UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Edvard DEMŠAR

Simulacija in vizualizacija fizikalnih pojavov

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Matija MAROLT

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01571/2009
Datum: 01.06.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **EDVARD DEMŠAR**

Naslov: **SIMULACIJA IN VIZUALIZACIJA FIZIKALNIH POJAVOV**
SIMULATION AND VISUALISATION OF PHYSICAL PHENOMENA

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Kompleksnost razlage nastanka nekaterih fizikalnih pojavov nemalokrat terja poglobljeno strokovno znanje opazovalca. Za učinkovitejše reševanje problemov in razlage posameznih pojavov, je opazovalcu v pomoč vizualna metoda predstavitve fizikalnega problema in njegove simulacije v realnem času. Kandidat naj razišče didaktično vrednost simulacije in vizualizacije fizikalnih pojavov ter jih ustrezno podkrepi s praktičnim primerom.

Mentor:

doc. dr. Matija Marolt

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani **Edvard DEMŠAR**, z vpisno številko **24014112**, sem avtor diplomskega dela z naslovom **Simulacija in vizualizacija fizikalnih pojavov.**

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom **doc. dr. Matije Marolta**,
- so elektronska oblika diplomskega dela, naslov, povzetek ter ključne besede identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne **29. 6. 2009** Podpis avtorja: _____

Kazalo

Povzetek

To delo nam skuša predstaviti temeljne lastnosti upodabljanja in simulacij. Za lažje razumevanje nas popelje skozi kratek pregled teoretičnih osnov. Pri upodabljanju se dotaknemo in razložimo tudi človeško razumevanje dejanskega sveta v treh dimenzijah, ki nas obkroža. To je pomembno zato, ker se kasneje v sistemu JxyZET skušamo čim bolje približati vsem zakonitostim človeškega sprejemanja 3D okolice. Ker je bistveni del te naloge didaktični pripomoček pri učenju fizike, programski produkt JxyZET implementiran v programskem jeziku Java, predstavimo zelo na kratko tudi to močno programersko orodje. Poudarek je seveda na sposobnostih upodabljanja in računalniške simulacije v didaktične namene.
V drugem delu z naslovom JxyZET bralcu najprej skušamo prikazati didaktičen pomen produkta in ga nato popeljemo skozi njegove osnove. Nato razložimo potek simulacije sistema JxyZET v skladu s teoretičnimi osnovami, ki smo jih spoznali v prvem delu. Celoten postopek tudi matematično in fizikalno utemeljimo. Podobno storimo s 3D računalniškim upodabljanjem, ki ga predstavimo v naslednjem poglavju.
V zadnjem poglavju predstavimo iskanje in upodabljanje silnic električnega polja ter ekvipotencialnih ploskev, ki dajeta sistemu JxyZET profesionalno podobo učnega pripomočka pri fizikalnih poskusih in hkrati predstavljata matematično najzahtevnejši del tega programa.

Ključne besede:

upodabljanje, simulacija, JxyZET, fizikalni pojavi

Abstract

This work tries to present basic features of 3D rendering and simulations of experiments in physics. To understand things better there is a short overview of theoretical basics. When reading about rendering we also try to explain human perception of three dimensional world that surrounds us. This is an important issue because in our product JxyZET we try to get as close as possible towards human perception of 3D world. Basic part of this work is to present software product JxyZET which ecapsulates didactic potential by learning of physics and is implemented in programming language Java where you can also find short overview of this powerfull language. Main goal is to present capabilities of rendering and simulation in educational purposes.

Second part of this work, entitled JxyZET, tries to convince the reader about didactic potential of the product. Afterwards we present all basic functionalities of the product, we explain the sequence of simulation steps based on theory presented in the first part. Complete procedure is mathematically an physically proofed. We do the similar with 3D rendering which is presented in the next chapter.

In the last chapter we present searching and rendering of electrical field lines and equipotential surfaces which give to JxyZET real professional look of educational resource in learning physical experiments.. The topics explained in this chapter represented mathematically the most difficult part to be implemented in JxyZET.

Keywords:

rendering, simulation, JxyZET, physical event

1. Uvod

Fizikalni pojavi v naravi terjajo poglobljeno razumevanje opazovalca in so nemalokrat preveč kompleksni za razumevanje. Način pridobivanja znanja, potrebnega za razumevanje teh pojavov, je navadno dolgotrajen in naporen. Psihologi, pedagogi in drugi šolski delavci se že dolga leta trudijo optimizirati proces učenja zato naloga skuša predstaviti didaktičen smisel interaktivnih simulacij fizikalnih pojavov. Ker je fizika eksperimentalna znanost, mora pouk fizike temeljiti na eksperimentih in problemski zasnovi. Ker ima pri fiziki eksperiment vodilno vlogo in ker je izvedba eksperimenta nemalokrat otežena, ima računalnik kot didaktični pripomoček lahko veliko vlogo. Mnogo eksperimentov je moč izvesti s pomočjo posebnih simulacijskih programov, kjer lahko sestavimo model eksperimenta, ga simuliramo in ustrezno upodobimo v realnem času.

