže ostali su slobodni, tj. nisu pridruženi niti jednoj kosti. Zbog toga se prilikom pokreta nekih kostiju koža modela rasteže i često stvara probleme u animaciji ili prilikom njihovog učitavanja u programe pomoću raznih radnih okvira.

Slika 3.6. Rezultat automatskog *skinninga*

Prednosti skeletalne animacije je mnogo, a neke od njih su jednostavno korisničko sučelje za ostvarivanje pokreta, neovisnost kostiju, te mogućnost ostvarivanja velikih i kompliciranih pokreta bez razmišljanja o složenoj matematičkoj teoriji koja stoji iza njih. Glavna mana skeletalne animacije je nerealističan pokret mišića i kože. Ovaj problem moguće je riješiti dodavanjem posebnih kontrolora mišića na odgovarajuće kosti (slika 3.7.).



Slika 3.7. Kontrolori mišića povezani na kostur (eng. *muscle controllers*)

4. Kinematika

Kinematika je grana mehanike koja se bavi proučavanjem gibanja tijela ali ne uzimajući u obzir sile pod čijim se utjecajem to gibanje odvija. To područje proučava druga grana klasične mehanike – dinamika. Da bi pojednostavili, reći ćemo da se kinematika bavi formalnim opisivanjem kretanja. Kretnje se najčešće opisuju grafičkim modelima čijim stvaranjem postavljamo ograničenja i ovisnosti pokreta pojedinih dijelova tijela.

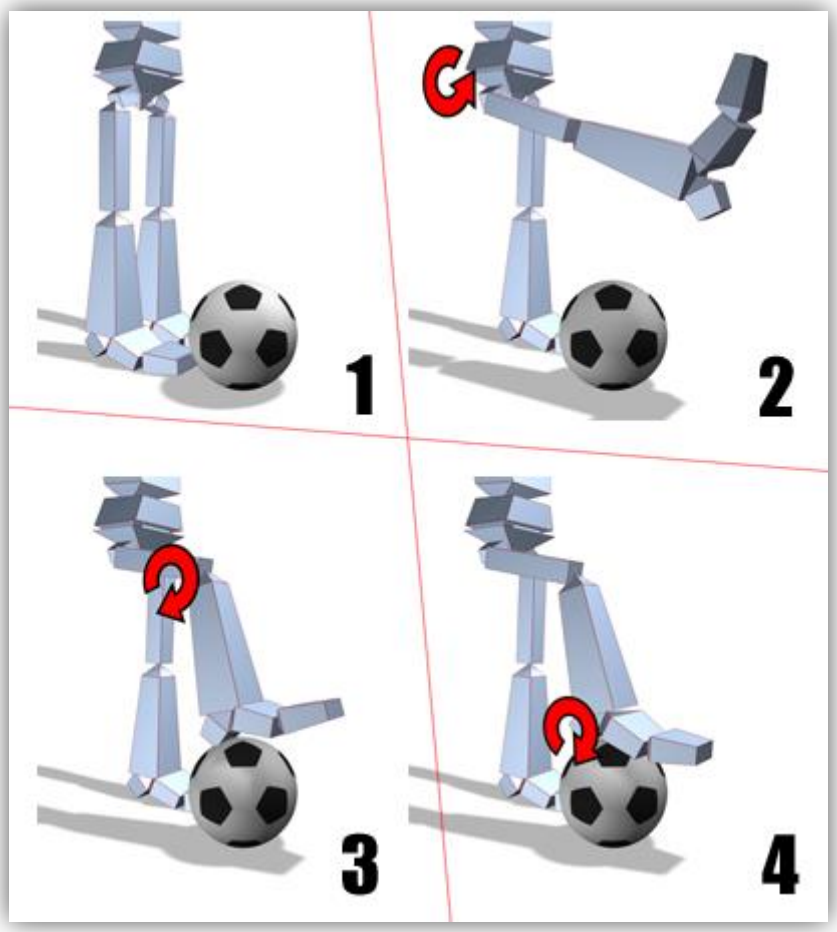
Kinematika je najveću primjenu našla u robotici, a u novije vrijeme znanja iz robotike primjenjuju se i u računalnoj animaciji. Najjednostavniji pristup u animaciji zasniva se na modelu kinematičkih lanaca. Radi se o hijerarhijskom modelu stablaste strukture sastavljenom od krutih segmenata – kostiju, te pokretnih grana – zglobova koji su ograničeni stupnjem slobode kretanja (DOF). Ovakav model detaljnije je opisan u prethodnom poglavlju i kao takav se najčešće koristi u metodama skeletalne računalne animacije.

Klasična mehanika razlikuje dvije vrste kinematike: **unaprijednu** i **inverznu**. Više o njima u nastavku ovog poglavlja.

4.1. Unaprijedna kinematika

Metode koje se zasnivaju na unaprijednoj kinematici danas se rijetko koriste u animaciji, jer zahtijevaju velik broj ulaznih podataka, a često kao rezultat daju neprirodne pokrete. Da bi opisali osnovni princip rada unaprijedne kinematike koristit ćemo termin *manipulator* koji je preuzet iz robotike, a predstavlja model koji se sastoji od nepomičnog zgloba – baze, te skupa povezanih pomičnih zglobova. Unaprijedna kinematika za manipulator zadaje položaje i kutove svih zglobova, te na osnovu njih izračunava položaj krajnje točke manipulatora. Dakle, unaprijedna kinematika bavi se problemom pretvorbe iz prostora manipulatora u prostor kartezijevih koordinata. Ako skeletalna animacija koristi unaprijednu kinematiku, najčešće se koriste modeli sa slijednim manipulatorima kod kojih se računanje položaja krajnje točke svodi na množenje matrica svih transformacija među susjednim segmentima. Slika 4.1. prikazuje lanac kostiju noge kao primjer manipulatora. Da bi krajnja točka manipulatora (u ovom slučaju stopalo) dohvatila

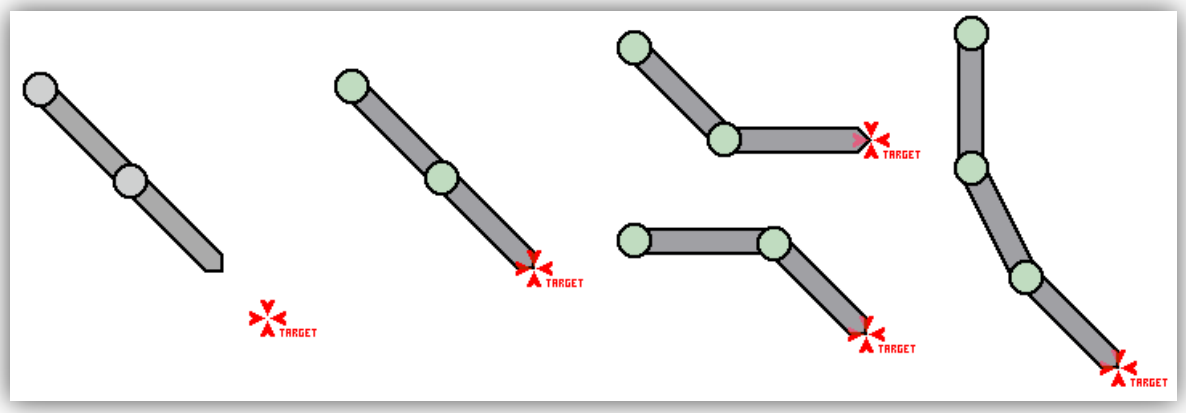
određenu točku u prostoru potrebno je odrediti niz transformacijskih matrica kojima se redom transformiraju segmenati u kinematičkom lancu. No ukoliko želimo transformirati roditeljske segmente u lancu pomicanjem zadnjeg segmenta lanca, koristit ćemo inverznu kinematiku.



Slika 4.1. Unaprijedna kinematika

4.2. Inverzna kinematika

Inverzna kinematika za zadanu točku prostora pronalazi položaje i kutove svih zglobova manipulatora takvih da njegova krajnja točka bude u zadanoj točki ako je ona dohvatljiva, ili što bliže zadanoj točki u slučaju da nije dohvatljiva. Ovo je puno složeniji problem nego onaj kojim se bavi unaprijedna kinematika, te kao takav često može biti riješen s beskonačno mnogo rješenja. Slika 4.2. prikazuje nekoliko problema koje rješava inverzna kinematika. U prvom slučaju cilj nije dohvatljiv, pa tako ne postoji rješenje koje dovodi krajnju točku manipulatora do ciljne točke u prostoru. Drugi slučaj predstavljen je ciljem koji se od baze manipulatora nalazi na udaljenosti koja je jednaka zbroju duljina svih njegovih segmenata. Ovaj slučaj ima jedno rješenje koje dovodi krajnju točku manipulatora do cilja. Ostala dva slučaja imaju više mogućih rješenja. Treći slučaj ima 2 rješenja, a četvrti njih nekoliko.



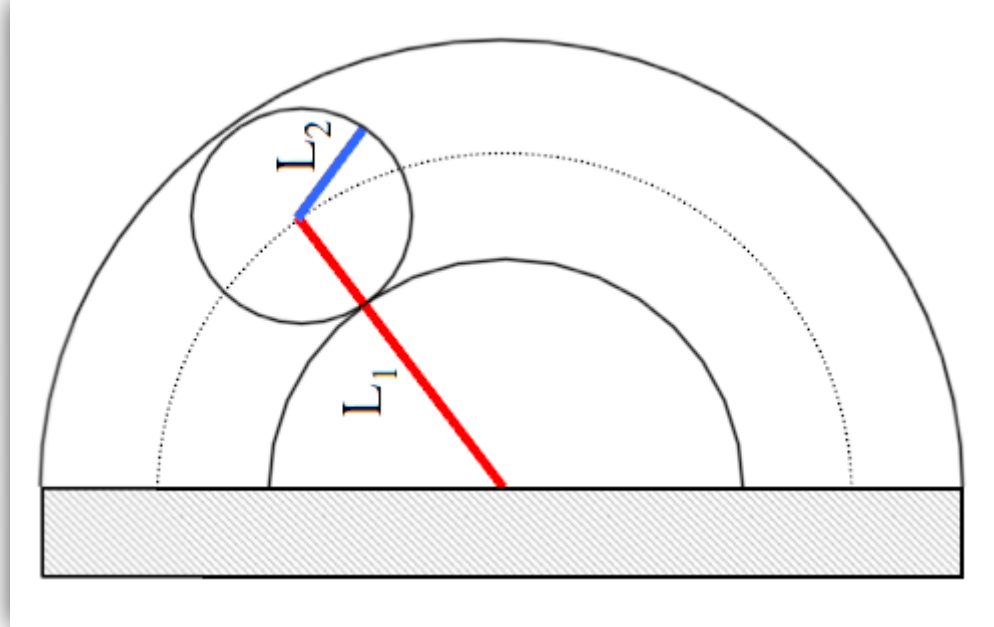
Slika 4.2. Primjer problema koje rješava inverzna kinematika

Prethodni primjer nas navodi na razmišljanje o upotrebljivosti inverzne kinematike. Ona se danas najčešće koristi u robotici i računalnoj animaciji, te je istisnula iz upotrebe unaprijednu kinematiku. Prilikom animiranja modela puno nam je jednostavnije postaviti krajnje točke manipulatora na temelju kojih će se izračunati odgovarajući položaji i kutovi zglobova manipulatora, nego zadavati čitav niz slijednih transformacija zglobova da bi dohvatili neku točku. Ukoliko bi problem sa slike 4.1. rješavali koristeći inverznu kinematiku, tada bi bilo dovoljno navesti točku prostora u kojoj se lopta nalazi, a ova metoda izračunala bi odgovarajuće transformacije zglobova.

Kao uvod u problematiku inverzne kinematike promotrit ćemo primjer jednostavnog manipulatora koji se sastoji od samo dva segmenta. Segmenti su kruti dijelovi modela sa fiksnom duljinom. Prije nego što krenemo na rješavanje ovog problema, opisat ćemo pojam **dohvatljivog prostora** (eng. *reachable workspace*). Na slici 4.3. je prikazan naš jednostavni manipulator s dva segmenta. Punom polukružnicom označen je doseg drugog segmenta, a isprekidanom doseg prvog segmenta. Ako su L_1 i L_2 duljine segmenata manipulatora, tada dohvatljivi prostor matematički možemo opisati sljedećom formulom:

$$L_1 - L_2 \leq dohvatljivi\ prostor \leq L_1 + L_2 \qquad (4.2.1)$$

Važno je opet napomenuti da zglobovi mogu imati ograničenja u kretanju, pa tako prvi zglob na slici 4.3. može pomicati segment L_1 samo 180°.



