SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1311

Izrada kinematičkog modela vlaka

Danijel Vicković

Zagreb, lipanj 2010.

Zahvaljujem kolegi i prijatelju Tinu Englmanu na svoj pomoći koju mi je pružio tijekom pisanja ovoga rada, kolegi Ivanu Kravarščanu na pomoći oko spremnika, profesorici iz fizike dr. sc. Ivani Capan na pojašnjavanju nekih osnovnih zakona fizike, te mentorici prof. dr. sc. Željki Mihajlović na uputama i smjernicama.

Sadržaj

Uvod

Fizikalni zakoni se vrlo često moraju implementirati unutar virtualnog okruženja, bila to virtualna stvarnost, računalna igra ili nešto treće. Zadatak ovog rada je proučiti osnovne kinematičke strukture i pomoću kinematičkog lanca izgraditi kinematički model vlaka. Za simulaciju primijenit će se model vlaka u svemiru koji se giba u 3D prostoru.

Kako bi se napravila programska aplikacija koja simulira gibanje kinematičkog lanca, prvo je potrebno upoznati se sa samim pojmom kinematičkog lanca, njegovim karakteristikama i vrstama, te se taj dio objašnjava u poglavlju (1). U sljedećem poglavlju, opisani su korišteni alati. Za izradu rada potrebne su četiri vrste alata: alat za 3D modeliranje, alat za 2D grafiku pomoću kojeg se izrađuju teksture za 3D objekte, alat pomoću kojeg povezujemo grafički dio i kod, te programsko okruženje u kojemu će se pisati kod. Nakon toga se, u poglavlju (3), pristupa osnovnim zahtjevima koje sustav mora zadovoljiti, te opisivanju modela kinematičkog lanca koji će se koristiti u radu. Poglavlje (4) opisuje izradu programske potpore, od dizajniranja 3D modela do fizičkih zakona koji su uklopljeni u ovu simulaciju. Zadnje poglavlje prikazuje dobivene rezultate i ponašanje programa.

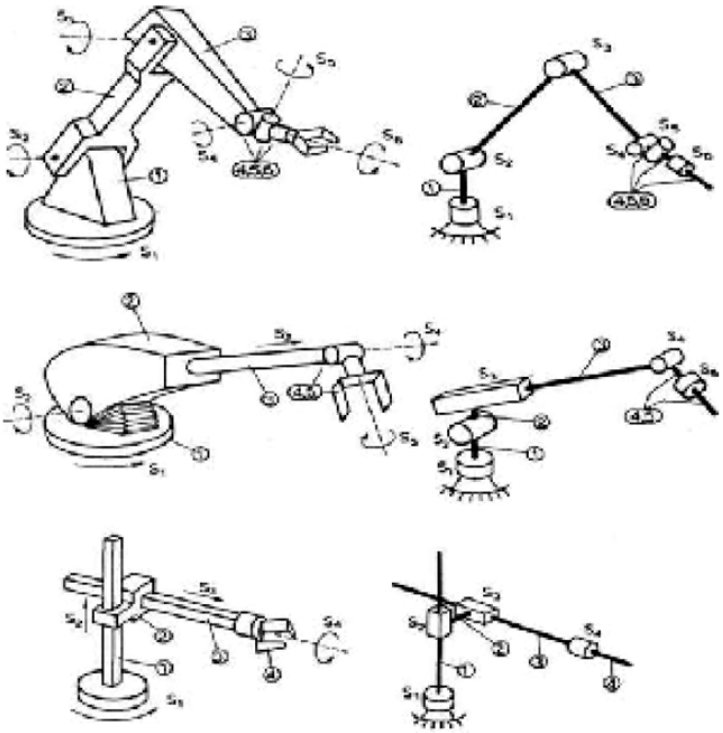
1. Kinematički lanac

Kako bi napravili kinematički model vlaka pomoću kinematičkog lanca, najprije je potrebno definirati što je kinematika te što je kinematički lanac. Kinematika je grana mehanike koja se bavi proučavanjem gibanja tijela ne uzimajući u obzir sile pod čijim se djelovanjem to gibanje zbiva. Za to je zadužena dinamika koja se izričito bavi promatranjem sila na tijelo u gibanju.

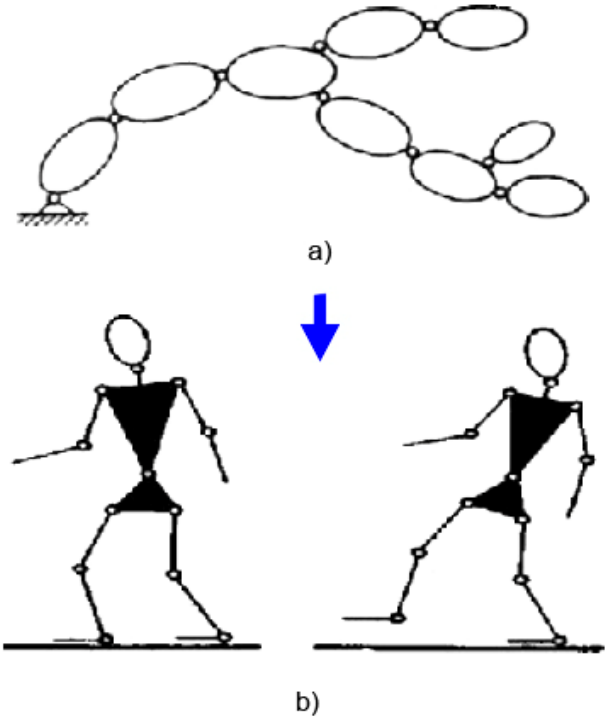
Kinematički lanac je niz kinematičkih članaka koji su međusobno povezani zglobovima. Dva međusobno povezana elementa lanca koji omogućuju gibanje nazivaju se kinematičkim parom. Elementi od kojih je lanac sastavljen još se nazivaju i karikama ili segmentima. Kinematički lanci najviše se koriste u robotici, te su se prvi put i pojavili u industriji.

1.1. Vrste kinematičkih lanaca

Kinematički lanci mogu se podijeliti na jednostavne i složene (razgranate). Kod jednostavnih lanaca, svaki članak je povezan s točno dva članka (jedan članak prije i jedan članak poslije), osim početnog i završnog članka koji su povezani sa samo jednim člankom. Početni članak mora biti vezan za bazu dok krajnji može biti slobodan ili također pričvršćen za bazu. Na slici (Slika 1.1) prikazani su primjeri jednostavnog kinematičkog lanca. Kod složenog kinematičkog lanca, za razliku od jednostavnog, na jedan članak može biti povezano više od dva članaka, što se može i vidjeti na slici (Slika 1.2 a). Na slici (Slika 1.2 b) prikazan je primjer primjene složenog kinematičkog lanca.



Slika 1.1 Jednostavni kinematički lanac.

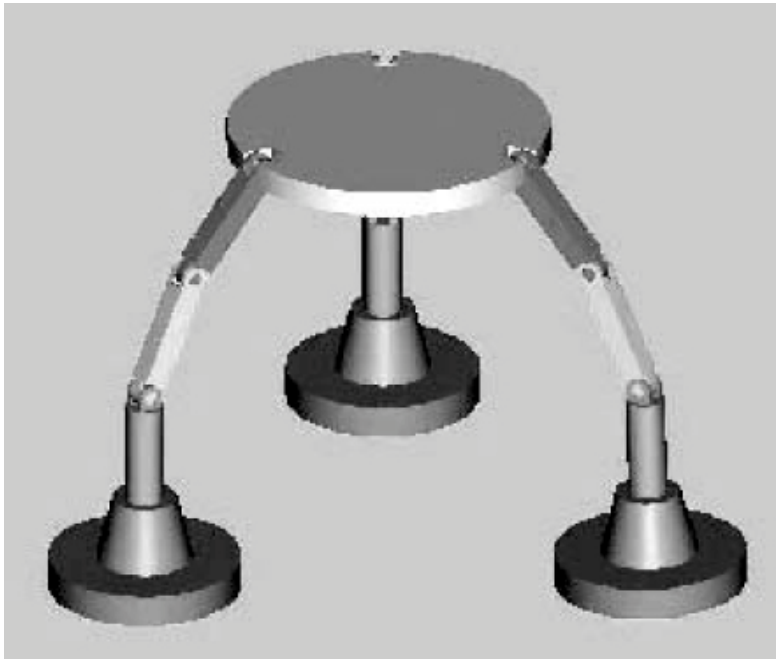


Slika 1.2 a) Složeni kinematički lanac b) Shema antropomorfnog hodajućeg robota.