V nalogi se najprej dotaknem upodabljanja kot takega. Pri upodabljanju je pomembno človeško dojemanje sveta, zato se najprej srečamo z analizo človeškega dojemanja tridimenzionalnega sveta, ki nas obkroža. Dobra psihološka podlaga nam namreč kasneje koristi pri implementaciji upodabljanja modelov s pomočjo računalniškega programa JxyZET.

Zaradi poudarka na didaktiki je treba predstaviti didaktične sposobnosti računalniških simulacij z njihovo uporabo ter prednostmi. V podpoglavjih si lahko ogledamo simulacije kot pripomoček pri učenju in nadaljnje didaktične pristope k abstrakciji in uporabi matematičnih modelov.

Glavni in s tem tudi praktični del naloge je opisan v drugih dveh tretjinah. Gre za obsežen računalniški program JxyZET, katerega sem razvijal v laboratoriju za računalniško grafiko in multimedije, pod vodstvom prof. dr. Saše Divjaka in v okviru mednarodnega projekta CoLoS. CoLoS (Conceptual Learning of Science), projekt konceptualnega učenja znanosti, združuje raziskovalne skupine mnogih univerz po Evropi in ostalih državah sveta kot na primer ZDA, Rusija in Tajska. Njegov cilj je promocija inovativnih učnih metod na področju znanosti in tehnologije. Glavne teme, na katere se osredotoča, so:

- učenje in razumevanje temeljnih konceptov v znanosti;
- integracija kvalitativnega razumevanja s kvantitativnimi metodami in
- uporaba simulacij v učnih procesih.

Cilj CoLoS-a je izboljšanje učenja znanosti z uporabo inovativne in interaktivne programske opreme. CoLoS prav tako raziskuje nove načine uporabe računalnikov v učenju znanosti in skuša spodbuditi razvoj intuitivnega razumevanja in občutka za znanost pri študentih.
Sistem JxyZET je del projekta CoLoS in lepo izpolnjuje njegova načela in cilje ter se lepo vključuje med mnoga dela, ki so prav tako del tega mednarodnega združenja.

JxyZET lahko poimenujemo tudi orodje ali simulator, zato v nalogi podrobno opišemo njegovo strukturo. Začnemo s predstavitvijo univerzuma ter njegovih osnovnih primitivov, delcev. Nadaljujemo z lastnostmi slednjih ter medsebojnimi relacijami. Univerzum ne bi igral nobene vloge, če ne bi vplival na naše delce skozi zunanje dejavnike, ki so prav tako podani.

Ko je naš univerzum, skupaj z vsemi elementi, zmodeliran, ga je treba upodobiti. Upoštevajoč vse teoretične podlage, ki sem jih uvodoma navedel, so le-te v veliki meri tudi zajete pri algoritmih upodobitve. Načini upodabljanja in tehnike so torej opisane v nadaljevanju.

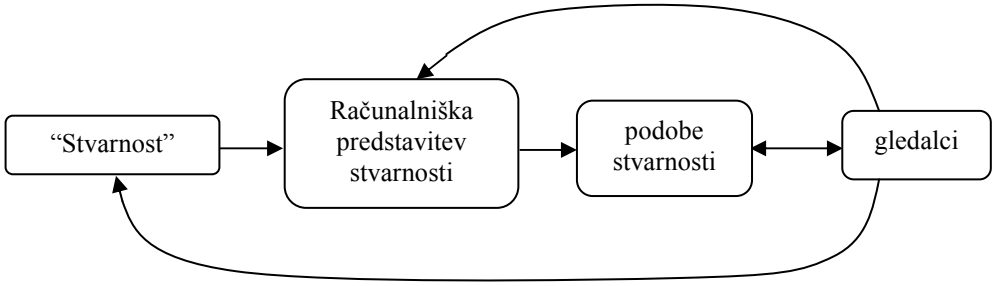
V zadnjem delu naloge, opišem dva od bolj zahtevnih algoritmov za upodabljanje in simulacijo silnic električnega polja ter ekvipotencialnih ploskev. Ta dva algoritma za upodabljanje in simulacijo razlikujeta orodje JxyZET od ostalih, ker mu povečujeta uporabnost. Za oba problema predstavim teoretično podlago, ki jo v programu upoštevam pri simulaciji in upodobitvi.