Slika 4.3. Dohvatljivi prostor

Vratimo se sada na naš primjer koji je opisan slikom 4.4. Naš zadatak je pronaći kutove θ_1 i θ_2 iz zadanog položaja vrha manipulatora $V(x_e, y_e)$. Da bi to uspjeli koristit ćemo se kosinusovim poučkom:

$$C^2 = A^2 + B^2 - 2AB\cos\theta \qquad (4.2.2)$$

Računamo najprije kut θ:

$$\cos\theta = \frac{x_e}{\sqrt{x_e^2 + y_e^2}} \tag{4.2.3}$$

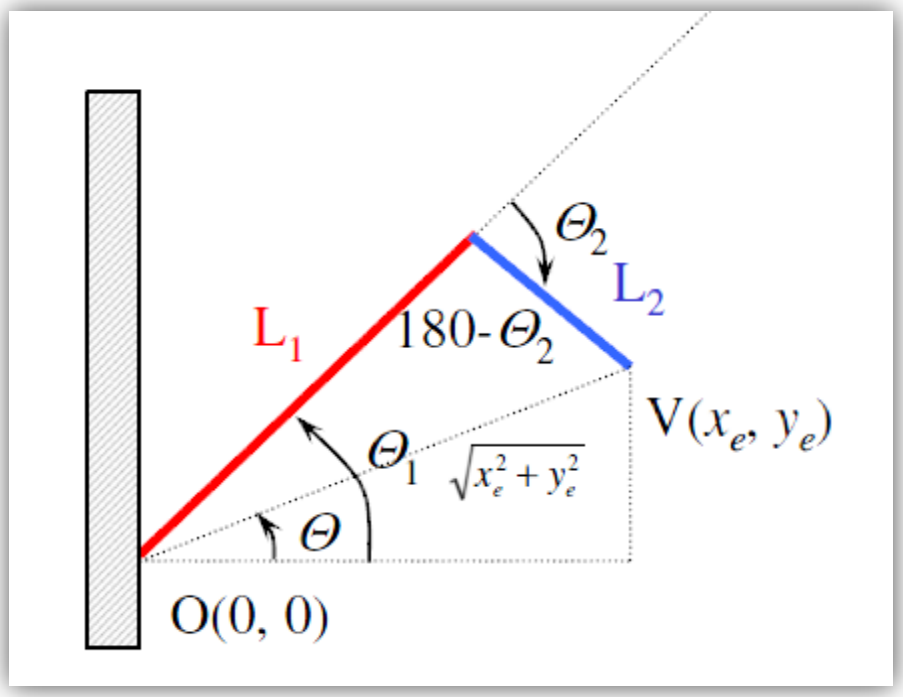
Sada kut θ_1 možemo izračunati preko (4.2.2) i (4.2.3):

$$\cos(\theta_1 - \theta) = \frac{L_1^2 + x_e^2 + y_e^2 - L_2^2}{2L_1\sqrt{x_e^2 + y_e^2}} \tag{4.2.4}$$

a kut θ_2 kao:

$$\cos(180° - \theta_2) = \frac{L_1^2 + L_2^2 - (x_e^2 + y_e^2)}{2L_1L_2} \tag{4.2.5}$$

Ovim postupkom dobivamo dva rješenja za primjer zadan slikom 4.4. Ona su simetrična u odnosu na liniju koja spaja ishodište lanca i zadanu točku cilja (na slici 4.4. je prikazana isprekidano). Broj ovako dobivenih rješenja povećava se povećanjem složenosti manipulatora, tj. povećanjem broja njegovih segmenata.



Slika 4.4. Primjer manipulatora s dva segmenta

Glavni nedostatak opisanog postupka izračuna kutova zglobova je teška aproksimacija linearnih pokreta. Razlog tome je nelinearnost funkcija sinus i kosinus koje koristimo u ovom izračunu. Ovaj problem riješit ćemo numeričkim rješavanjem koristeći Jakobijevu matricu.

4.2.1. Jakobijeva matrica

Za početak opisat ćemo vektorski jednostavni kinematički lanac. Neka su θ_i ulazne varijable, kutovi u zglobovima lanca, a $V(x_e, y_e, z_e)$ pozicija vrha manipulatora. Tada će nam vektor $\mathbf{X}$ predstavljati kutove rotacije cijelog kinematičkog lanca:

$$\mathbf{X} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \tag{4.2.6}$$

a vektor $\mathbf{Y}$ poziciju kraja lanca:

$$\mathbf{Y} = \begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix}. \tag{4.2.7}$$

U unaprijednoj kinematici vrijedit će sljedeća relacija (4.2.8) koja omogućava izračun pozicije ciljne točke na osnovu zadanih kutova svih segmenata modela.

$$\mathbf{Y} = \mathbf{F}(\mathbf{X}) \tag{4.2.8}$$

U (4.2.8) $\mathbf{F}$ je vektor nelinearnih funkcija f_1, f_2 i f_3:

$$\mathbf{F} = \begin{bmatrix} f_1(\theta_1 \dots \theta_n) \\ f_2(\theta_1 \dots \theta_n) \\ f_3(\theta_1 \dots \theta_n) \end{bmatrix}. \tag{4.2.9}$$

Jakobijeva matrica linearno modelira kretanje kraja lanca, tj. promjenu vektora $\mathbf{Y}$, u odnosu na trenutne promjene položaja segmenta sustava. Jakobijeva matrica ili kraće *Jakobijan* je matrica parcijalnih derivacija čitavog kinematičkog lanca u odnosu na vektor $\mathbf{Y}$.

$$\mathbf{J} = \frac{\partial \mathbf{Y}}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial x_e}{\partial \theta_1} & \cdots & \frac{\partial x_e}{\partial \theta_n} \\ \frac{\partial y_e}{\partial \theta_1} & \cdots & \frac{\partial y_e}{\partial \theta_n} \\ \frac{\partial z_e}{\partial \theta_1} & \cdots & \frac{\partial z_e}{\partial \theta_n} \end{bmatrix} \tag{4.2.10}$$

Svaki stupac matrice $\mathbf{J}$ u izrazu (4.2.10) predstavlja iznos vektora $\mathbf{Y}$ u odnosu na kut određenog segmenta θ_i, tj. pojedini članovi Jakobijana određuju doprinos pojedinih zglobova promatranoj promjeni na vrhu manipulatora.

S druge strane, problem inverzne kinematike opisan je sljedećim izrazom:

$$\mathbf{X} = \mathbf{F}^{-1}(\mathbf{Y}) \tag{4.2.11}$$

Za infinitezimalne vrijednosti promjene vektora kuta $\mathbf{X}$ i vektora pozicije ciljne točke $\mathbf{Y}$ vrijedi jednakost:

$$\Delta\mathbf{X} = \mathbf{J}^{-1}\Delta\mathbf{Y} \tag{4.2.12}$$

Prema jednakosti (4.2.12) potrebno je invertirati Jakobijevu matricu kako bi došli do rješenja, no ona nije nužno kvadratna. Tada se određuje pseudo-inverzna matrica tako da se jednadžba pred-multiplicira s transponiranom Jakobijevom matricom:

$$\mathbf{J}^{+} = (\mathbf{J}^{\mathbf{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathbf{T}} \tag{4.2.13}$$

Budući da je izračun pseudo-inverzne matrice vremenski zahtjevan, te je ona nestabilna u blizini singulariteta, umjesto nje često koristimo transponirana Jakobijevu matricu. Njeno korištenje nije matematički egzaktno, međutim teži ka rješenju.

Nakon što smo se upoznali s pozadinskom matematičkom teorijom, opisat ćemo algoritam izračuna svih kutova vektora $\mathbf{X}$:

1) izračunaj Jakobijan $\mathbf{J}$

2) izračunaj pomak $\Delta\theta$

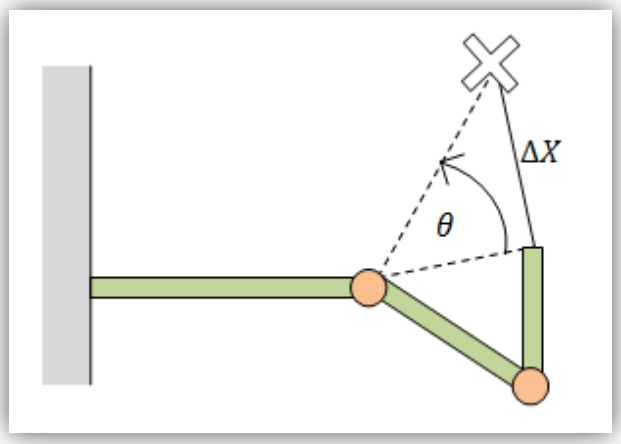
3) $\theta = \theta + \Delta\theta$

4) ažuriraj sve zglobove na osnovu vektora $\mathbf{X}$

5) ponavljaj prethodne korake dok $\mathbf{Y}_{trenutno}$ nije unutar prostora tolerancije ciljne pozicije

4.2.2. CCD algoritam

CCD algoritam (eng. *Cyclic Coordinate Descent*) zasniva se na jednostavnoj ideji koja rješava probleme s jednim stupnjem slobode (1DOF) duž kinematičkog lanca. Radi se o iterativnom postupku koji za svaki zglob u iteraciji traži transformaciju koja će minimizirati udaljenost vrha manipulatora do cilja. Ovaj postupak, iako je vrlo jednostavan i intuitivan, najčešće se izbjegava u tehnikama računalne animacije. Razlog tome su gibanja koja nisu glatka, te prisutna neujednačenost promjena u zglobovima.



Slika 4.5. Primjer pomaka koji minimizira udaljenost do cilja

5.1. Uvod

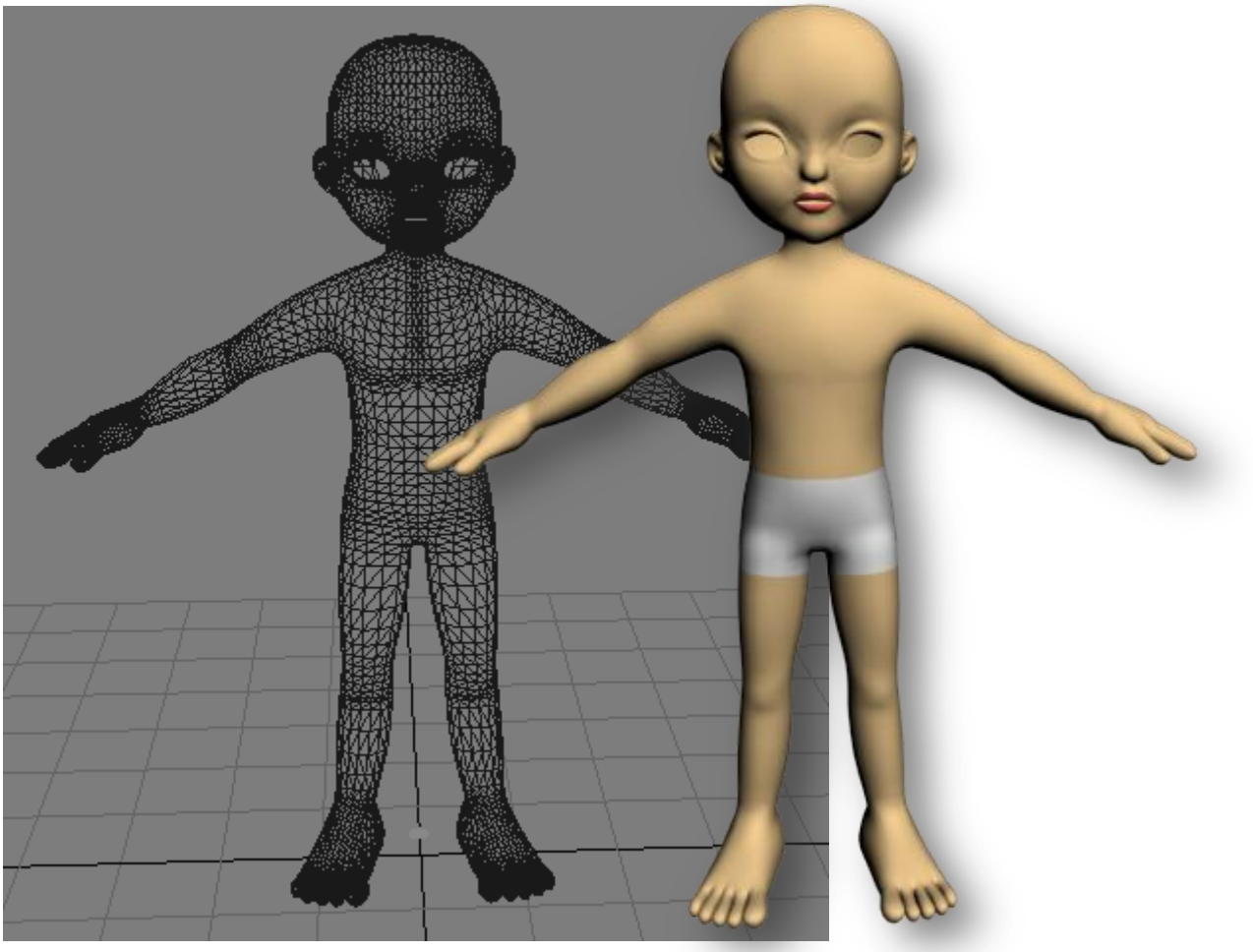
Rezultat ovog završnog rada je programsko rješenje koje omogućuje interaktivno upravljanje pokretima i kretanjem personificiranog 3D objekta. U ovom poglavlju opisan je cjelokupni proces programskog ostvarenja zadatka. Prilikom implementacije korišten je *Microsoft XNA Framework 4.0*, radni okvir namijenjen izradi igara za Windows platforme. *Microsoft Visual Studio 2010* poslužio je kao interaktivno razvojno okruženje, a kod je pisan u jeziku C#. Prilikom ostvarenja ovog programskog produkta bilo je potrebno pripremiti 3D model za animaciju, što je učinjeno programskim alatom *Autodesk 3D Studio Max 2010 Student Version*. Sam model je besplatno preuzet preko internetske stranice *3DModelFree* (www.3dmodelfree.com).