Cilj ovog rada je ostvariti model vlaka koristeći jednostavni kinematički lanac. Kinematički lanci se također mogu podijeliti na otvorene i zatvorene. Kod otvorenih, jedan kraj lanca je pričvršćen za bazu (nepomični objekt: pod, zid...), dok je drugi kraj slobodan. U otvorenim lancima ne postoji zatvoreni niz, odnosno iz jedne točke lanca u drugu može se doći samo jednim putem. Primjer otvorenog kinematičkog lanca može se vidjeti na slici (Slika 1.3).

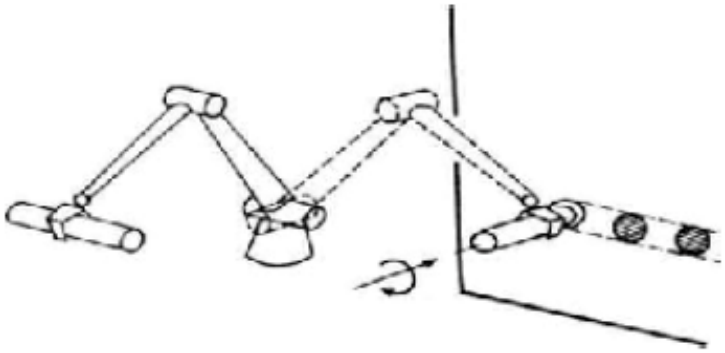
Slika 1.3 Otvoreni kinematički lanac.

Za razliku od otvorenih, zatvoreni lanac je jedan zatvoreni niz segmenata, odnosno iz jedne točke lanca u drugu može se doći barem na dva načina, te se primjer ove vrste lanca može vidjeti na slici (Slika 1.4). Zatvoreni lanci su prilično kompliciraniji od otvorenih, i što se tiče izvedbe i analize. Za analizu je potrebno uzeti u obzir statička svojstva te utjecaje ostalih članaka itd.



Slika 1.4 Zatvoreni kinematički lanac.

Na slici (Slika 1.5) prikazan je primjer robotskog mehanizma kod kojeg kinematički lanac mijenja svoju strukturu iz otvorenog u zatvoreni i obratno.



Slika 1.5 Manipulacijski robot u zadatku montaže.

1.2. Stupnjevi slobode kinematičkog lanca

Stupnjevi slobode *(*eng. *degree of freedom; d.o.f)* kinematičkog lanaca dijele se na: translacijske i rotacijske. Translacija je takvo gibanje kada se dva dijela robota, koja su međusobno povezana, gibaju tako da jedan ulazi u drugi ili pomiče okomito na vrh prethodnog elementa. Rotacija je takvo gibanje kada se dijelovi zgloba okreću oko zajedničkog dijela. U 2D prostoru najveći mogući broj stupnjeva slobode koje zglob može imati je 3, dok je u 3D prostoru 6.

Bilo koji zglob u 3D prostoru može imati najviše 3 translacijska i 3 rotacijska stupnja slobode, što sveukupno čini 6 stupnjeva slobode. U ovom radu kinematički lanac će biti postavljen u 3D prostoru. U tablici (Tabela 1.1) prikazano je svih 5 klasa zglobova, odnosno kinematičkih parova, kojih ima pet. Također je prikazano koliko stupnjeva slobode imaju.

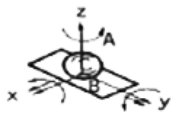
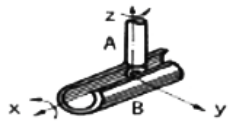
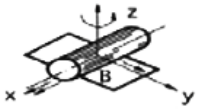
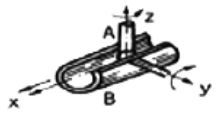
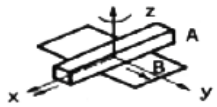
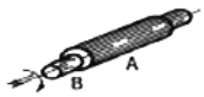
class of task	nr. of links	nr. of d.o.f.	TYPES OF PAIRS								
			I			II			III		
			nr. of movem.	rot.	lin.						
			allowed	3	2						
			restricted	0	1						
I	1	5									
			nr. of movem.	rot.	lin.	nr. of movem.	rot.	lin.			
			allowed	3	1	allowed	2	2			
			restricted	0	2	restricted	1	1			
II	2	4									
			nr. of movem.	rot.	lin.	nr. of movem.	rot.	lin.	nr. of movem.	rot.	lin.
			allowed	3	0	allowed	2	1	allowed	1	2
			restricted	0	3	restricted	1	2	restricted	2	1
III	3	3									
			nr. of movem.	rot.	lin.	nr. of movem.	rot.	lin.			
			allowed	2	0	allowed	1	1			
			restricted	1	3	restricted	2	2			
IV	4	2									
			nr. of movem.	rot.	lin.	nr. of movem.	rot.	lin.			
			allowed	1	0	allowed	0	1			
			restricted	2	3	restricted	3	2			
V	5	1									

Tabela 1.1 Klase zglobova.

1.3. Direktna i inverzna kinematika

Robotski manipulator može se promatrati kao lanac krutih članaka koji su povezani zglobovima. Da bi se mogla odrediti točka u prostoru u koju je postavljen vrh alata s obzirom na bazu robota, treba odrediti vezu između zglobova i pozicije odnosno orijentacije alata. Rješavanje tog problema nalazi se u direktnoj kinematici.

Dakle, ako je zadan vektor varijabli zglobova robotskog manipulatora, tada treba odrediti položaj i orijentaciju alata u odnosu prema koordinatnom sustavu pridruženom bazi. Rješavanje direktnog kinematičkog problema nije previše složeno. Svodi se na množenje matrica i rješenje je jedinstveno.

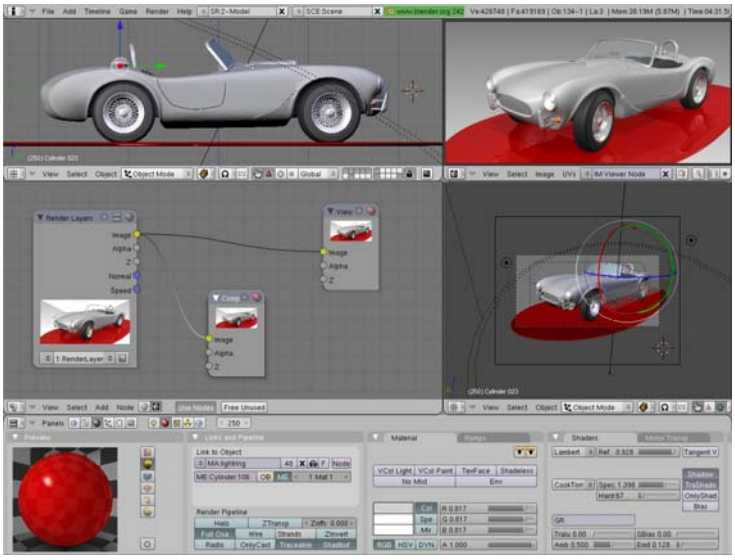
Princip inverzne kinematike inverzan je od direktne. Odnosno, za zadanu točku u prostoru, potrebno je naći pripadajuće vrijednosti pozicije i orijentacije zglobova. Problem rješavanja inverznog kinematičkog problema složeniji je od rješavanja direktnog jer ne postoji jedinstven postupak rješavanja kao što postoji u rješavanju direktnog kinematičkog problema. Iz tog razloga, inverzni kinematički problem se rješava posebno za svaku grupu sličnih robota. Dva najčešća postupka rješavanja su iterativni numerički postupci i analitička rješenja. Jedan od problema prilikom rješavanje inverzne kinematike (kada postoji rješenje inverznog kinematičkog problema) je da dobiveno rješenje često nije jedinstveno. Principi direktne i inverzne kinematike se, gotovo isto kao i u robotici, primjenjuju u računalnoj animaciji.

2. Opis alata

U ovom poglavlju dan je kratak opis alata koji su se koristili pri izradi kinematičkog modela vlaka. U kasnijim poglavljima biti će opisana izrada praktičnog dijela rada.

2.1. Blender

Postoji mnogo alata za 3D modeliranje od kojih su među najkorištenijima 3D Studio, Maya i Blender. No, Blender nije vlasnički format, za razliku od ostala dva, te je potpuno besplatan. Međutim, to ga ne čini inferiornijim od velikih i skupih alata, već naprotiv, vrlo je pristupačan te se zbog toga širi enormnom brzinom. Ima podržane osnovne funkcije koje svaki alat za 3D modeliranje mora imati: modeliranje, teksturiranje, UV razmotavanje (eng. *UV unwrap*), animiranje, iscrtavanje (eng. *rendering*), sustave čestica (eng. *particle systems*). Uz to, podržava i napredne simulacije krutih tijela (eng. *rigid body*), tekućine, tkanine, modeliranje pomoću modifikatora, pisanje skripti u Phytonu itd. Počevši kao mali amaterski alat, razvio se u ozbiljnu aplikaciju koja se koristi za izradu vrlo kompliciranih objekata i animacija. Iako slovi kao program kojeg je teško naučiti, oni koji su se navikli na rad u njemu mogu koristiti prečice na tipkovnici za skoro svaku naredbu i tako vrlo brzo kreirati modele. Pod okriljem GNU General Public License, ovaj besplatni alat dostupan je na operacijskim sustavima Linux, Mac OS X i Microsoft Windows.