V okviru projekta CoLoS obstajajo celoviti tečaji iz elektrike in mehanike, kamor je integriran tudi JxyZET. Oblikovani so v obliki spletnih strani tako, da so prenosljivi preko svetovnega spleta ali njegovih inačic zaprtega tipa. Sistem JxyZET, skupaj s tečaji, tvori visoko interaktiven učbenik za učenje fizike na srednjih in visokih šolah.

2. Upodabljanje

2.1. Definicija besede upodobitev

Upodobitev je nekaj, kar je vidno človeškemu umu ali domišljiji. Pomeni tudi tvoriti miselno podobo, sliko ali risbo nečesa nevidnega ali podobo neke abstrakcije. (Povzeto iz [1])
V tem delu se bomo osredotočili na računalniško upodabljanje 3D prostora. Zato je prav, da si ogledamo simboličen potek upodabljanja od realnega ali računalniškega sveta do gledalca. Spodnja slika dovolj nazorno prikazuje preslikavo računalniške predstavitve sveta z izbiro tehnike, ki poveča človeško razumevanje in komunikacijo.



Slika 2.1: Upodobitev stvarnosti in interakcija gledalca

2.2. Zakaj sploh upodabljanje in kakšni so cilji?

Ker je odgovorov na zastavljeno vprašanje več, jih strnimo v naslednji seznam:

- raziskovanje in izkoriščanje podatkov in informacij;
- stopnjevanje razumevanja procesov in konceptov;
- narediti nevidno vidno;
- učinkovita predstavitev pomembnih značilnosti;
- kontrola kvalitete simulacij in meritev;
- povečanje produktivnosti v znanosti in
- uporabljeno kot medij za sodelovanje in sporočanje.

2.3. Dojemanje sveta

Če prezremo razne teorije, ljudje živimo v pravem tri-dimenzionalnem evklidskem prostoru. Ko govorimo o 3D upodabljanju, imamo v tem kontekstu pravzaprav v mislih 2,5 dimenzije. Pri naših očeh to pomeni, da lahko vidimo le v eni smeri in le površine raznih objektov v tej smeri [7].
Zakaj torej vidimo le 2,5 dimenzije? Z vsakim posameznim očesom smo sposobni videti le dve dimenziji. Vsako oko nam preslika trenuten prostor v dveh dimenzijah **D x D** v skupino atributov **A**, ki predstavljajo barvo, izrazitost barv, mogoče tudi teksturo, itd. To skupino atributov dobijo naši možgani. Torej je v vsakem očesu vgrajena naslednja preslikava:

D x D -> A (2.1)

S stereoskopskim vidom in ostalimi informacijami o globini naši možgani v bistvu tvorijo še dodatno dimenzijo D poleg množice atributov A, ki nam daje sposobnost ocenjevanja razdalje objektov v katerikoli smeri:

D x D -> (A x D) (2.2)

Če bi radi videli pravi 3D prostor, lahko že sedaj opazimo, da stoji pri naši preslikavi ena od dimenzij D na napačni strani enačaja.
V stvarnem svetu obstaja nekaj, ali pa tudi nič, v vsaki točki prostora. Za vsako točko 3D prostora obstajajo fizični atributi, ki jih označimo s **P**. Preslikava v atribute **P** je definirana tako:

D x D x D -> P (2.3)

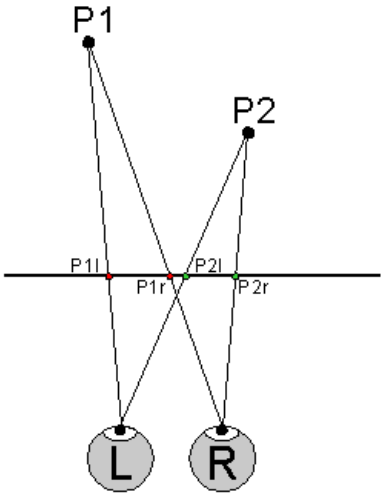
Problem gledanja je v tem, da bi morala preslikava (2.3) sovpadati s preslikavo (2.2), vendar temu ni tako. Od tu torej omejenost človeškega očesa pri gledanju 3D prostora. V eni smeri lahko vidimo le en objekt, navadno tistega, ki nam je najbližji. Pri računalniškem upodabljanju 3D prostora ljudje lahko vidimo objekte na neki konstantni razdalji, objekte z določenimi lastnostmi ali celo zelo oddaljene objekte. Kakorkoli že, vse skupaj vidimo kot 2,5D prostor. Ta simbolična polovica dimenzije glede na človeško zaznavanje 3D prostora pomeni tako malo, da je naš pogled bližje 2,00001D.
Razlika med 2D in 2,5D je sposobnost človeškega zaznavanja globine, o kateri spregovorimo v naslednjem poglavju.