Budući da *XNA* radni okvir pruža samo djelomičnu podršku za skeletalnu animaciju, pri izradi ovog programskog produkta iskorištene su neke gotove klase i programska rješenja koja su dostupna na stranicama *Microsoft MSDN-a* specijaliziranim za izradu igara i mobilnih aplikacija (www.create.msdn.com). Konkretno, korištena su programska rješenja primjera nazvanog *Skinned Model Sample,* koja se mogu slobodno koristiti prema uvjetima licence *Microsoft Permissive License* (*Ms-PL*). Više o detaljima ovog programskog rješenja bavimo se u nastavku ovog poglavlja.

Slika 5.1. Korištena programska podrška

5.2. Priprema 3D modela

Da bi uz što manje muke učitali model u program te ga zatim animirali, za početak je potrebno pronaći što jednostavniji model ljudskog lika, kojeg kasnije možemo postepeno nadograđivati. Iz tog razloga na samom početku implementacije izabran je model koji se sastoji od samo jedne mreže (eng. *mesh*) čime je uvelike olakšan postupak skinninga, pa tako i animiranja (slika 5.2.).



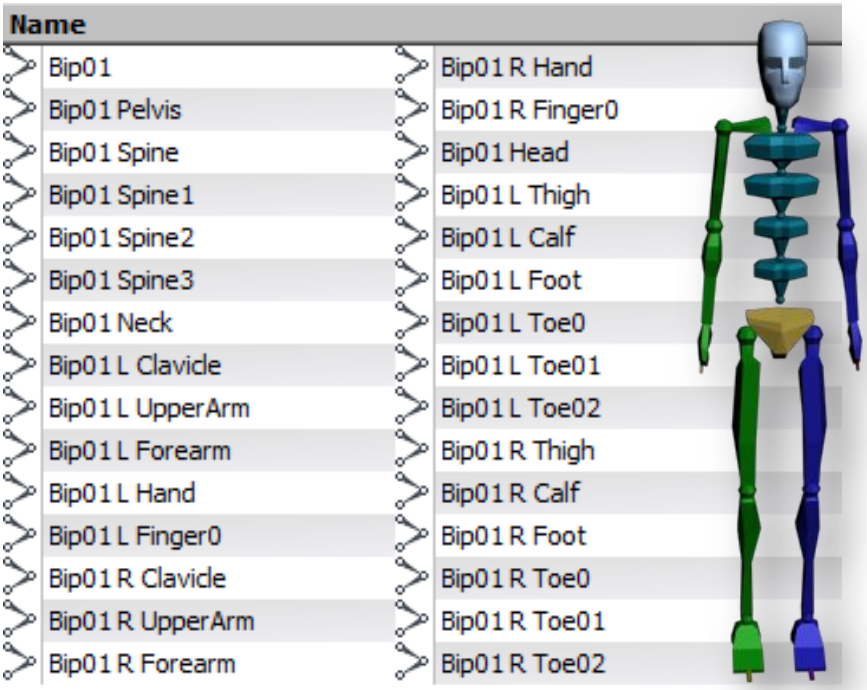
Slika 5.2. Prvotni 3D model ljudskog lika

Ovakav model pogodan je za prvi susret s postupcima skinninga, skeletalne animacije, ali i učitavanjem modela pomoću XNA radnog okvira. U slučaju kompliciranijeg modela moguće su brojne pogreške prilikom njegovog učitavanja u program, a tada je teško povezati pogreške s njihovim stvarnim uzrokom.

5.2.1. Skinning

Postupak povezivanja kože (mreže) modela s kosturom, tj. nizom kinematičkih lanaca, poznat je pod nazivom ***skinning***. Radi se o izuzetno zahtjevnom poslu koji se sastoji od pridruživanja vrhova mreže odgovarajućim kostima. Na sreću danas postoji niz programskih alata koji olakšavaju skinning, pa tako i Autodeskov 3D Studio Max pruža vrlo dobru podršku.

Da bi povezali vrhove mreže s kosturom potreban nam je niz kinematičkih lanaca koji predstavljaju kosti ljudskog tijela. Lance, tj. kosti, možemo sami konstruirati ukoliko radimo s neuobičajenim likovima, no za naše potrebe dobro će koristiti gotov ljudski kostur koji je u 3DS Maxu nazvan ***biped***. Na slici 5.3. prikazan je *biped*, te niz kostiju koje ga sačinjavaju.

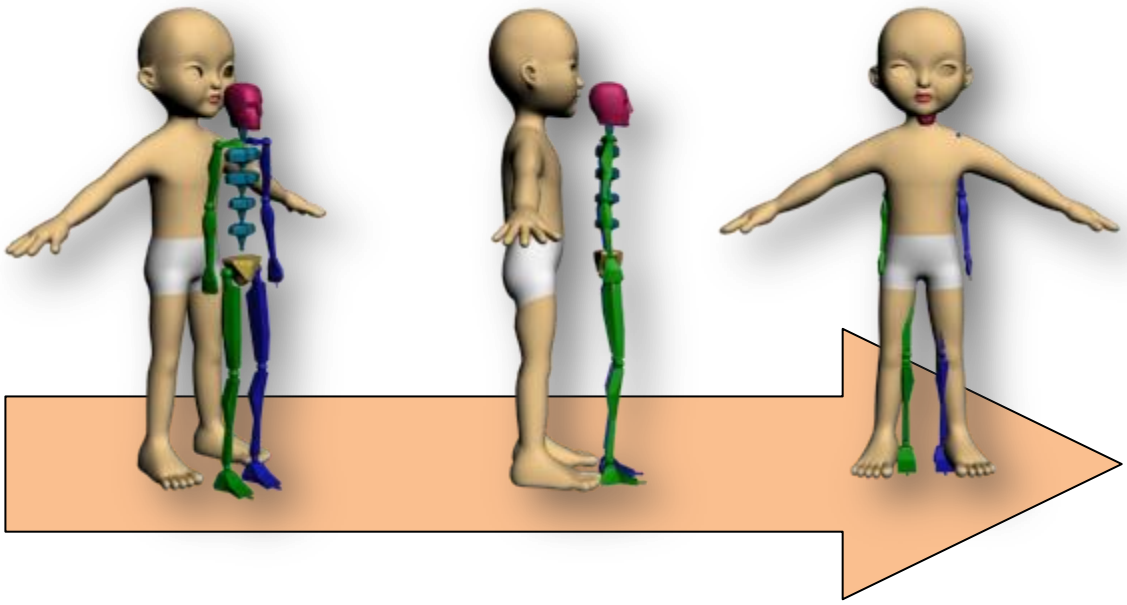


Slika 5.3. *Biped* i kosti koje ga sačinjavaju

Prvi korak u procesu skinninga je stvoriti kostur odgovarajuće veličine i smjestiti ga unutar modela. Budući da proporcije kostura često ne odgovaraju onima modela, potrebne su transformacije (translacija, rotacija, skaliranje) određenih kostiju kako bi ostvarili što kvalitetnije podudaranje modela i kostura. Primjerice, često se ne podudaraju dimenzije udova kostura i kože, pa se odgovarajuće kosti skaliraju. Također, ponekad je potrebno

rotirati kosti oko zglobova, jer se modeli pripremljeni za animaciju uobičajeno isporučuju s raširenim rukama i nogama kako bi se pojednostavili postupci skinninga.

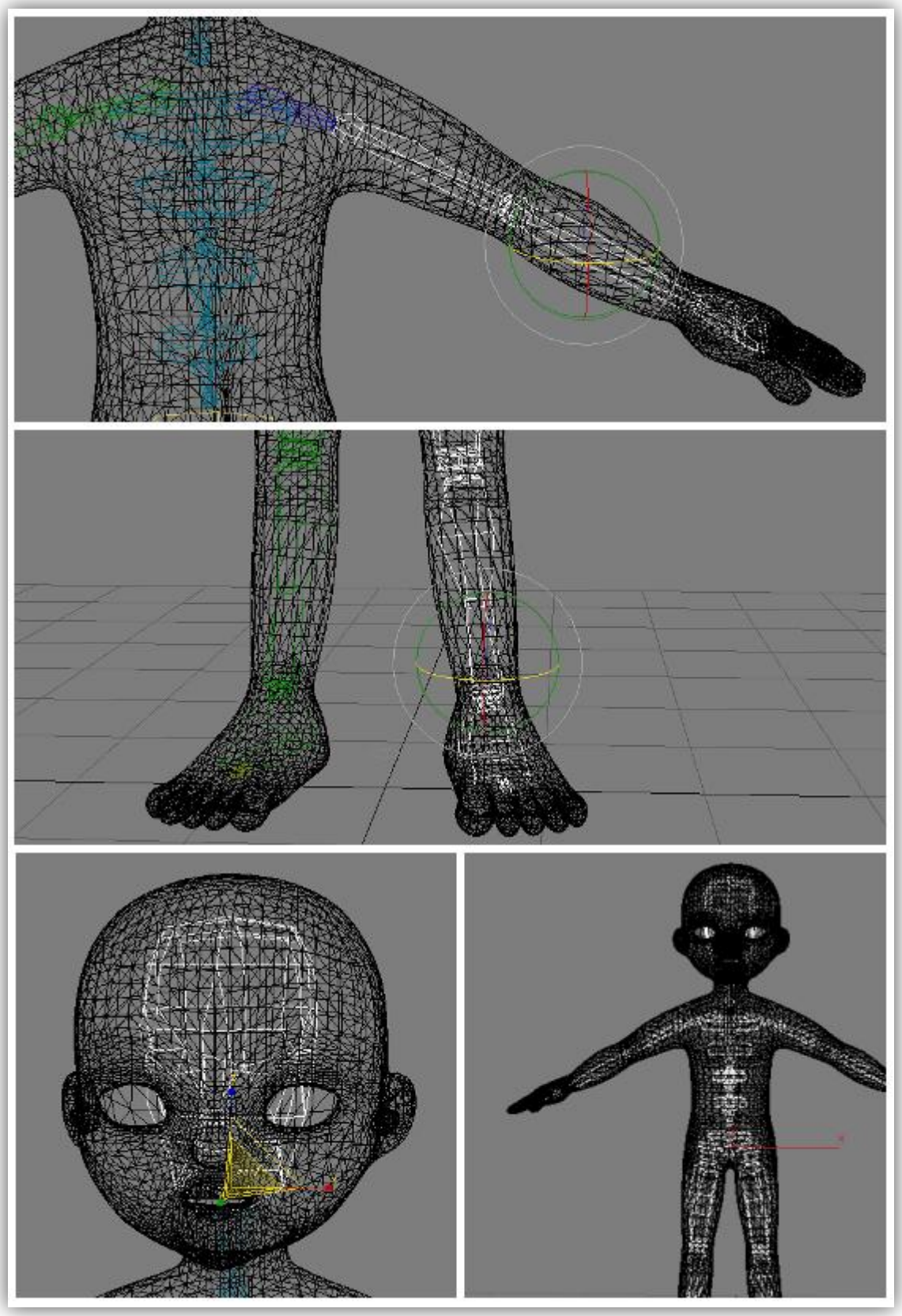
Stvorit ćemo kostur jednake visine kao i naš prvotni model. Kostur smještamo unutar mreže, te zatim transformiramo kosti kako bi postigli da se niti jedan dio kostura ne nalazi izvan mreže (slika 5.4.).