Slika 2.1 Blender razvojno okruženje.

2.2. Gimp

Gimp (GNU Image manipulator program) je besplatni program za obradu slike. Najčešću primjenu nalazi kao alat za obradu i retuširanje slike. Osnovne funkcije obrade slike su: promjena veličine slike, rezanje, spajanje više slika te pretvaranje slika iz jednog formata u drugi. Također, može se koristiti za izradu jednostavnih GIF animacija. GIMP se najviše koristi za amaterske obrade slike te velik dio svoje popularnosti zahvaljuje činjenici da je *open source* program. No pokazao se vrlo dobrim i u profesionalnom svijetu, a to dokazuju i česte recenzije i kritike, u kojima ovaj program neki uspoređuju i s popularnim Adobeovim alatom Photoshop. Iako GIMP smatran kao dobra zamjena za velike komercijalne programe, sve se više i više prihvaća na profesionalnom tržištu. U početku je bio dostupan samo na operacijskom sustavu Unix, no od tada su se razvile verzije i za Microsoft Window i Mac OS X.

Slika 2.2 Alat za obradu slike – GIMP.

2.3. DirectX

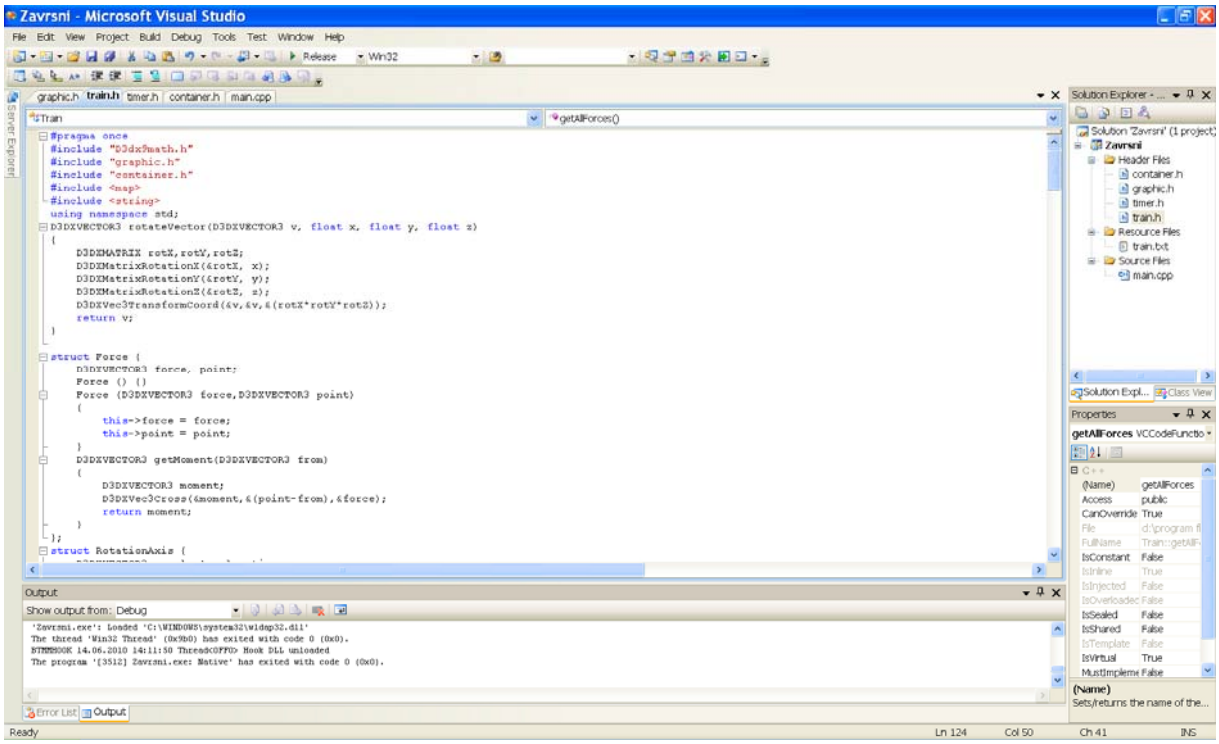
DirectX je Microsoftovo programsko sučelje (eng. *Application programing interface; API*) koje se koristi za izradu višemedijskih aplikacija, videa, te najviše za izradu računalnih igara. U samom DirectXu sadržano je još mnogo API-a i svi započinju riječju Direct: Direct3D, DirectDraw, DirectMusic, DirectPlay, DirectSound itd.

Programska sučelja se općenito mogu podijeliti na programska sučelja visoke razine, poput Ogre, i na programska sučelja niske razine, u koje spada OpenGL i DirectX. Programska sučelja niske razine pružaju vrlo izravnu vezu s grafičkim protočnim sustavom, te najveću fleksibilnost u korištenju svih njegovih funkcija bez ograničenja. Na višim razinama jednostavnije je programiranje predviđenih aplikacija kroz korištenje ponuđenih struktura, što omogućuje vrlo brz razvoj aplikacije, no to ponekad može djelovati kao ograničenje. Direct3D se vrlo često koristi u razvoju računalnih igara za Microsoft Windows, Xbox i Xbox 360. Pošto je Direct3D najraširenija komponenta DirectX paketa, često se ta dva pojma koriste kao sinonimi. DirectX SDK (eng. *Software development kit*) sadrži biblioteke u binarnom obliku zajedno s pripadajućom dokumentacijom i zaglavljima (eng. *headers*) koja se mogu koristiti prilikom programiranja. Na slici (Slika 2.3) može se vidjeti razlika prilikom korištenja dvije različite verzije DirectXa.

Slika 2.3 DirectX 9.0.

2.4. Microsoft Visual Studio

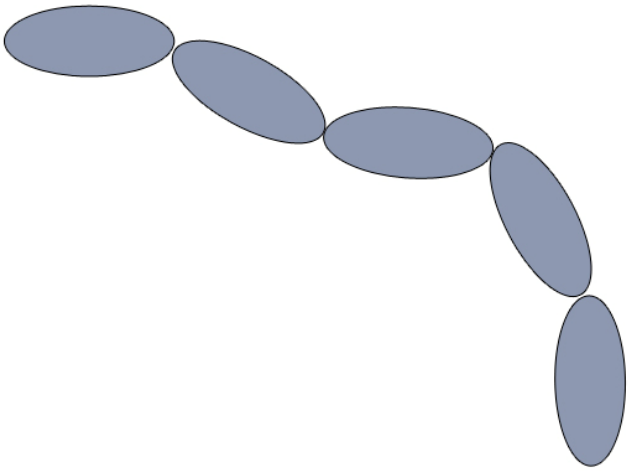
Visual Studio (VS) je razvojno okruženje (eng. *Integrated development enviroment; IDE*) koje se koristi za razvoj konzolnih aplikacija i aplikacija s grafičkim sučeljem. Podržava različite programske jezike, i u kontekstu uređivača koda (eng. *Code editor*), i u kontekstu *debuggera*. Programski jezici koji su ugrađeni u sam paket su: C/C++ (uz Visual C++), Basic (uz Visual Basic .NET), C#, (uz Visual C#). Također se može dodati podrška i za ostale jezike, poput podrške za: Phyton, M, Ruby itd. Uz to, VS podržava XML/XSLT, HTML/XHTML, JavaScript i CSS. Slika (Slika 2.4) prikazuje razvojnu okolinu MS Visual Studio 2008.



Slika 2.4 Microsoft Visual Studio 2008.