2.4. Dojemanje globine

Naše oči so torej sposobne generirati le 2D sliko iz mrežnice in nimajo posebnih komponent za dojemanje globine. To zahteva interpretacijo psiholoških pokazateljev, ki omogoča uporabno dojemanje globine. Sodeč po številnih optičnih iluzijah, ta interpretacija ni vedno pravilna [2].
Oglejmo si nekaj dejavnikov, ki ključno vplivajo pri dojemanju globine.

2.4.1. Neskladje in konvergenca

Približno 6 cm razdalje med dvema očesoma nam omogoča, da vidimo dve različni sliki okolice. To razliko med horizontalno pozicijo slik iz obeh oči imenujemo binokularno neskladje. To je nedvomno najpomembnejši dejavnik za dojemanje globine.
Ko opazujemo okolico z dvema očesoma, se ne zavedamo dejstva, da večina videnih objektov stimulira različna področja obeh mrežnic. Objekti, ki so dovolj blizu, se projicirajo na različna območja na mrežnici, ker jih vsako oko vidi na drugačni razdalji. To povzroča določeno neskladje, ki ga človeški razum izkoristi za pridobitev informacije o globini opazovanega objekta tako kot kaže spodnja slika 1.2.



Slika 2.2: Neskladni sliki na obeh mrežnicah očesa pri opazovanju bližnjega objekta

Binokularno informacijo o globini lahko pridobimo s konvergenco obeh osi gledanja na objekt, ki ni oddaljen več kot 3 metre. Pri objektih, oddaljenih več kot 3 metre, je razlika v kotih obeh osi gledanja premajhna. Informacijo o konvergenci naši možgani pridobijo od očesnih mišic.

2.4.2. Gibalna paralaksa

Opazujemo mirujoče objekte na različnih razdaljah. Ob premikanju glave vidimo, da se različno oddaljeni objekti premikajo z različno hitrostjo. Ta informacija se imenuje relativna gibalna paralaksa. S pomočjo relativne razlike hitrosti opazovanih objektov so naši možgani sposobni pridobiti informacijo o globini posameznih objektov. Za pridobivanje tovrstne informacije je dovolj že eno oko.

2.4.3. Slikovni dejavniki

Različne dejavnike dojemanja globine je moč najti na slikah. Tu govorimo o navadnih slikovnih delih s strani umetnikov in nikakor ne npr. stereogramih.

2.4.4. Prekrivanje

Prekrivanje nam daje informacijo, da je objekt, ki prekriva del drugega objekta, bližje kot delno prekriti objekt.

2.4.5. Senčenje

Osenčen opazovani objekt nam daje dodatno 3D informacijo ter informacijo o izvoru svetlobe.

2.4.6. Relativna velikost

Objekti enake velikosti na različnih razdaljah generirajo različne velikosti slik na mrežnici. Razmerje velikosti in razdalje je dejavnik zaznave oddaljenosti objektov znane absolutne velikosti.

2.4.7. Linearna perspektiva

Kot posledica razmerja velikosti in razdalje, se perspektiva odraža pri fizičnih vzporednicah tako, da vse črte konvergirajo proti neki oddaljeni točki. Zanimivo je tudi, da se ta dejavnik globinske informacije ni uporabljal pred letom 1400. Pri umetniških delih, pred letom 1400, so se uporabljala le senčenja, prekrivanja in relativne velikosti za prikaz različnih globin objektov.

2.4.8. Teskturni gradient

Teksturni gradient je še en faktor, ki je odvisen od razmerja velikost-oddaljenost. Teksture namreč vsebujejo manjše strukture s povečevanjem oddaljenosti.