Slika 5.4. Stvaranje kostura i smještanje unutar mreže

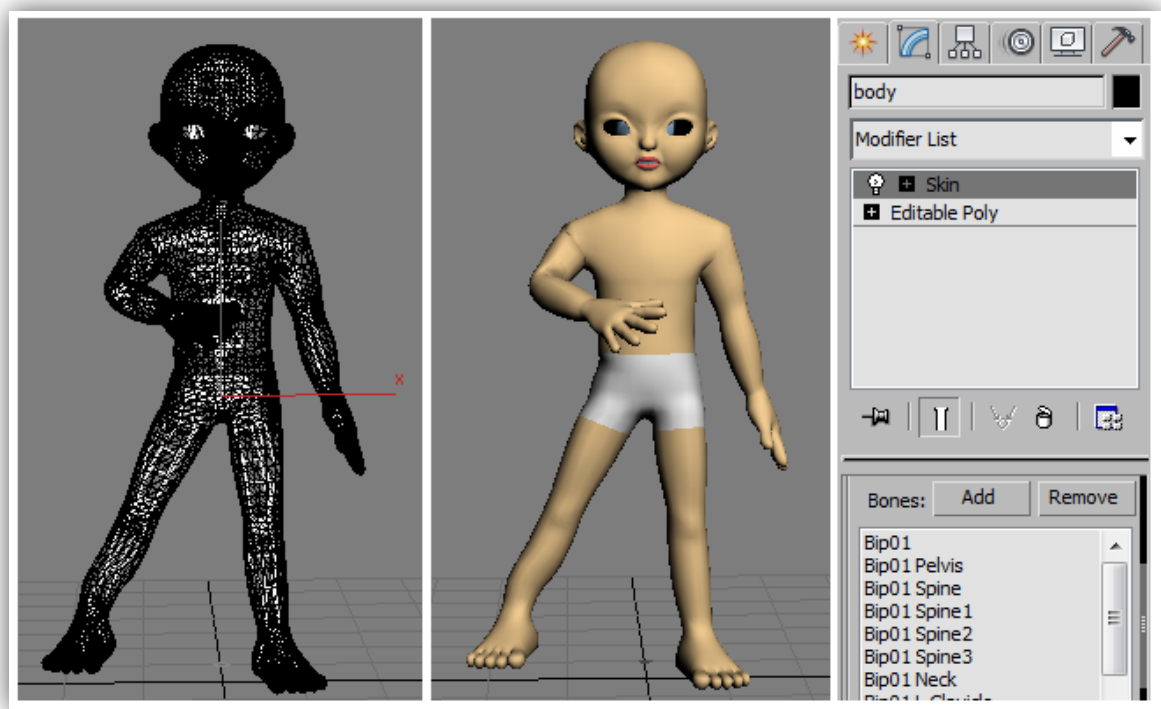
Budući da naš model ima malo kraće ruke i noge, skalirat ćemo kosti udova. Također rotirat ćemo ruke oko zglobova ramena, tj. noge oko zglobova zdjelice, kako bi udovi „sjeli“ unutar mreže. Nakon što smo smjestili kostur unutar mreže, izvršit ćemo još manje transformacije položaja ključnih kostiju i vrata, te ćemo skalirati glavu kostura. Slika 5.5. prikazuje upravo opisani proces smještanja kostura unutar mreže.

Prethodno opisani postupak iziskuje dosta vremena. Ukoliko želimo da animacija izgleda što uvjerljivije valja potrošiti dodatno vrijeme i postići što bolje podudaranje mreže i kostura. Radi jednostavnosti naš kostur nema prste na rukama i nogama, jer bi to uvelike zakompliciralo ovaj proces, no ako želimo uvjerljiviju animaciju stvorit ćemo dodatne kosti i pažljivo ih smjestiti unutar mreže.



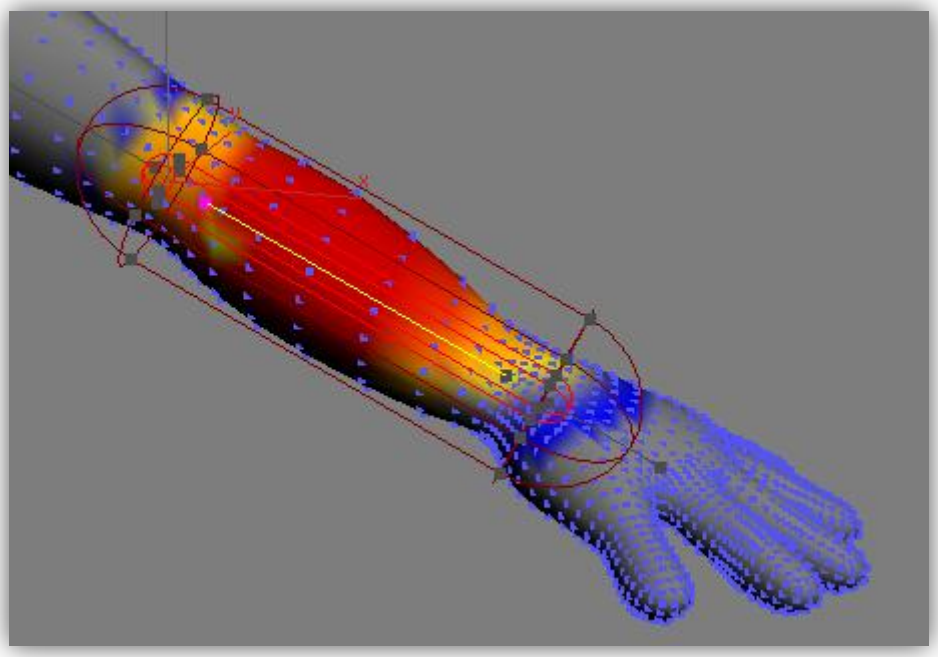
Slika 5.5. Transformacija kostiju
a) skaliranje i translacija ruke
b) skaliranje i translacija noge
c) skaliranje glave
d) podudaranje kostura i mreže

Nakon što smo ispravno smjestili kostur unutar mreže, pridružit ćemo vrhove mreže kostima. Ovdje ćemo koristiti gotove heurističke algoritme koji povezuju kostur i mrežu, a dostupni su u 3DS Maxu preko tzv. ***skin modifiera***. Postupak je vrlo jednostavan: obilježit ćemo mrežu, te na nju primijeniti *skin modifier* kojem ćemo zadati sve kosti kostura (slika 5.6. - desno). Naš prvotni model sastoji se od samo jedne mreže, pa nam je posao olakšan. Kasnije, kada ćemo dodavati dodatne objekte, tj. mreže, vezat ćemo ih samo na određene kosti, ne za čitavi kostur. Primjerice, ako dodajemo mrežu koja predstavlja kosu, vezat ćemo ju preko *skin modifiera* samo za kosti glave.



Slika 5.6. Pokreti kostiju utječu na izgled i položaj mreže

Nakon što smo primijenili *skin modifier*, pomicanjem kosti utjecat ćemo na izgled i položaj mreže koja je vezana za tu kost (slika 5.6.), stoga je važno da niti jedan vrh mreže ne ostane slobodan, tj. nevezan za niti jednu kost. To će rezultirati rastezanjem mreže i nepravilnim pokretima, a najvjerojatnije nećemo uspjeti učitati model u program. U slučaju da su nam neki vrhovi ostali slobodni jednostavno ćemo proširiti radijus u kojem odgovarajuća kost dohvaća vrhove. To ćemo učiniti transformacijom tzv. **omotnice** (eng. *envelope*), koja obuhvaća sve vrhove na koje se primjenjuje *skin modifier* (slika 5.7.).



Slika 5.7. Omotnica (eng. *envelope*)

Na slici 5.7. prikazana je omotnica podlaktične kosti. Omotnica obuhvaća vrhove obojane nijansama crvene boje. Nijansa crvene boje označava i težinu kojom kost utječe na vrh mreže, pa će tako kost jače utjecati na vrhove u crvenom području, a slabije na vrhove u žutom, dok neće uopće utjecati na vrhove u sivom području.

Nakon što smo uspješno završili s postupkom skinninga možemo krenuti na sljedeći korak: animaciju 3D modela.

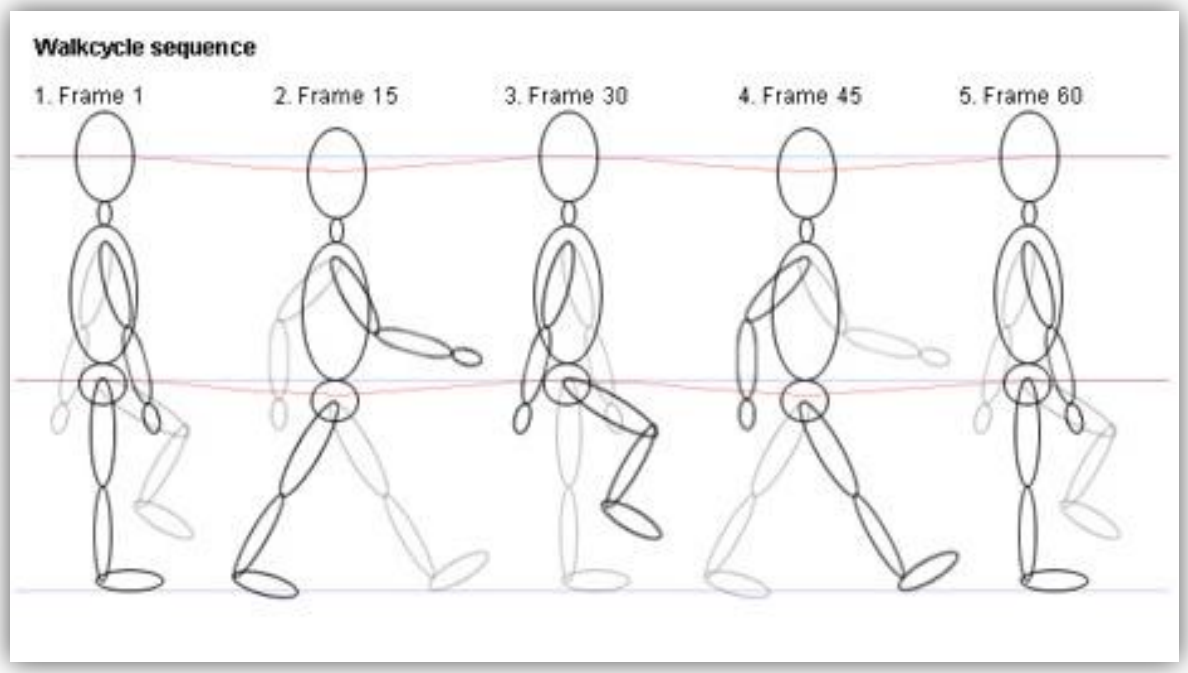
5.3. Animacija 3D modela

Prethodno oblikovani model spreman je za animaciju. Da bi animirali naš ljudski lik ponovo ćemo se poslužiti 3D Studio Maxom, a zatim ćemo model s gotovom animacijom učitati u program metodama XNA radnog okvira.