3. Izrada teorijskog modela

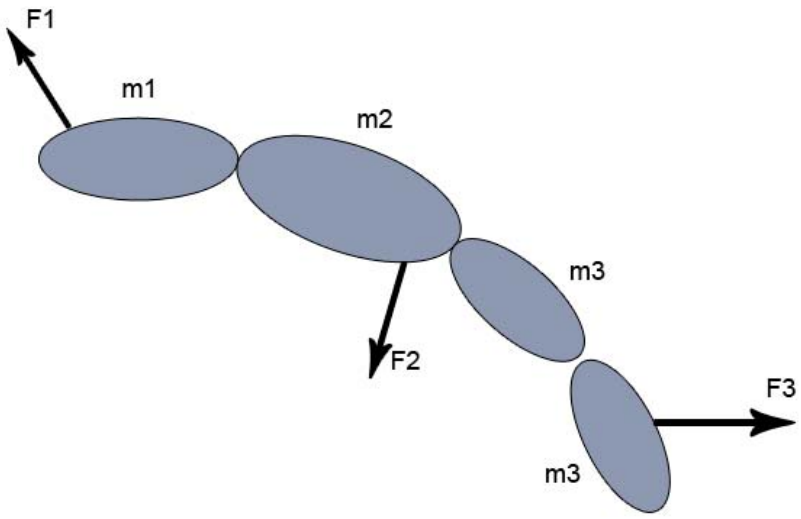
U prijašnjim poglavljima objašnjeno je sve što je potrebno da bi mogli definirati osnovne zahtjeve za izradu kinematičkog modela vlaka. Kao što se može vidjeti na slici (Slika 3.1), koristit će se jednostavni kinematički lanac. Ova vrsta lanca nije niti otvorena ni zatvorena, može se reći da je slobodna, pošto nije vezan ni na početku ni na kraju.



Slika 3.1 Shema vlaka prikazanog kao kinematički lanac.

Prilikom izrade vlaka, mora se uzeti u obzir njegovo kretanje. Postavljaju se dvije mogućnosti: gibanje u 2D prostoru ili gibanje u 3D prostoru. Odabran je 3D prostor i iz tog razloga će biti napravljen model vlaka u svemiru. Ovaj odabir je zapravo najviše utjecao na broj stupnjeva slobode zglobova. U ovom slučaju to može biti bilo koji broj od 1 do 6, i u ovom radu će se implementirati zglobovi sa 3 stupnja slobode. Dakle, vlak će moći obavljati rotaciju oko sve tri osi.

Što se tiče samog kretanja vlaka, primijenit će se model u kojem će biti zadane početne sile, te će simulacija pokazati što se događa. Dakle, moći će se zadati početne sile na bilo koji članak u lancu. Uz sile, za bilo koji članak još se mogu zadati i: hvatište sile, masa, orijentacija te moment tromosti. Uz sve te parametre, rezultati simulacije mogu biti vrlo nepredvidivi. Na slici (Slika 3.2) može se vidjeti shematski prikaz vlaka s definiranim parametrima. No, treba imati na umu da je ovo shematski prikaz u dvije dimenzije. Sve sile i orijentacije se mogu zadavati u tri dimenzije.



Slika 3.2 Primjer vlaka sa zadanim parametrima.

Samo gibanje kinematičkog lanca bit će ostvareno koristeći princip direktne kinematike. Dakle, prvo će se izračunati položaj i orijentacija početnog članka (lokomotive), pa zatim položaj i orijentacija sljedećeg članka u nizu, pa sljedećeg i tako sve do zadnjega.

4. Izrada praktičnog dijela

U ovom poglavlju bit će opisano na koji način je ostvaren ranije opisani teorijski model. Na koji način je napravljeno, zašto je tako napravljeno, moguća druga rješenja te problemi u izradi.

4.1. Izrada 3D modela

Za izradu 3D modela korišten je ranije opisan alat Blender. Pošto se za povezivanje programskog i grafičkog dijela koristi DirectX, odnosno SDK, pri odabiru alata za 3D modeliranje, jedan od najvažnijih zahtjeva koje odabrani alat mora zadovoljavati je da ima funkciju prebacivanja (eng. *export*) 3D modela u DirectX format (koji ima nastavak .X). Pošto Blender ispunjava zadani uvjet, a uz to je i besplatan, to ga čini idealnim alatom za izradu modela u ovom radu.

Svi modeli morali su zadovoljavati jedan zahtjev. Taj zahtjev glasi da konačna DirectX datoteka ne smije biti prevelika. Ako je DirectX datoteka prevelika, duže se učitava i iscrtava, te bi se na taj način usporila cijela simulacija, a možda i srušila. U skladu s tim, postavljena je gornja granica veličine DirectX datoteke – 300 KB. Postoji više načina za smanjivanje veličine konačne datoteke, no većina se svodi na jedno, a to je smanjivanje broja poligona. Naime, svi modeli se sastoje od poligona, i što je više poligona, potrebno je više internih podataka spremati. Neki od tih podataka su: zapis svakog poligona, zapis svake točke, povezivanje poligona i točaka, zapis normala, zapis koordinata tekstura itd. Preventivni način kojim se postiže manji broj poligona je jednostavno unaprijed skicirati i definirati 3D model. Što je detaljnije skiciran model na papiru, to će se lakše izmodelirati u aplikaciji. Drugi način je jednostavno brisanje poligona i stvaranje novih, no ovo je pomalo primitivan i prilično spor način. Nadalje, umjesto brisanja poligona, može se primijeniti ugrađeni modifikator *Decimate* (eng. *modifier*) unutar Blendera, no tada nije moguće mijenjati boje na različitim dijelovima modela. Kao rješenje za nepraktični modifikator nameće se vrlo zgodno rješenje – ugrađena skripta *Poly Reducer*. Pomoću ove skripte moguće je vrlo jednostavno podesiti željenu razinu detalja odabranog modela.

4.1.1. Modeliranje lokomotive

Koliko god kompliciran model bio, sve uvijek počinje od jednostavnih geometrijskih elemenata poput kocke, kugle, valjka, stošca, ravnine ili čak obične linije.

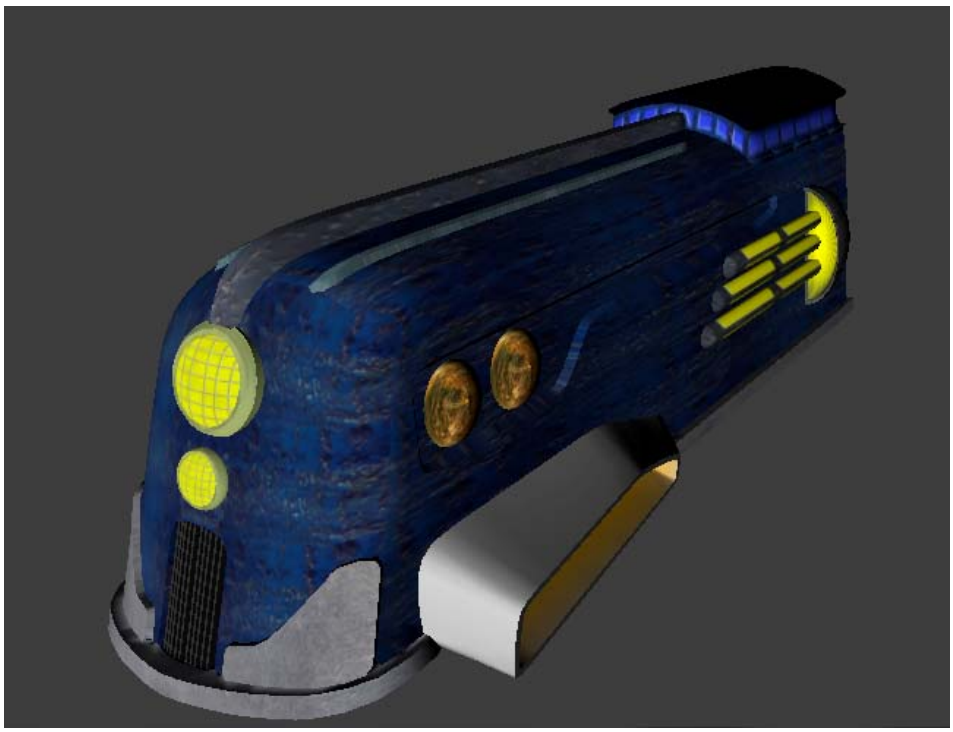
Na slici (Slika 4.1) prikazan je proces izrade vlaka, te se na slici (Slika 4.1 a) prikazan element od kojeg se počinje modelirati. Inicijalno je u Blenderu podešeno da se pri pokretanju, svaki put pojavljuje scena u kojoj se nalazi jedna kocka, kamera i izvor svijetlosti. Pošto je oblik vlaka više sličan kvadru nego kocki, kocku smo izdužili jednostavnom transformacijom „skaliranja“ (eng. *scale*). Kako je već bilo osmišljeno da prijelazi modela ne budu previše izraženi te da budu glatki, bilo je potrebno više poligona kako bi se postigao dobar efekt. Razdjeljivanje poligona na više se radi pomoću opcije „razdijeli“ (eng. *subdivide*), i ona radi tako da jednostavno jedan poligon u obliku kvadrata podijeli na četiri poligona. Ako radimo i sljedeću iteraciju, onda će se tih četiri poligona podijeliti na 16 itd, te se rezultat može vidjeti na slici (Slika 4.1 b).

Pošto je model vlaka zapravo simetričan, vrlo je korisno upotrijebiti ugrađeni modifikator u Blender koji se zove „zrcaljenje“ (eng. *mirror*). Koristi se tako da se prvo obriše jedna cijela strana modela, primjeni se modifikator zrcaljenja, te će se kao rezultat dobiti obrisana strana koja se referencira na dio koji se nije obrisao. Sada se model zapravo sastoji od dva dijela: originala i zrcaljenog dijela (koji je identičan originalu, samo je zrcaljen). Pošto je zrcaljeni dio kopija koja se referencira na original, sve što se napravi na originalu, napraviti će se i na njegovoj zrcaljenoj kopiji. Ovaj modifikator zbog toga uvelike olakšava modeliranje i vrlo je često korišten. Na slici (Slika 4.1 c) original je prikazan kao skup poligona kojima se može upravljati, dok je zrcaljeni, sivi, dio jednostavno njegova kopija na kojoj se kopira sve što se radi na originalu. Također se mogu vidjeti neki detalji koji su dobiveni korištenjem operacije „izvuci“ (eng *extrude*). Ova operacija doslovno izvlači selektirani poligon te se na taj način mogu stvarati bilo kakvi oblici. Na taj način je napravljena kabina na stražnjem dijelu vlaka, izbočina na vrhu, izbočine sa strane vlaka, te branici pri dnu. Kao što se može vidjeti na slici, sve te operacije su kopirane na zrcaljenu stranu. Kako bi se napravila postolja za prednja svijetla, jednostavno su napravljena dva valjka koja su dodana na prednji dio vlaka. Zbog zrcaljenja, i ove objekte je potrebno prerezati na pola.

Slika 4.1 Proces izrade vlaka.

Na slici (Slika 4.1 d) dodan je još jedan detalj s prednje strane, dok su na slici (Slika 4.1 e) dodani detalji s prednje strane vlaka. To su zapravo samo kvadratičasta tijela umetnuta u pripremljenu udubinu na prednjoj strani. Na slici (Slika 4.1 f) dodano je još jedno udubljenje poput onoga na prednjoj strani, te jedan polukružni objekt sa udubljenjem sa svake strane vlaka. Oba udubljenja napravljena su pomoću operacije *extrude* prema unutarnjoj strani modela. Krov, koji je također dodan koristeći operaciju *extrude*, dodan je na slici (Slika 4.1 g). Slika (Slika 4.1 h) prikazuje gotov model u DirectX Vieweru, koji je SDK-ova dodatna komponenta. Za razliku od prošle slike, tu je dodano još par detalja poput dvije polukugle i valjkasta tijela s obje strane vlaka.

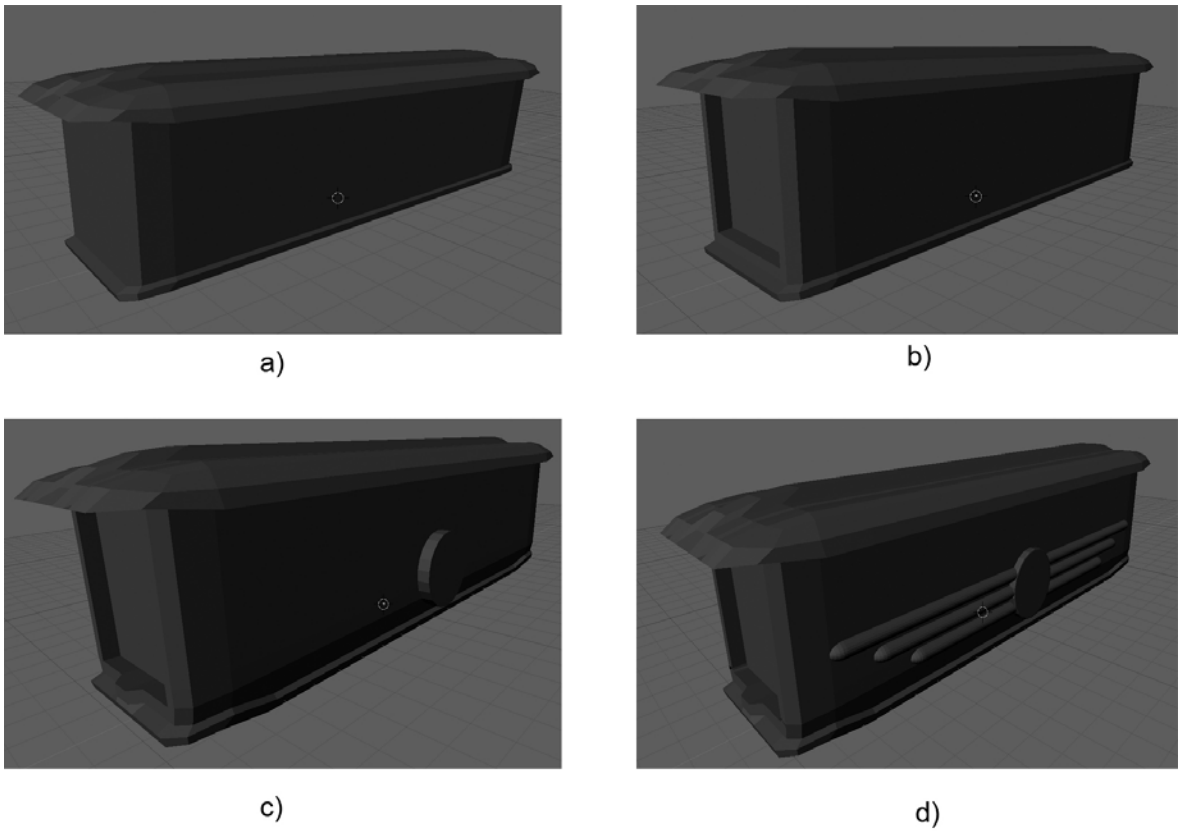
Slika (Slika 4.2) prikazuje gotov model vlaka sa stavljenim teksturama. Teksture su kompletno izrađene u Gimpu ili skinute s Interneta, pa onda obrađene.



Slika 4.2 Gotov model vlaka.

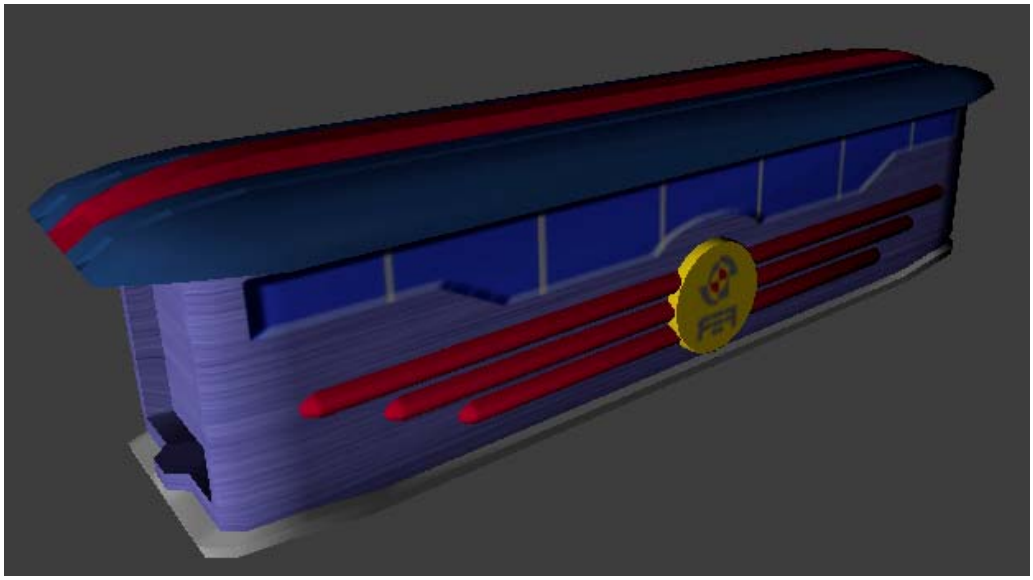
4.1.2. Modeliranje vagona

Vagon je modeliran na identičan način kao i lokomotiva, koristeći zrcaljenje, *extrude*, dodavanje dodatnih objekata i na kraju teksturiranje. Na slici (Slika 4.3) prikazan je proces izrade vagona.



Slika 4.3 Proces izrade vagona.

Slika (Slika 4.3 a) prikazuje objekt dobiven operacijom *extrude* iz kocke. Na slici (Slika 4.3 b) dodano je udubljenje s prednje i stražnje strane vagona također pomoću operacije *extrude*, dok se na slici (Slika 4.3 c) može vidjeti dodani valjak. Za kraj modeliranja dodano je još malo detalja na vagonu, što se može vidjeti na slici (Slika 4.3 d). Nakon toga su još samo postavljene teksture, te je završni rezultat prikazan na slici (Slika 4.4).



Slika 4.4 Gotov model vagona.