3D Studio Max pruža vrlo dobru podršku za 3D animaciju, te uvelike olakšava njen proces izrade. Prilikom animiranja našeg lika koristit ćemo tzv. **ključne trenutke ili sličice** (eng. *keyframe*). Svaka animacija sastoji se od niza trenutaka, tj. sličica, koje prikazuju objekt u određenom položaju. Njihovim brzim prikazivanjem dobivamo dojam pokreta. Kada bi morali animirati objekt tako da svaku sličicu sami slažemo, tj. da za svaki trenutak transformiramo objekt u određeni položaj, izgubili bismo puno vremena, a animacija nam najvjerojatnije ne bi bila „glatka“. Iz tog razloga najčešće koristimo ključne sličice i matematičku metodu **interpolacije**. 3D Studio Max na temelju dvije ključne sličice, koje mu zadamo u određenom vremenskom razmaku, interpolira promjenu položaja 3D modela kako bi ostvario glatku animaciju između zadanih ključnih sličica. Ovaj princip nam uvelike olakšava postupke animacije, te ju čini uvjerljivijom.

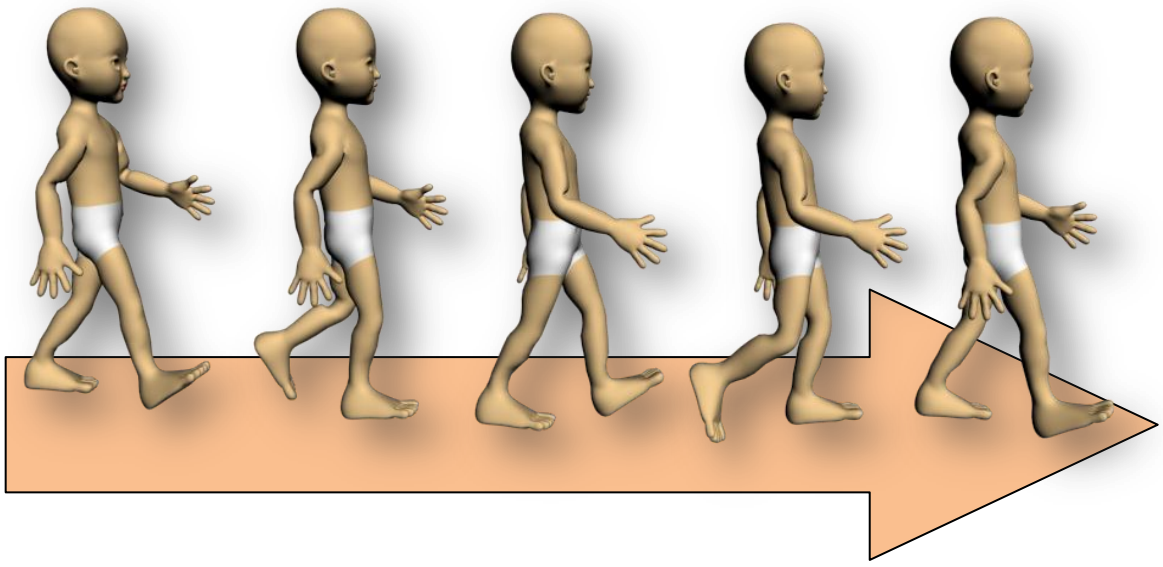
Za početak ćemo animirati hod našeg ljudskog lika. Budući da animaciju hoda želimo učitati u program i ostvariti interaktivno kretanje modela po sceni, animirat ćemo samo jedan ciklus hoda (eng. *walk cycle*) u kojem su prva i zadnja sličica jednake. Uzastopnim ponavljanjem ovog ciklusa ostvaruje se dojam hodanja. Ovaj princip često se koristi prilikom animacija likova u video igrama. Pritiskom na kontrole za pomicanje modela, on se translatira, a prilikom translacije pokreće se „snimljena“ animacija hoda na mjestu. Iako 3D Studio Max pruža mogućnost za jednostavnu animaciju hoda postavljanjem otisaka stopala (eng. *footsteps*), mi ćemo koristiti ključne sličice kako bi animirali ciklus od dva koraka u hodu na mjestu.

Prilikom animacije hoda koristit ćemo se nekim znanjima iz kineziologije. Zanimaju nas položaji određenih dijelova tijela prilikom pokreta u hodu. Na slici 5.8. prikazani su položaji ljudskog tijela tijekom dva koraka hoda. Ova shema služit će nam kao osnova za animaciju hoda našeg lika. Najprije ćemo postaviti ključne sličice prema shemi na slici 5.8., a zatim ćemo postepeno dodavati dodatne ključne sličice kako bi postigli glađu i uvjerljiviju animaciju.



Slika 5.8. Ciklus hoda

Na sljedećoj slici prikazan je naš 3D model u ključnim trenutcima koji su odabrani prema shemi na slici 5.8. Pokreti između ključnih trenutaka ostvareni su interpolacijom, a da bi oni bili što uvjerljiviji dodavat ćemo dodatne ključne sličice.



Slika 5.9. 3D model u ključnim trenutcima

Prethodno opisani način animiranja ljudskih likova nije pretjerano kompliciran, ali ako želimo postići uvjerljivu i glatku animaciju morat ćemo utrošiti više vremena i isprobavati animaciju dodavanjem raznih ključnih sličica u određenim trenutcima. Ovaj postupak, osim dosta truda i znanja, zahtjeva i kreativnost animatora.

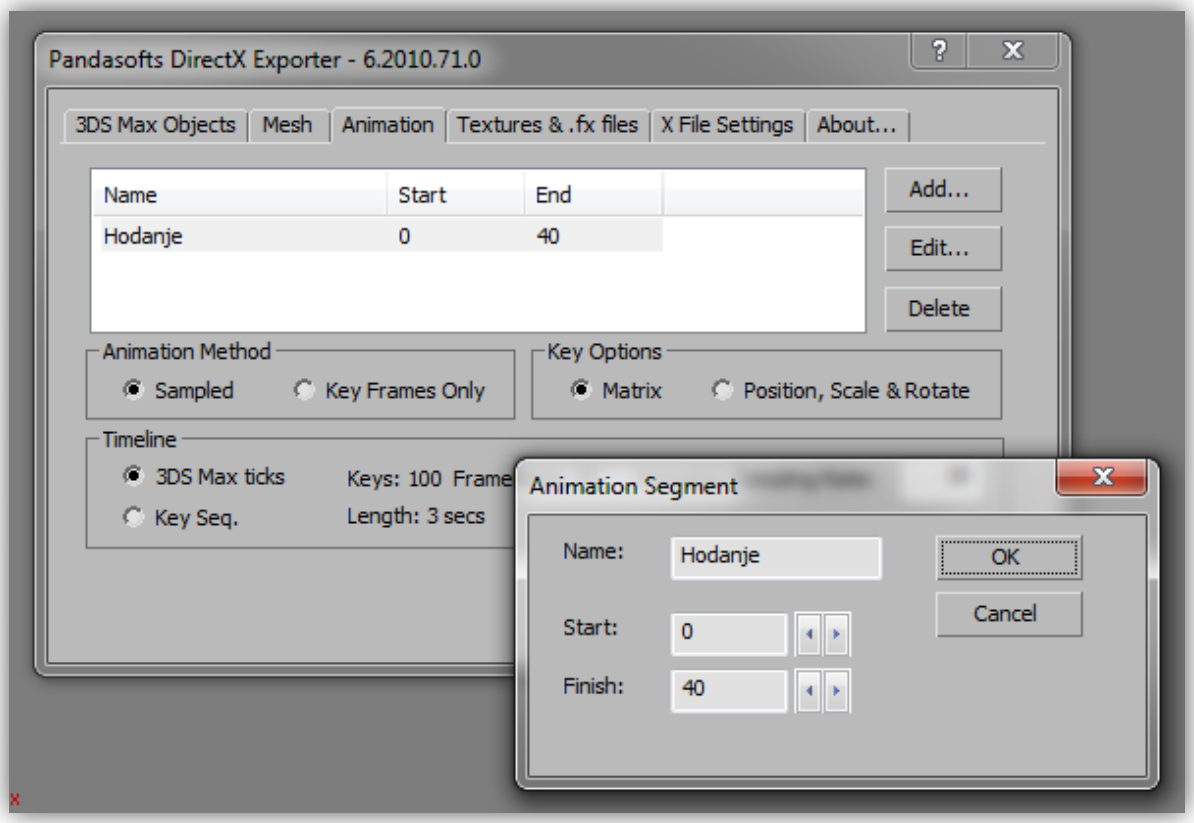
Dakle, sve animacije koje se koriste u ovom programskom produktu ostvarene su na sličan način: odabirom ključnih sličica, te postepenim dodavanjem novih. Odabir ključnih sličica često se oslanja na znanja iz drugih znanosti kao što je kineziologija, ali i na iskustvu drugih animatora. Tako su se u ovom radu koristile sheme i naputci iz knjige *3DS Max Animation with Biped* čiji su autori animatori Michele Bousquet i Michael McCarthy.

Nakon što smo animirali naš ljudski lik preostaje nam ga učitati u program korištenjem XNA radnog okvira i gotovih programskih rješenja za skeletalnu animaciju koja će biti kasnije dodatno objašnjena.

5.4. Učitavanje animiranog 3D modela

Budući da koristimo XNA radni okvir morat ćemo spremiti naš animirani model u točno određenom formatu. XNA podržava dva osnovna formata zapisa 3D modela: **FBX**, format u vlasništvu Autodeska, te **X**, standardni DirectX format. U našem programskog rješenju koristimo format X.

Da bi izvezli animirani 3D model u X formatu koristit ćemo besplatnu nadogradnju za 3D Studio Max: ***Panda DirectX Exporter*** tvrtke *Pandasoft*. Panda Exporter omogućava nam jednostavan izvoz modela u X formatu uz odabir raznih korisnih opcija. Na slici 5.10. nalazi se grafičko sučelje Panda Exportera. Prilikom izvoza pozornost treba obratiti na animacije. Panda Exporter omogućava izvoz većeg broja animacija koje ograničavamo početnom i završnom sličicom (eng. *frame*). Svakoj animaciji pridodajemo ime pomoću kojeg ćemo joj moći pristupiti iz programa. Ostale postavke za sada ostavljamo nepromijenjenima.



Slika 5.10. Panda DirectX Exporter

Nakon ovog koraka dobili smo animirani 3D model u X formatu koji je spreman za učitavanje u program.

Iako učitavanje animiranog modela pomoću programskog rješenja danog primjerom *Skinned Model Sample,* dostupnog na stranicama MSDN-a, ne zahtjeva puno posla, ovaj trenutak predstavlja „usko grlo“ u implementaciji našeg rješenja. Naime, prilikom učitavanja modela „redovito“ se javljaju brojne greške čiji je uzrok vrlo teško odgonetnuti. Zato smo krenuli s relativno jednostavnim modelom kako bi si za početak olakšali proces ispravljanja pogrešaka. Primjeri grešaka i upozorenja koji su se javljali prilikom učitavanja modela opisani su u sljedećem potpoglavlju 5.4.1. Problemi prilikom učitavanja modela.

Budući da je programsko rješenje dano primjerom *Skinned Model Sample* detaljno opisano u jednom od sljedećih poglavlja, za sada ćemo dati samo grubi opis koda za učitavanje modela.

```
protected override void LoadContent()
{
     // Load the model.
     currentModel = Content.Load<Model>("cura");

     // Look up our custom skinning information.
     SkinningData skinningData = currentModel.Tag as
SkinningData;

     if (skinningData == null)
        throw new InvalidOperationException
           ("This model does not contain a SkinningData tag.");

     // Create an animation player, and start decoding an
animation clip.
     animationPlayer = new AnimationPlayer(skinningData);

     AnimationClip clip =
skinningData.AnimationClips["Hodanje"];

     animationPlayer.StartClip(clip);
}
```

Prikazani dio koda je tijelo obavezne metode `LoadContent` XNA radnog okvira koja prilikom pokretanja programa učitava sve potrebne resurse kao što su modeli, teksture, zvukovi, animacije i sl. Model učitavamo pozivom metode `Content.Load<Model>` i navođenjem imena modela kao parametra metode. U sljedećem koraku dohvaćamo sve podatke o modelu animiranom skeletalnom animacijom, te stvaramo nove objekta tipa `AnimationPlayer` i `AnimationClip`. Zatim dohvaćamo animacije pomoću

njihovog imena koje smo im zadali prilikom izvoza Panda Exporterom. Metodom `StartClip(AnimationClip clip)` pokrećemo dohvaćenu animaciju. Za sada nećemo ulaziti u detalje svih klasa i metoda koje se ovdje koriste, već ćemo to ostaviti za sljedeće poglavlje.