4.2. Fizikalna pozadina i korelacija s programskim kodom

Fizikalni zakoni su važan dio ovog rada. Vlak se giba u 3D prostoru poštivajući fizikalne zakone, i u ovom poglavlju će biti objašnjeni fizikalni principi korišteni u radu. No, prije ulaska u fizikalne proračune, potrebno je objasnit neke dijelove vezane uz programski kod i njegov način rada.

4.2.1. Upoznavanje s radom programa

Program radi tako da definicije članaka prima iz ulazne datoteke. Na slici (Slika 4.5) može se vidjeti primjer ulazne datoteke. Svaka definicija vlaka započinje ključnom riječju `train`, dok definicije vagona započinju ključnom riječju `cart`. To zapravo znači da na sceni može biti i više vlakova.

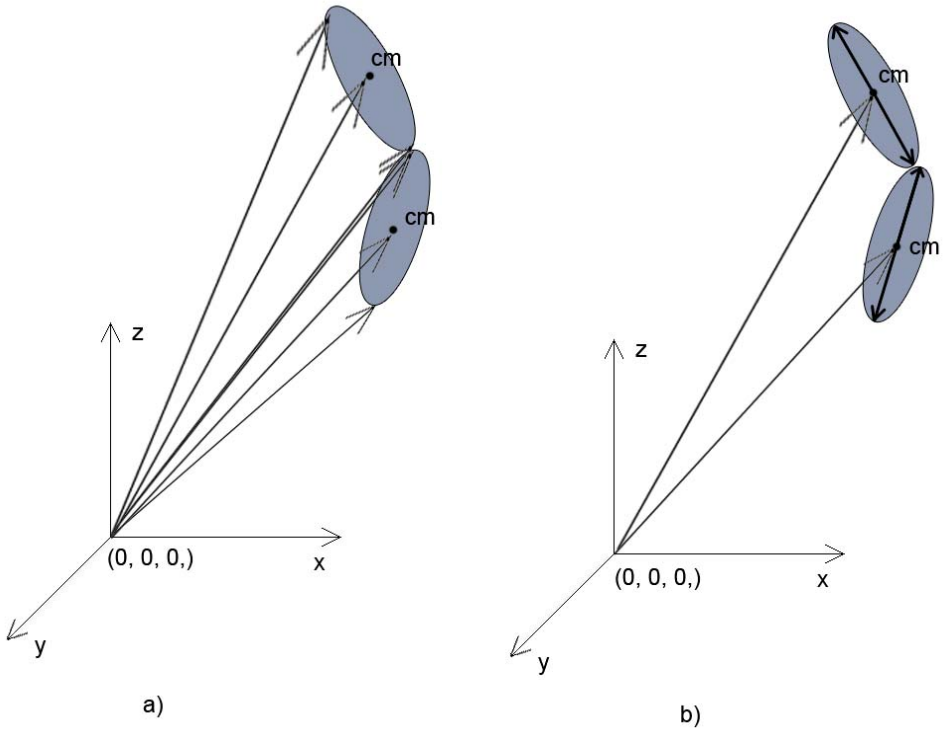
```
train
force:
9000 6000 0
25 0 0
F1
position:
20 20 0
lowerJoint:
-75 0 0
upperJoint:
75 0 0
mass:
10
inertion:
0.08
graphic:
locomotive.X
cart
position:
-30 20 0
lowerJoint:
-75 0 0
upperJoint:
75 0 0
mass:
10
inertion:
0.08
graphic:
vagon.X
cart
position:
-20 -20 0
lowerJoint:
-75 0 0
upperJoint:
75 0 0
mass:
100
inertion:
0.08
graphic:
vagon.X
```

Slika 4.5 Primjer ulazne datoteke.

Parametri koji se mogu zadavati u ulaznoj datoteci su sljedeći:

- Sila na pojedini članak (`force`)
    1. smjer i jačina sile
    2. hvatište sile
- Faktor inercije (`inertion`)
- Pozicija donjeg zgloba članka (`lowerJoint`)
- Pozicija gornjeg zgloba članka (`upperJoint`)
- Masa članka (`mass`)
- Članak se može nalaziti pod zadanim kutevima (`angX`, `angY`, `angZ`)
- 3D model u DirectX formatu koji će poslužiti kao model članka (`graphic`)

Još jedna bitna stvar osim ulazne datoteke je koordinatni sustav scene. Program interno ima definiran koordinatni sustav unutar kojega se stvaraju naši objekti, a pošto koristimo dosta vektorskih veličina (sila, akceleracija, hvatište sile, udaljenosti itd.), potrebno je znati njihove koordinate. Postoje dva načina zapisa koordinata: relativno i apsolutno. U konačnici, program uvijek koristi apsolutne koordinate, no kada bi te koordinate koristili u proračunima i u iscrtavanju, program ne bi dobro radio. Krivo bi računao položaje, a kao posljedica toga, i objekti bi se na krivom mjestu iscrtali na ekranu. Zato se za proračune i definiranje položaja objekta koriste relativne koordinate koje se, nakon izračunatog položaja objekta, pretvaraju u apsolutne. Na slici (Slika 4.6) se može vidjeti razlika između apsolutnih i relativnih koordinata. Centar mase je bitna komponenta i na slici je označen sa *cm*. Kao što se može vidjeti na slici, pomoću relativnih koordinata se može pamtiti samo vektor koji pokazuje na centar mase i onda pomoću njega izračunati sve ostalo.



Slika 4.6 Apsolutne i relativne koordinate

4.2.2. Fizikalni proračun

Početna pretpostavka od koje se krenulo pri izradi ovog rada glasi: smjer brzine ($\hat{v}$) je konstan te je gibanje jednoliko ubrzano ili usporeno. Od te početne pretpostavke se dalje gradi teorija. Ako je gibanje jednoliko ubrzano, onda je akceleracija ($\vec{a}$) konstantna (i po smjeru i po iznosu) , što se može vidjeti po formulama (1) i (2).

$$\vec{v} = \frac{d\vec{s}}{dt} \tag{1}$$

$$\vec{a} = \frac{d\vec{v}}{dt} \tag{2}$$

Nadalje, ako je akceleracija konstantna, onda lako možemo izračunati sile ($\vec{F}_i$) koja djeluje u sustavu, pošto imamo zadane mase svih članaka, po formuli (3). Indeks *i* u izrazu označava da se sila računa za *i*-ti članak u lancu. Masu svakog članka možemo zadati u ulaznoj datoteci, no to će biti objašnjeno kasnije.

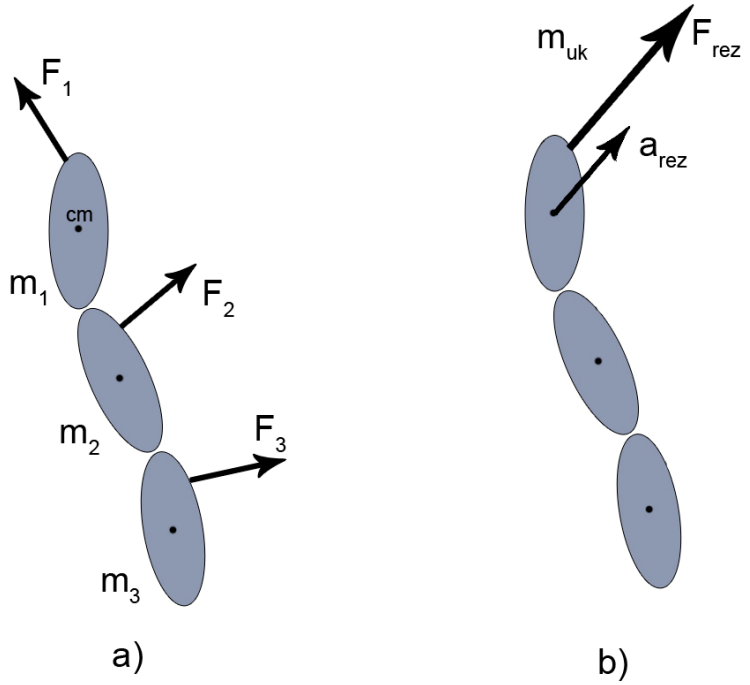
$$\vec{F}_i = m_i * \vec{a} \tag{3}$$

Dakle, brzina i akceleracija su konstantne ali im trenutno ne znamo ni iznos ni smjer. Zapravo nam je bitan iznos i smjer rezultantne akceleracije jer će se u tom smjeru gibati vlak. Silu za pojedini članak možemo dobiti iz izraza (3), te zbrajanjem tih sila pomoću izraza (4) dobivamo rezultantnu silu.

$$\vec{F}_{rez} = \sum_{i=0}^{n} \vec{F}_i \tag{4}$$

Pošto sada znamo rezultantnu silu $\vec{F}_{rez}$ i ukupnu masu m_{uk}, možemo izračunati rezultantnu akceleraciju po izrazu (3). Sada kada je poznata rezultantna akceleracija, znamo u kojem će se smjeru gibati čitav lanac.