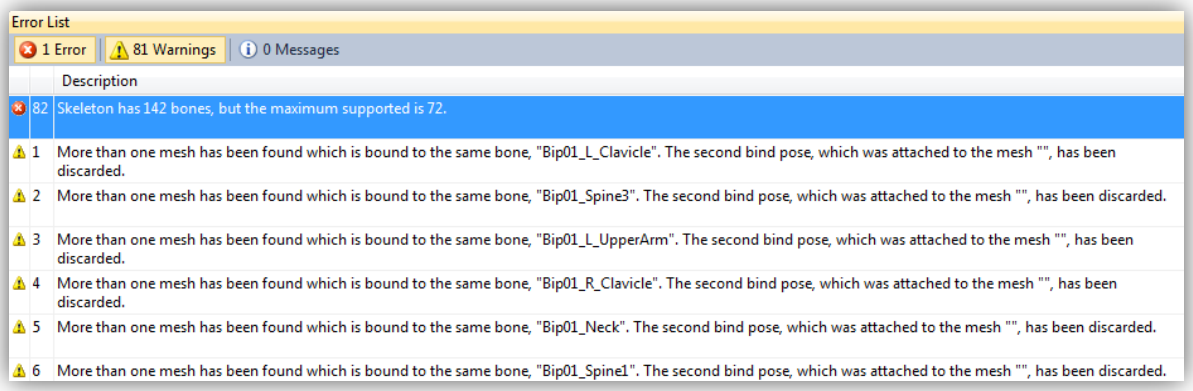
Nakon uspješnog učitavanja modela i pokretanja programa dobivamo animirani ljudski lik prikazan na slici 5.11.

Slika 5.11. Izvođenje programa nakon nadogradnje prvotnog modela

5.4.1. Problemi prilikom učitavanja modela

Prilikom izrade programskog dijela ovog završnog rada utrošeno je jako puno vremena na pronalaženje uzroka brojnih grešaka koje su se javljale prilikom učitavanja modela u program. Iz tog razloga su u ovom poglavlju dokumentirani najčešći problemi i njihovi uzroci.

Na slici 5.12. prikazane su česte greške i upozorenja vezana za odnos mreže i kostiju. Iz opisa greške, odnosno upozorenja, često nije moguće doći do pravog uzroka njihovog javljanja. U ovom slučaju nije riječ o prevelikom broju kostiju kako je opisano, već je ovdje uzrok nedovoljno dobar skinning. Naime, neki vrhovi su najvjerojatnije ostali slobodni i budući da je mreža vezana za nekoliko različitih kostiju program ne uspijeva pravilno učitati model.



Slika 5.12. Greška prevelikog broja kostiju

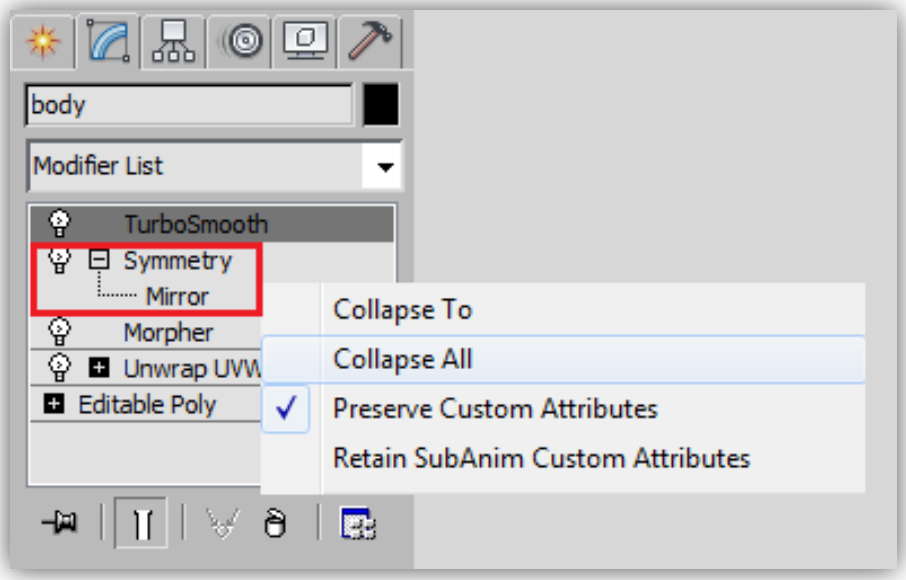
Sljedeći problem ilustriran je slikom 5.13. Napokon smo uspjeli učitati model, no on je potpuno crn. Svi vrhovi mreže su tamni i ne razaznajemo boje. Uzrok ovog problema leži u programskom rješenju primjera *Skinned Model Sample*. Ono ne boja vrhove mreže već pretpostavlja da model sadrži teksture. Ovaj problem je najlakše riješiti da našem model, koji inače ne sadrži teksture, dodamo teksture koje će odgovarati trenutnim bojama u modelu.

Slika 5.13. Problem tamnog modela

Još jedna česta i vrlo uznemirujuća greška je pojavljivanje modela koji se potpuno „razletio“ (slika 5.14.). Uzroci ovakvog problema mogu biti različiti, ali najčešće su vezani za nedovoljno dobar skinning (neki vrhovi ostali su slobodni), te nekompatibilnost određenih efekata programskih alata za modeliranje i XNA radnog okvira. Ako bolje pogledamo sliku 5.14. možemo uočiti da su neki dijelovi mreže ostali nepromijenjeni, dok su drugi dijelovi rastegnuti. Razlog tome su vrhovi mreže koji su ostali slobodni, pa se dio vrhova mreže „kreće“ s kostima, dok drugi dio ostaje na mjestu.

Slika 5.14. Model se razletio

Na primjeru na slici 5.14. ilustriran je još jedan problem. Naime, prikazana je samo desna polovica našeg modela. Uzrok ovog problema leži u činjenici da često programska rješenja, izvozitelji modela (eng. *exporter*) ili radni okviri ne prihvaćaju neke efekte pomoću kojih je modeliran 3D objekt. U ovom slučaju riječ je o korištenju zrcaljenja (eng. *mirror modifier*) prilikom modeliranja. *Mirror modifier* se često koristi prilikom modeliranja simetričnih objekata. On nam omogućava da modeliramo samo jednu polovicu objekta koja se zatim udvostručuje osnom simetrijom. Ovaj problem možemo razriješiti uklanjanjem *mirror modifiera*. Na slici 5.15. prikazan je postupak kojim se *u* 3D Studio Maxu uklanjaju razni efekti pomoću naredbe *collapse all*.



Slika 5.15. Uklanjanje *Symmetry Modifiera*

5.5. XNA okruženje i programsko rješenje *Skinned Model Sample*

Ovo poglavlje donosi detaljniji opis implementacije programskog rješenja primjera *Skinned Model Sample* dostupnog na stranicama MSDN-a. Prije samog objašnjenja krenut ćemo s kratkim uvodom u XNA radno okruženje.

5.5.1. XNA okruženje

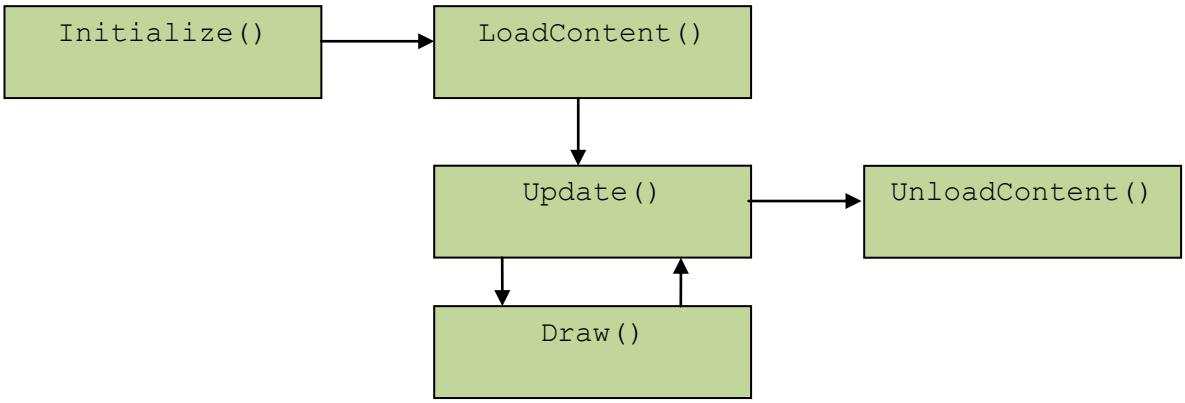
XNA je Microsoftova razvojna platforma kojoj je glavni cilj pružiti korisnicima jednostavan i elegantan pristup razvoju video igara. Svoju popularnost duguje visokoj razini apstrakcije ostvarenoj kroz vješto osmišljene biblioteke namijenjene rješavanju raznih kompleksnih zadataka. Specifičnost XNA radnog okvira leži u mogućnosti izvršavanja istog koda uz minimalne preinake na nekoliko različitih platformi kao što su Windows računala, Xbox 360, Zune, te Windows Phone 7.

Naziv „XNA“ izvorno dolazi od „**X**Box **N**ew **A**rchitecture development“, no nakon izlaska XBox platforme 2005. godine autori su se odlučili našaliti igrom riječi te ovaj radni okvir dobiva puni naziv „**X**NA is **N**ot an **A**cronym“. Ovaj skup alata nastao je 2004. godine, a službeno je objavljen 2 godine kasnije u Kaliforniji. Službeno razvojno okruženje za razvoj XNA aplikacija naziva se *XNA Game Studio* (trenutne verzije 4.0) i oslanja se na poznato razvojno okruženje Visual Studio.

5.5.1.1. Struktura XNA programskog koda

Da bi ostvarili ciklički program kao što su računalne igre potrebno je implementirati 5 osnovnih metoda klase `Game` radnog okvira XNA (slika 5.16.). Prva metoda je `Initialize()` koja je zadužena za inicijalizaciju raznih postavki aplikacije kao što su na primjer veličina i pozicija prozora te pozadinske boje. Metoda `Initialize()` izvršava se pri samom pokretanju programa, a nakon nje izvršava se metoda `LoadContent()`. Ova metoda zadužena je za učitavanje svih resursa potrebnih za pravilno izvođenje naše aplikacije. To mogu biti 3D modeli, teksture, zvukovi, video isječci, fontovi i sl. Po završetku rada programa poziva se metoda `UnloadContent()` koja „odbacuje“ učitane resurse kako bi se rasteretio grafički uređaj sustava. Nakon što je program učitao potrebne resurse ciklički se izvršavaju dvije najvažnije metode klase `Game`: `Update()` i

`Draw()`, čiji su pozivi sinkronizirani s otkucajem sata procesora i ovise o postavkama programa. `Update()` metoda služi za obradu logike i manipulaciju podacima potrebnima za ostvarenje raznih funkcionalnosti aplikacije. U `Update()` metodi se tako uobičajeno nalazi programski kod koji obavlja akcije pokrenute preko kontrola od strane igrača, tj. korisnika aplikacije. Metoda `Draw()` služi za iscrtavanje svih komponenti igre kao što su 3D modeli, teksture, efekti, i sl.