Na slici (Slika 4.7) se može vidjeti shematski prikaz računanja rezultantne akceleracije.



Slika 4.7 Primjer računanja akceleracije.

Sada kada znamo iznos i smjer akceleracije, možemo izvesti translaciju u izračunatom smjeru. No, kada bi sada direktno izveli translaciju cijeli sustav bi se translatirao bez ikakve rotacije. Dakle, na primjeru na slici (Slika 4.7) sva tri članka bi zadržala trenutne orijentacije i tako bi se translatirali. Takav način nije realan i zato se, prije izvedbe translacije, prelazi na sljedeći korak u računu, a to je rotacija.

Kako bi izveli rotaciju, potrebna nam je kutna akceleracija (α). Kako bi dobili kutnu akceleraciju, potrebni su moment sile (M) i moment tromosti (I), kao što se može vidjeti u izrazu (5).

$$\alpha = \frac{M}{I} \tag{5}$$

Moment sile računamo po formuli (6), pri čemu je $\vec{r}$ udaljenost od centra mase do hvatišta sile $\vec{F}$.

$$\vec{M} = \vec{r} \times \vec{F} \tag{6}$$

Proračun momenta sile je jedan od razloga zašto se koriste relativne koordinate umjesto apsolutnih. Sada je izračunat moment sile, i jedino što je još nepoznato je moment tromosti, a to možemo izračunati po formuli (7).

$$I = I_{cm} + m * \vec{r}^{\,2} \tag{7}$$

I_{cm} je moment tromosti centra mase, a on se računa po formuli (8), pri čemu je k konstanta i za ovaj slučaj iznosi 0,08, l je duljina članka, a m masa članka, tj. vagona.

$$I_{cm} = k * l^2 * m \tag{8}$$

Koristeći sve ove formule, izračuna se kutna brzina i tek se tada može raditi rotacija članka.

4.2.3. Princip rada programa

Program se sastoji od pet dijelova:

- Main.cpp
- Train.h
- Timer.h
- Graphic.h
- Container.h

Sve počinje od funkcije `main` koja u biti samo poziva drugu funkciju - `WinMain`. Takav način rada je jednostavno nužan jer program koristi DirectX i namijenjen je za Windows okruženje.

Osnovni dio programa nalazi se u toj funkciji, a to je beskonačna petlja prikazana u kodu (Kod 4.1).

```
while(1)
	{
		double dt = timer.getTimeFromLastCall();
		detect_keys();
		for(int i = 0; i < objects.size(); i++)
		if(!objects[i]->physicsIgnore) objects[i]-
>move(dt);
		renderFrame(dt,objects);
	}
```

Kod 4.1 Glavna petlja.

Pomoću naredbe u 1. retku mjeri se vrijeme proteklo između dvije iteracije (za što nam služi zaglavlje `timer.h`), dok pomoću sljedećeg retka detektiramo i obrađujemo unos s tipkovnice. Vrijeme `dt` je bitan element jer se koristi u gotovo svim proračunima, direktno ili indirektno. U `for` petlji pomoću funkcije `move` obrađujemo fiziku. Pomoću zadnjeg retka se objekti crtaju na ekran.

Funkcija `move` rješava fiziku na sljedeći način. U 1. iteraciji, u funkciju `move` se ulazi s objektom lokomotive, te je ova funkcija zapravo namijenjena samo tom slučaju. Dalje se poziva glavna funkcija, odnosno funkcija u kojoj se rješava cijela fizika – `moveCart`, unutar zaglavlja `train.h`. Prvo što se unutar te funkcije radi je pretvaranje relativnih u apsolutne koordinate. Dalje se računa cijeli postupak objašnjen u poglavlju (4.2.2) za svaki članak u lancu i to na sljedeći način. Izračuna se rezultantna sila za 1. član, te se obavlja translacija. Nakon toga se izračuna kutna akceleracija te se obavlja rotacija.

Nakon što je izračunata i translacija i rotacija, ulazi se dublje u rekurziju, odnosno ulazi se u istu funkciju, no, ovaj put se računa fizika za sljedeći član u lancu, no s jednom razlikom.

S obzirom da se akceleracija množi s masom, a za jedan član lanca smo već sve izračunali, ovaj put će sila biti manja pošto je i masa manja za masu već izračunatog člana. Što se dublje ulazi u rekurziju, sila postaje sve manja i manja, te se može reći da se sila troši. Ovaj princip nije fizički potpuno ispravan, no pokazao se vrlo dobar za implementaciju, u odnosu na druge pristupe. Nakon što se izračunaju svi potrebni parametri za sve članke, slijedi iscrtavanje.

Zbog lakše implementacije, zglobovi su promatrani kao da ih ima dvostruko više. Prvi članak sadrži vektor koji pokazuje od centra mase do gornjeg zgloba, te još jedan vektor koji pokazuje od centra mase do donjeg zgloba.

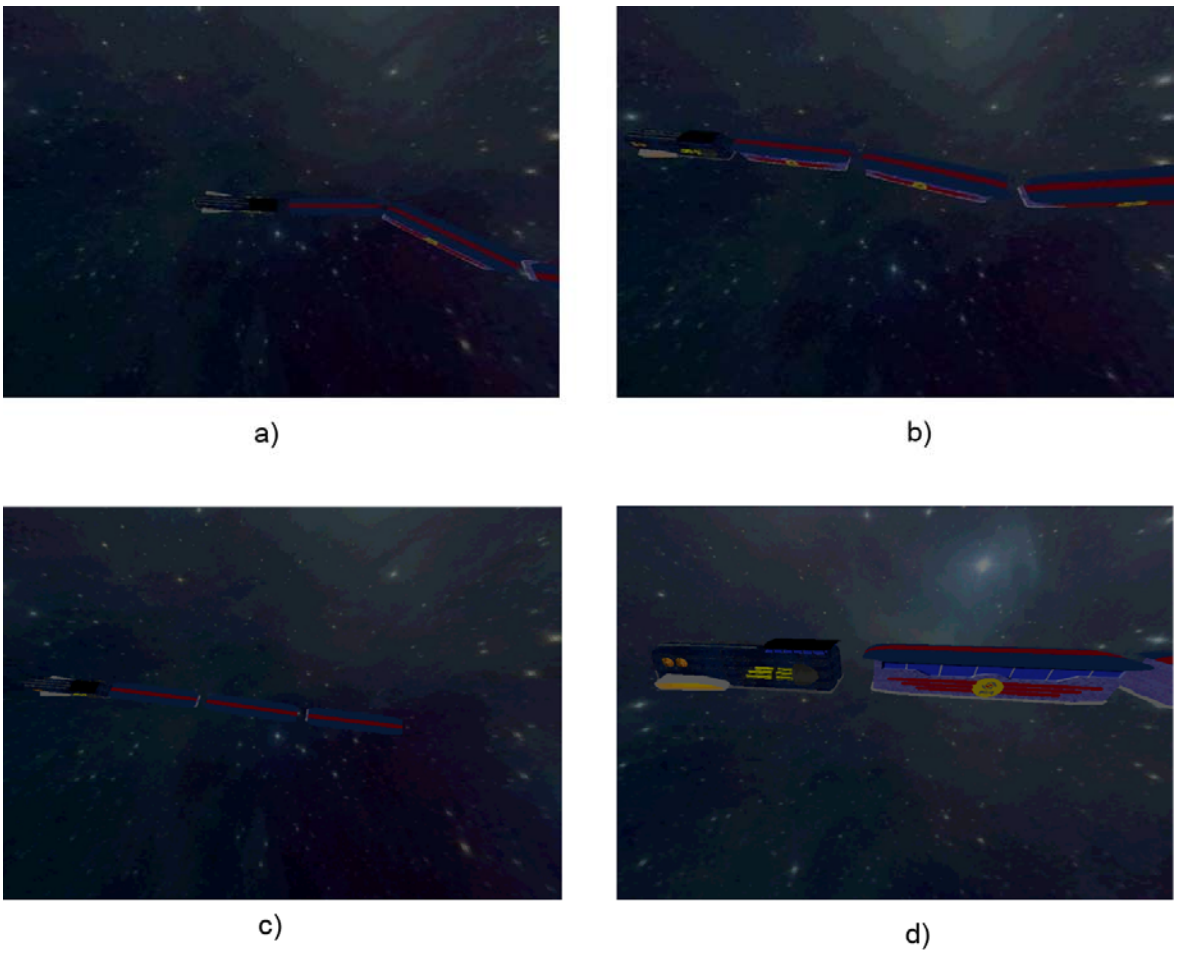
Isto tako ima i slijedeći članak te onaj nakon njega i tako dalje. Svi vektori unutar programa su implementirani pomoću ugrađene DirectXove strukture `D3DXVECTOR3`.

Unutar `for` petlje bitno je i ispitivanje `bool` varijable `physicsIgnore`. Ispitivanjem te varijable, cijela fizika se računa samo za 1. objekt, odnosno lokomotivu. Tako bar to izgleda iz ove perspektive. No, zapravo se, zbog rekurzivnih poziva objašnjenih ranije, fizika računa i za svaki vagon koji je povezan sa lokomotivom te se zbog toga funkcija `move` poziva samo za lokomotivu, a onda se dalje računa fizika za vagone.

Pomoću zaglavlja `graphic.h` objekti se iscrtavaju na ekran. Kako se vlak sastoji od lokomotive i vagona koji su zapravo identični modeli, grafički dio se olakšava pomoću zaglavlja `container.h`. Pomoću tog zaglavlja smo definirali neku vrstu spremnika unutar kojeg se spremaju korišteni modeli. Dakle, u ovom slučaju u spremniku se nalaze 3D modeli lokomotive i vagona. Umjesto da svaki objekt vagona (u programu) ima svoj 3D model, u spremniku se nalazi jedan model lokomotive i jedan model vagone, te svi objekti vagon pokazuju na jedan model vagona u spremniku.

5. Rezultati

Na slici (Slika 5.1) može se vidjeti konačni programski produkt.



Slika 5.1 Rezultati- prikaz programa.

Dakle, program radi tako da mu se zadaju početni uvjeti u ulaznoj datoteci, te se simulira ponašanje kinematičkog lanca. Na slici (Slika 5.1 a) prikazan je vlak u početnom položaju. Može se vidjeti da su neki vagoni zakrenuti pod određenim kutom, dok se na sljedećoj slici vide vidi kako vlak kreće. Dakle, vlak će se kretati u smjeru rezultantne sile i na slici (Slika 5.1 c) se već može vidjeti u kojem smjeru vlak ide. Na istoj slici se također može vidjeti da su se vagoni već poprilično izravnali te se kreću u smjeru lokomotive.

Također je implementirana pokretna kamera. Dakle, kamera se može micati lijevo i desno te se može zakretati. U privitku će biti opisano kako se rukuje programom.

U programu nije implementirana detekcija sudara zbog nedostatka vremena, te zbog toga objekti mogu prolaziti jedni kroz druge. Pošto je ovo sustav s mnogo parametara, vrlo se lako može pretjerati s početnim parametrima, te se sustav može ponašati nepredvidivo. Npr. može se dogoditi da rezultantna sila djeluje u jednu stranu te time zakrene lokomotivu tako jako da ona prođe kroz vagone. Nepovoljni početni uvjeti mogu rezultirati nepogodnom ponašanju sustava, poput:

- Prevelika sila može rezultirati prebrzim kretanjem vlaka tako da ga kamera ne može pratiti
- Sila zakrenuta u jednom smjeru koja je također prejaka može rezultirati nekontroliranom vrtnjom lokomotive ili cijelog sustava
- Prevelike mase mogu rezultirati kompletnom zastoju sustava, dok se zbog premalih masa sustav može prebrzo kretati
- Nepovoljni kutevi članaka također mogu pridonijeti nepredvidivosti sustava itd.

Dakle, ovo je vrlo delikatan sustav i zapravo postoji beskonačno mnogo ishoda.

Zaključak

Krajnji produkt ovog rada prikazuje simulaciju kretanja kinematičkog modela vlaka. Za zadane početne uvjete izračunava smjer kretanja vlaka i prikazuje simulaciju. Za rješavanje problema bilo je potrebno proučiti osnovne fizikalne zakone koji djeluju u tom sustavu. Iako se zapravo ne zna točno što bi se u stvarnosti događalo u tom sustavu, u ovom radu je dan jedan mogući slučaj. Nakon što su postavljeni temeljni zakoni koji djeluju u sustavu, bilo je potrebno programski zapisati potrebne jednadžbe, te taj programski kod povezati s napravljenim 3D modelima.

Cilj ovog rada je bio prikazati kinematički model vlaka i to je ostvareno. No, pošto sam sustav ovisi o mnogo varijabli, lako se može dogoditi nepredvidljivi tijek simulacije. Mogućih ishoda simulacije praktički ima beskonačno mnogo, no, za dobro postavljene početne uvijete, vrlo dobro se može vidjeti i pratiti gibanje kinematičkog modela vlaka.

Literatura

[1] Milington, I. *Game Physics Engine Development*, San Francisco, CA, 2007.

[2] Pandžić S., I. *Virtualna okruženja*, Element, Zagreb, 2004.

[3] Borjanović, V. *prezentacije iz Fizike 1*, lipanj 2009.

[4] Hlavač, V. *Robot Kinematics*, Faculty of Electrical Engineering, Praha, Češka, 2005.

[5] Kovačić, Z., Smolić-Ročak, N., Punčec, M. *Upute za laboratorijske vježbe, Osnove robotike*, Fakultet elektrotehnike i računarstva, Zavod za automatiku i procesno računarstvo, Zagreb, 2005.

[6] Grupa autora, *Blender 3D: Noob to Pro*, http://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro, lipanj 2009.

[7] GIMP, http://en.wikipedia.org/wiki/GIMP, lipanj 2009.

[8] DirectX, http://en.wikipedia.org/wiki/DirectX, lipanj 2009.

Popis slika

Popis tablica

Sažetak

Izrada kinematičkog modela vlaka

U radu je prikazana izrada vlaka koji se giba u svemiru, tj. može se gibati u 3D prostoru. Model vlaka izrađen je korištenjem kinematičkog lanca na koji djeluju osnovni fizikalni zakoni gibanja. Opisan je proces izrade 3D modela, razrada fizikalnih jednadžbi, te povezivanje grafike, fizike i programskog koda. Također je pobliže opisan sam pojam kinematičkog lanca i njegove vrste.

Ključne riječi: kinematički lanac, kinematika, svemir, vlak, Blender, Direct3D, DirectX SDK, simulacija gibanja, sila, akceleracija, translacija, rotacija, 3D modeli.

Summary

Kinematic model of a train

In this paper, the reader can see the proces of constructing a model of a train that can move in space, that is, in 3D space. The model of a train is constructed using a kinematic chain that followes some basic laws of physics, more correctly, the laws of motion. This paper describes the proces of making 3D models, elaborates motion equations and the connection between the graphical part, the physics and the program code.

Key words: kinematic chain, kinematics, space, train, Blender, Direct3Dm DirectX SDK, motion simulation, force, akceleration, translation, rotation, 3D models.

Skraćenice

3D – trodimentionalno

2D – dvodimenzionalno

d.o.f. – (eng. *degree of freedom*) – stupanj slobode

API – (eng. *Application programing interface*) – programsko sučelje

SDK – (eng. *Software Development Kit*)

KB – kilobajt

VS – programski alat Visual Studio

GIMP – (eng. *GNU Image manipulator program*)

OS – (eng. *Operating system*) – operacijski sustav

Privitak

Instalacija programske podrške

Kako bi se programski produkt mogao pokrenuti, nužno je na računalo instalirati DirectX SDK. Izvršna datoteka se obavezno mora nalaziti u zajednočkom direktoriju, tj. u direktoriju gdje su smješteni svi modeli, sve slike, klase itd.

Upute za korištenje programske podrške

Početni parametri se zadaju u datoteci `train.txt`. Format datoteke je:

```
train
force:
1000 0 0
25 0 0
F1
lowerJoint:
75 0 0
upperJoint:
75 0 0
mass:
10
inertion:
0.08
graphic:
locomotive.X
cart
position:
-30 20 0
lowerJoint:
-75 0 0
upperJoint:
75 0 0
mass:
10
inertion:
0.08
graphic:
vagon.X
```

Nakon što se pokrene simulacija, prikazati će se ekran s vlakom u svemiru (pod pretpostavkom da su početni uvijeti dobri i da vlak nije nestao iz vidokruga kamere). Kamerom se može upravljati pomoću tipki:

- W/UP – kamera se pomiče prema vrhu ekrana
- A/LEFT – kamera se pomiće prema lijevoj strain ekrana
- S/DOWN – kamera se pomiće prema dnu ekrana
- D/RIGHT – kamera se pomiće prema desnoj strain ekrana
- Q – kamera se rotira prema dolje s obzirom na horizontalu
- E – kamera se rotira prema gore s obzirom na horizontalu

Prilikom rotiranja kamere pomoću tipki Q ili E može se zarotirati na drugu stranu tako da se vidi dno vlaka, ako se previše rotira.

Iz simulacije se izlazi pritiskom na tipku Escape (ESC).