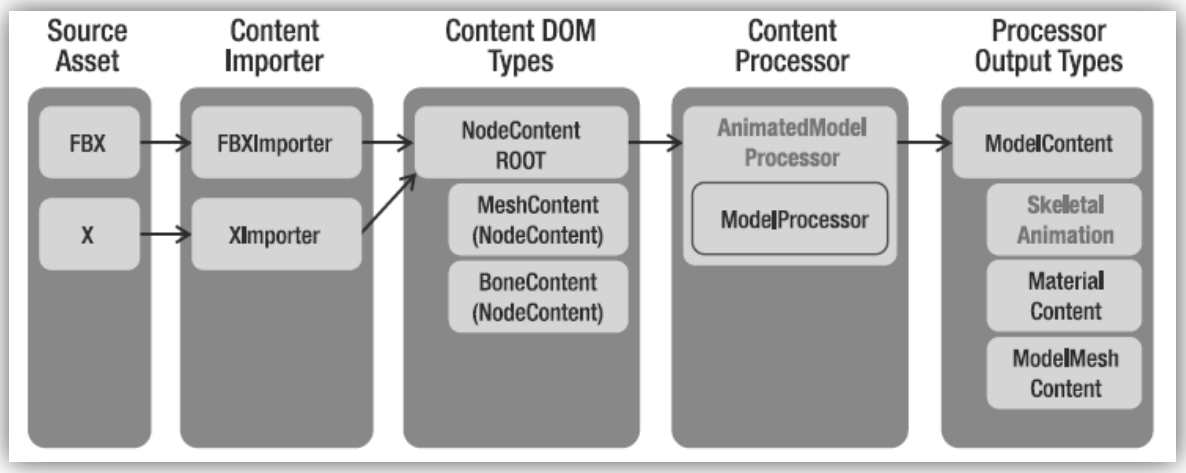


Slika 5.16. Struktura XNA programskog koda

5.5.2. Programsko rješenje *Skinned Model Sample*

Programsko rješenje *Skinned Model Sample* namijenjeno je edukaciji o animaciji modela metodom skeletalne animacije. Ovo rješenje zasniva se na proširenju tzv. sustava *Content Pipeline*, tj. cjevovoda resursa/sadržaja. Ovaj sustav je sastavni dio XNA radnog okvira i dizajniran je kako bi olakšao učitavanje različitog sadržaja u program. To uključuje 3D modele, teksture, animacije, efekte, fontove i dr., koji se prevode u oblik koji XNA razumije. Kako bi se neki resurs uspješno uveo u projekt potrebno ga je spremiti u odgovarajući format koji je podržan od strane jednog od mnogih XNA-ovih uvozitelja (eng. *importer*). Nakon uspješnog prevođenja kroz importer određeni objekt prosljeđuje se tzv. *Content Processoru* koji ih prevodi u oblik koji će se moći koristiti za vrijeme izvođenja programa. U slučaju da standardni procesori ne mogu obraditi određeni tip podataka moguće je napisati vlastiti *Content Processor*. U našem slučaju problem se pojavljuje prilikom učitavanja informacija o obavljenom skinningu 3D modela. Budući da standardni procesori ne omogućavaju obradu ovakve vrste informacija, gotovo programsko rješenje koje koristimo proširuje cjevovod sadržaja (Content Pipeline) vlastitim procesorom

nazvanim `SkinnedModelProcessor`. Ovako prošireni cjevovod koji nam omogućava učitavanje modela s kosturom u formatu X i FBX prikazan je na sljedećoj slici 5.17.



Slika 5.17. Content Pipeline

`SkinnedModelProcessor` je klasa koja nastaje nasljeđivanjem klase `ModelProcessor` i nadjačavanjem nekoliko njenih metoda. Glavna metoda klase `ModelProcessor` koja se ovdje nadjačava jest metoda `Process()`. Njome se prevađa objekt tipa `NodeContent`, koji sadrži podatke o mrežama, kosturu, i animacijama, u objekt tipa `ModelContent`. Metoda `Process()` najprije pronalazi kostur, tj. njegovu korijensku kost, metodom `MeshHelper.FindSkeleton(NodeContent input)`, a nakon toga pozivom metode `MeshHelper.FlattenSkeleton(BoneContent skeleton)` pretvaramo kostur iz strukture stabla u listu dubinskim pretraživanjem. Dio koda koji ostvaruje upravo opisanu funkcionalnost nalazi se u nastavku.

```
public override ModelContent Process(NodeContent input,
                                      ContentProcessorContext context)
{
    ValidateMesh(input, context, null);
    BoneContent skeleton = MeshHelper.FindSkeleton(input);
    if (skeleton == null)
       throw new InvalidContentException("Skeleleton not found!");
    FlattenTransforms(input, skeleton);
    IList<BoneContent> bones = MeshHelper.FlattenSkeleton(skeleton);

    ...
```

U sljedećom koraku za svaku kost u listi `bones` spremaju se podaci o položaju u kosturu kao i indeksi roditeljske kosti u odgovarajuće liste.

```
List<Matrix> bindPose = new List<Matrix>();
List<Matrix> inverseBindPose = new List<Matrix>();
List<int> skeletonHierarchy = new List<int>();

foreach (BoneContent bone in bones)
{
    bindPose.Add(bone.Transform);
    inverseBindPose.Add(Matrix.Invert(bone.AbsoluteTransform));
    skeletonHierarchy.Add(bones.IndexOf(bone.Parent as BoneContent));
}
   ...
```

Nakon što je uspješno učitan kostur modela, poziva se metoda `ProcessAnimation()` (više o njoj u sljedećem odlomku) koja pretvara podatke o animacijama modela u odgovarajući oblik koji se koristi u ovom programskom rješenju, a implementiran je klasom `AnimationClip`. Animacije se spremaju u *hash* strukturu, te im se pristupa imenom koji se zadaje animaciji prilikom izvoza modela. Nakon što su dohvaćeni i obrađeni podaci o animacijama stvara se novi objekt tipa `ModelContent` kojem je pridodana oznaka (*tag*) u obliku objekta klase `SkinningData` koji sadrži sve podatke o kostima i animacijama koje smo upravo učitali i obradili.

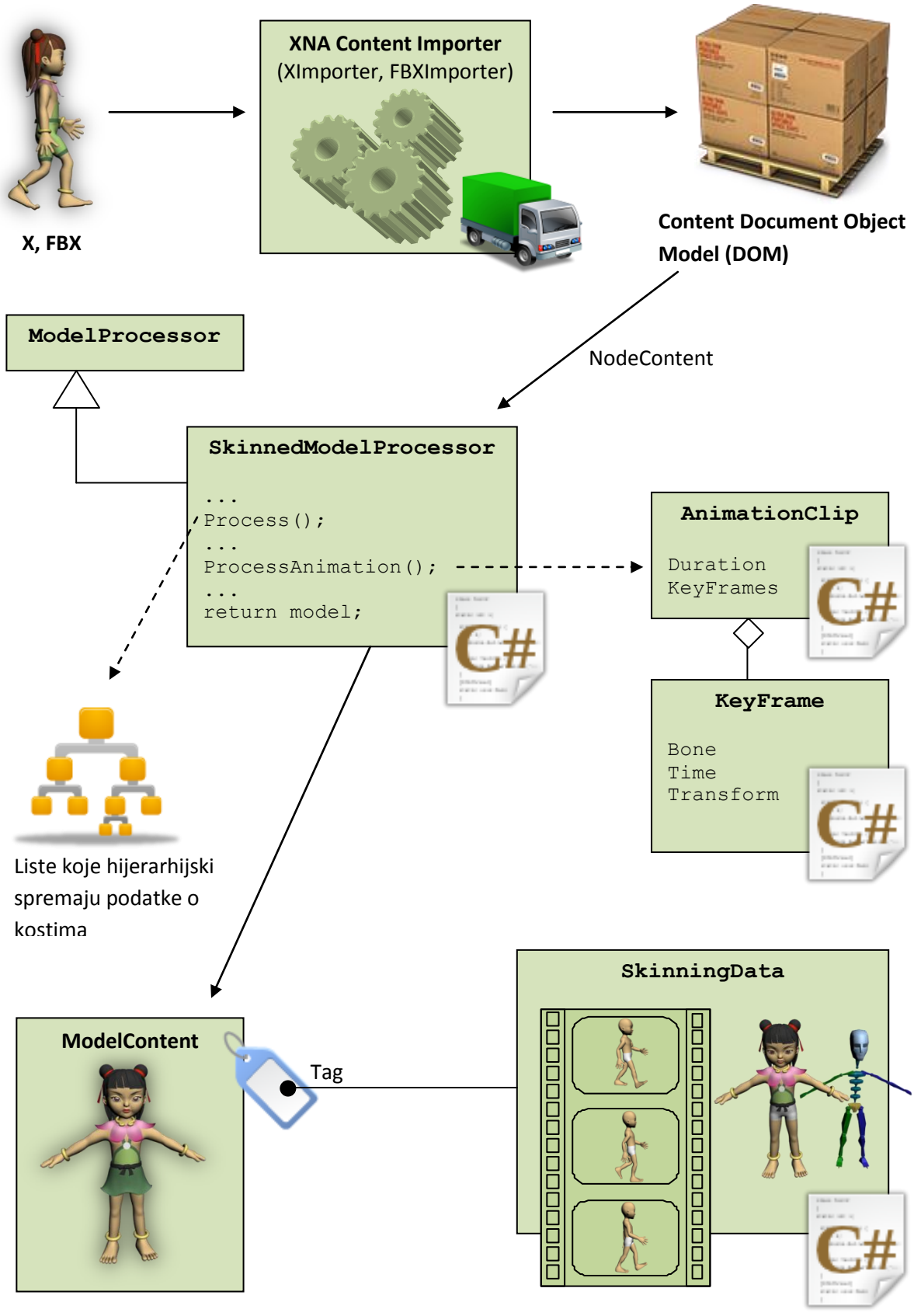
```
    Dictionary<string, AnimationClip> animationClips;
    animationClips = ProcessAnimations(skeleton.Animations, bones);

    ModelContent model = base.Process(input, context);
    model.Tag = new SkinningData(animationClips, bindPose,
                                 inverseBindPose, skeletonHierarchy);

    return model;
}
```

Prethodno spomenuta metoda `ProcessAnimation()` pretvara podatke o animaciji (obično su to položaji kostiju u pojedinim trenutcima) u objekte klase `AnimationClip`. Objekti klase `AnimationClip` sadrže informaciju o trajanju animacije te listu ključnih trenutaka. Ključni trenutci su implementirani klasom `KeyFrame` čiji objekti sadrže matricu transformacije nad određenom kosti u određenom trenutku.

Budući da je prethodno opisano programsko rješenje vrlo kompleksno te zahtjeva puno vremena za razumijevanje, sljedeća shema ga slikovitije opisuje.



Shema 2. Programsko rješenje *Skinned Model Sample*

5.5.3. Proširenje programskog rješenja *Skinned Model Sample*

Rezultat ovog završnog rada je programski produkt koji koristi programsko rješenje opisano u prethodnom poglavlju 5.5.2.

Krenimo od implementacije metode `LoadContent()` glavne klase `SkinningSampleGame`. Kao što je već ranije spomenuto, ova metoda zadužena je za učitavanje programu potrebnih sadržaja kao što su modeli, teksture, zvukovi, fontovi i sl. Ovom metodom učitat ćemo najprije naš animirani model. To ćemo učiniti jednostavnom naredbom `Content.Load<Model>("skinned_model")` gdje je `skinned_model` ime našeg animiranog modela u X formatu. Budući da želimo da učitavanje izvrši novonastali `SkinnedModelProcessor` u svojstvima modela `skinned_model` postavit ćemo ga kao *Content Processor* u polju *XNA Framework Content Pipeline*. Osim animiranog modela učitat ćemo još model scene za koji je nam je dovoljan standardni *XNA Framework Model Processor*. Nakon toga pristupamo podacima o animiranom modelu preko oznake `currentModel.Tag`, te preko njih dohvaćamo određenu animaciju koja je tipa `AnimationClip`. U našem slučaju za inicijalnu animaciju ćemo zadati animaciju stajanja. Stvaramo novi objekt tipa `AnimationPlayer` koji na temelju podataka o ključnim trenucima animacije izvršava potrebne transformacije.

```
protected override void LoadContent()
{
    currentModel = Content.Load<Model>("skinned_model");
    scena = Content.Load<Model>("scena");

    skinningData = currentModel.Tag as SkinningData;

    if (skinningData == null)
       throw new InvalidOperationException
          ("This model does not contain a SkinningData tag.");

    animationPlayer = new AnimationPlayer(skinningData);
    clip = skinningData.AnimationClips["stajanje"];
    animationPlayer.StartClip(clip);
    activeAnimation = "stajanje";
    ...
```

U metodi `LoadContent()` nalazi se još dio programskog koda kojim se dohvaćaju teksture i fontovi kojima je ostvareno HUD sučelje.

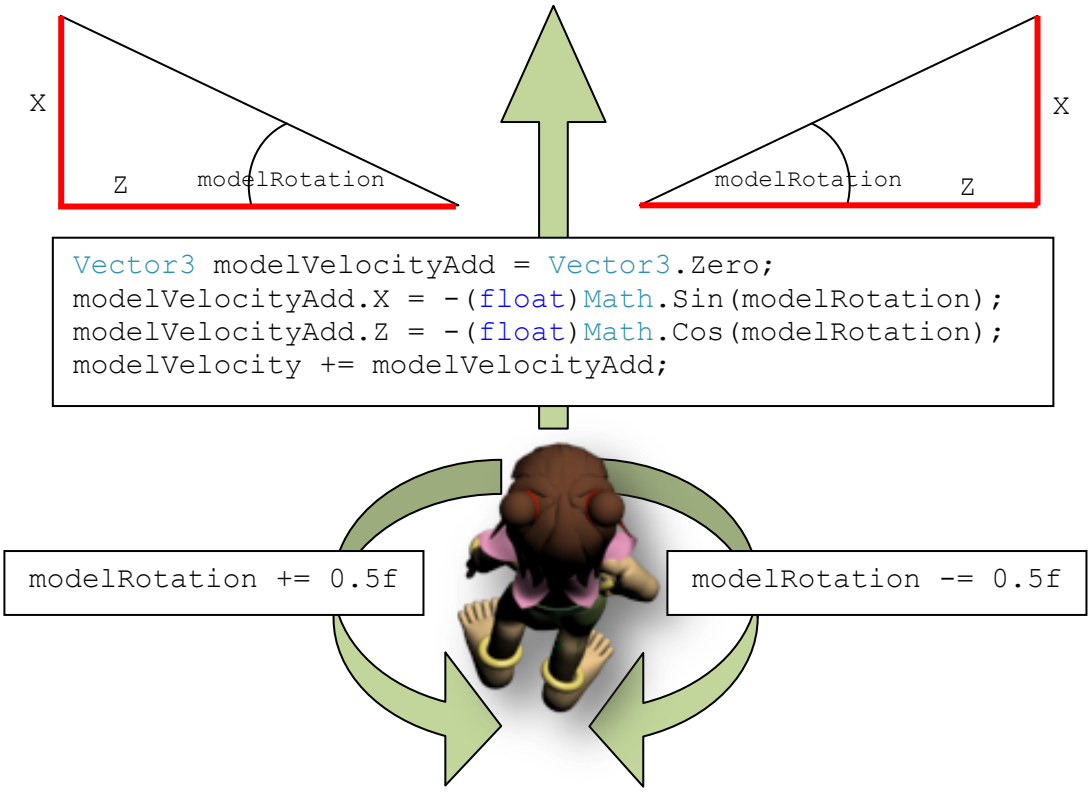
Ako u ovom trenutku pokrenemo program, na učitanoj sceni dobit ćemo model animiran beskonačnom animacijom stajanja. Ono što mi za početak želimo jest ostvariti interaktivnu promjenu animacija. Primjerice, pritiskom tipke W, koja će nam kasnije služiti za kretanje prema naprijed, želimo pokrenuti animaciju hoda na mjestu. Da bi to ostvarili moramo uvesti promjene u metodi `Update()`. Prilikom pritiska tipke W (eng. *on key down*) pokrenut ćemo animaciju hoda na mjestu, a prilikom otpuštanja tipke W (eng. *on key up*) vratit ćemo animaciju stajanja. U tu svrhu koristimo pomoćne varijable `oldState` i `newState` tipa `KeyboardState` kojima pamtimo što se dogodilo u prethodnoj interakciji korisnika i programa preko tipkovnice. Primjerice, u slučaju da novo stanje sadrži informaciju da tipka W nije pritisnuta, a staro stanje informaciju da ona je pritisnuta, zaključujemo da je tipka W upravo otpuštena i pokrećemo animaciju stajanja.

```
KeyboardState newState = Keyboard.GetState();

if (newState.IsKeyDown(Keys.W))
{
    if (!oldState.IsKeyDown(Keys.W))
    {
        clip = skinningData.AnimationClips["hodanje"];
        animationPlayer.StartClip(clip);
        activeAnimation = "hodanje";
    }
}
else if (oldState.IsKeyDown(Keys.W))
{
    clip = skinningData.AnimationClips["stajanje"];
    animationPlayer.StartClip(clip);
    activeAnimation = "stajanje";
}

oldState = newState;
```

Sljedeće što želimo ostvariti je translacija animiranog modela po sceni. Želimo dobiti dojam hoda kombinacijom translacije modela i animacije hoda na mjestu. Model ćemo kretati po sceni tipkama W – naprijed, A – lijevo, D – desno. Da bi uspješno ostvarili translaciju trebaju nam varijable kojima ćemo pamtiti trenutnu poziciju modela te izvršenu rotaciju i translaciju.



Shema 3. Ostvarenje kretanja modela po sceni

Shema 3. prikazuje rješenje kojim je ostvareno kretanje modela. Varijabla `modelRotation` predstavlja kut za koji zakrećemo model. Njena vrijednost povećava se pritiskom tipke A, tj. smanjuje pritiskom tipke D. Varijabla `modelVelocity` je vektor brzine našeg modela. U svakom pozivu metode `Update()` mijenjaju se vrijednosti njene X i Z koordinate izračunom sinusa, tj. kosinusa kuta `modelRotation`. Na taj način dobivamo odgovarajući pomak u smjeru X i Z osi. Svakim pozivom metode `Update()` dodajemo vektor `modelVelocity` vektoru trenutne pozicije modela `modelPosition`. Na temelju ovog vektora stvaramo matricu transformacije (rotacija i translacija) te ju šaljemo kao argument metodi `Update()` klase `AnimationPlayer` koja na temelju matrice transformacije vrši odgovarajuće promjene u animaciji modela. Također ostvarujemo „slabljenje“ vektora brzine množenjem s 0.005.

```
modelPosition -= modelVelocity;
modelVelocity *= 0.005f;
animationPlayer.Update(gameTime.ElapsedGameTime, true,
                       Matrix.CreateRotationY(modelRotation) *
                       Matrix.CreateTranslation(modelPosition));
```

Ovime je ostvarena najvažnija funkcionalnost našeg programskog produkta koji je rezultat završnog rada. Dodatno, ostvareno je jednostavno HUD sučelje koje korisniku daje upute za korištenje programa, te pruža informacije o trenutnoj animaciji i izgledu kostura modela. Ovime završavamo s praktičnim djelom završnog rada. Ostvarili smo aplikaciju koja interaktivno pokreće i animira trodimenzionalni model koristeći metode skeletalne animacije (slika 5.18.).

Slika 5.18. Završni izgled programskog produkta

6. Zaključak

Izrada kvalitetne, brze i uvjerljive animacije korištenjem metoda skeletalnog animiranja u praksi predstavlja vrlo zahtjevan i kompliciran proces koji iziskuje dosta vremena, ali i teoretskog znanja. Ovaj proces možemo donekle olakšati dobrim programskim alatima za modeliranje i animaciju, te radnim okvirima visoke razine apstrakcije kao što je Microsoft XNA Framework. Ovakvi radni okviri pružaju dobru podršku skeletalnom animiranju te jednom kada se shvate osnovni principi učitavanja animiranog modela i obrada podataka o izvršenom skinningu, ovaj proces postaje jednostavniji i brži. Najveći dio posla pri izradi aplikacija kao što je ova razvijena završnim radom predstavlja priprema modela (skinning, animacija, postavljanje tekstura i dr.). Ukoliko želimo postići glatku i uvjerljivu animaciju morat ćemo potrošiti dodatnu količinu vremena da ovaj dio posla obavimo što kvalitetnije. U tu svrhu vrlo dobrim se pokazao programski alat 3D Studio Max koji pruža kvalitetnu podršku skeletalnom animiranju.

Iako postupci skeletalnog animiranja zahtijevaju dosta vremena, oni se vrlo često koriste u video igrama i modernim animiranim filmovima. Prednosti ovakvih animacija je mnogo. Jedna od najvažnijih je činjenica da programski kod možemo osloboditi od kompliciranih i često vrlo zahtjevnih matematičkih transformacija kojima je ostvarena animacija, te jednostavno koristimo gotovu animaciju koju smo izradili u nekom drugom alatu koji pruža bolje sučelje za animaciju modela. Osim toga vrlo je jednostavno pristupiti bilo kojem dijelu ljudskog tijela preko odgovarajuće kosti koju onda možemo po volji transformirati u programskom kodu.

Programski dio završnog rada zahtijevao je puno posla. Iako je na kraju njime ostvarena ciljna funkcionalnost, ostalo je još dosta mjesta za poboljšanje i daljnji rad. Primjerice, animacije je moguće bolje i uvjerljivije izraditi, te bi skinning mogao biti precizniji kako se na nekim mjestima mreža ne bi nepravilno savijala oko kostiju.

Ovim radom stekao sam vrlo dobar uvid u mogućnosti računalne animacije kao i u kompleksnost procesa kojima se one ostvaruju. Tema me jako zainteresirala i nadam se daljnjem radu u smjeru računalne grafike i animacije.

- Wikipedia: Animation, veljača 2011., *Animation,* http://en.wikipedia.org/wiki/Animation, 5. ožujka 2011.
- Wikipedia: Computer animation, veljača 2011., *Computer animation,* http://en.wikipedia.org/wiki/Computer_animation, 5. ožujka 2011.
- Wikipedia: Traditional animation, veljača 2011., *Traditional animation,* http://en.wikipedia.org/wiki/Traditional_animation, 5. ožujka 2011.
- Wikipedia: History of animation, veljača 2011., *History of animation,* http://en.wikipedia.org/wiki/History_of_animation, 8. ožujka 2011.
- Željka Mihajlović, Predavanje iz predmeta računalna animacija, 2007., *Unaprijedna i inverzna kinematika*, http://www.zemris.fer.hr/predmeti/ra/predavanja/4_kinemat.pdf, 15. ožujka 2011.
- Željka Mihajlović, Predavanje iz predmeta računalna grafika, 2007., *Grafičke primitive i transformacije,* http://www.zemris.fer.hr/predmeti/irg/predavanja/3_primitive.pdf, 10. veljače 2011.
- Wikipedia: Inverse kinematics, ožujak 2011., *Inverse kinematics,* http://en.wikipedia.org/wiki/Inverse_kinematics, 15. ožujka 2011.
- M. Bousquet, M. McCarthy, 3ds Max Animation with Biped, New Riders, 2006.
- A. S. Lobao, B. Evangelista, J. A. Leal de Farias, R. Grootjans, Beginning XNA 3.0 Game Programming: From Novice to Professional, Apress, 2009.
- XNA Creators Club: Skinned Model Sample, veljača 2007, *Skinned Model*, http://create.msdn.com/en-US/education/catalog/sample/skinned_model, 13. ožujka 2011.

Sažetak

Ovaj završni rad daje uvid u najčešće metode računalne animacije s naglaskom na postupke animacije ljudskih likova. Rad također prati razvoj programskog rješenja, tehnološke demonstracije skeletalne animacije modela, koja je njegov praktični rezultat.

U uvodu je dan kratak uvid u povijest i razvoj animacijskih tehnika. Slijedi upoznavanje s poznatijim postupcima 2D i 3D računalne animacije s posebnim naglaskom na metodu skeletalne animacije. Dana je i matematička pozadina ove metode koja se zasniva na inverznoj i unaprijednoj kinematici.

Završni rad prateći razvoj programskog produkta koji demonstrira postupke animacije ljudskih likova upoznaje čitatelja s mogućnostima Autodesk 3D Studio Maxa, programskog alata za 3D modeliranje i animaciju. Objašnjeni su osnovni principi pripreme modela za animaciju tehnikom skeletalne animacije kao što je primjerice skinning. Implementacija programskog rješenja koristi Microsoftov XNA radni okvir namijenjen razvoju računalnih igara, pa su u radu objašnjene i osnovne klase i metode, kao i struktura koda uobičajenog XNA programa.

KLJUČNE RIJEČI: animacija ljudskih likova, skeletalna animacija, unaprijedna kinematika, inverzna kinematika, skinning, Autodesk 3D Studio Max, Microsoft XNA Framework

Summary

This baccalaureus graduate thesis provides insight into the most common methods of computer animation with emphasis on animation procedures of human figures. We also describe the development of software solutions, technology demonstration of skeletal animation, which is its practical result.

The introduction presents a brief overview of the history and development of animation techniques. The following section introduces better known methods of 2D and 3D computer animation with special emphasis on the skeletal animation method. We also explore mathematical background of this method which is based on inverse and forward kinematics.

Following the development of software product which demonstrates animation procedures of human figures we present possibilities of Autodesk 3D Studio Max, software tool for 3D modeling and animation. We explain basic principles of preparing models for animation using skeletal animation techniques, such as skinning. Implementation of software solution is based on Microsoft's XNA framework, which is designed for computer game development. We present its basic classes and methods, as well as the code structure of usual XNA program.

KEY WORDS: human character animation, skeletal animation, forward kinematics, inverse kinematics, skinning, Autodesk 3D Studio Max, Microsoft XNA Framework