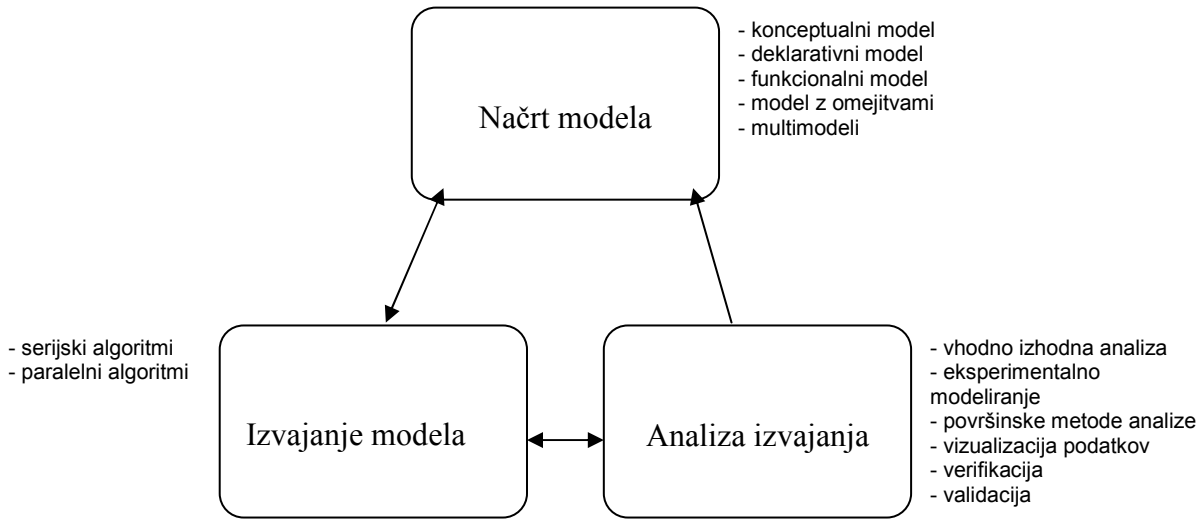
Polega vseh naštetih dejavnikov poznamo še dejavnik izvora svetlobe, kjer bolj osvetljeni objekt od dveh enakih dojamemo kot bližjega. Pri t. i. zračni perspektivi je bolj oddaljen objekt bolj zamegljen zaradi naravne poti svetlobe, ki se pri prehodu skozi zrak odbija in lomi.

Tehnike navidezne resničnosti nam dodajo še kanček informacije, ki je bližja gledanju 3D prostora, ker lahko vidimo izza objektov in mogoče tudi notranjost le-teh.
Omenjena razlaga nam bo prišla prav kasneje, ko se bomo dejansko lotili računalniškega upodabljanja zelo specifičnega prostora, prostora delcev v simulacijah fizikalnih pojavov. V naslednjem poglavju bo govora o simulacijah, s poudarkom na didaktičnih zmogljivostih le-teh. Programski produkt, ki bo opisan na koncu, je namreč didaktični pripomoček pri učenju fizike.

3. Didaktične sposobnosti simulacij

3.1. Računalniška simulacija

Računalniška simulacija je disciplina modeliranja modela dejanskega ali teoretičnega fizičnega sistema, izvajanja modela na računskem stroju in analiza izhodnih podatkov. Simulacija vključuje princip "učenja z grajenjem". Učenje na sistemu zahteva najprej izgradnjo modela in nato njegovo uporabo. Pri simulaciji moramo razumeti stvarnost in njeno kompleksnost ter zgraditi umetne objekte, na katerih dinamično odigramo njihove vloge. Računalniška simulacija je elektronska inačica igranja teh vlog in nam služi za igro sintetičnih okolij in navideznih svetov. Znotraj celotne naloge simulacije obstajajo tri glavna področja: načrtovanje modela, izvajanje modela ter analiza modela (glej sliko 3.1).



Slika 3.1: Tri področja računalniške simulacije

Če hočemo simulirati fizični objekt, moramo najprej zgraditi matematični model, ki predstavlja ta fizični objekt. Modeli so lahko v več oblikah, kot deklarativni, funkcijski, modeli z omejitvami ali multimodeli. Multimodel je model sestavljen iz več modelov, ki predstavljajo stopnjo podrobnosti fizičnega sistema.
Ko je model zgrajen, se lotimo naslednje naloge, izvajanja modela na računalniku. Zato moramo izdelati program, ki se izvaja po časovnih korakih, medtem ko ažurira stanje in spremenljivke našega matematičnega modela. Obstaja več načinov "izvajanja po časovnih korakih". Časovno prehajanje lahko zagotovimo z dogodkovnim dodeljevalnikom časa (event scheduler) ali implementiramo povečevanje časa s pomočjo časovnih rezin. V našem produktu JxyZET sta uporabljeni obe metodi "izvajanja po časovnih korakih". Program lahko izvajamo tudi na računalniku z visoko stopnjo vzporednosti. Takšno simulacijo imenujemo vzporedna in porazdeljena simulacija. Za bolj zapletene modele je to edini možni način pridobivanja rezultatov v nekem zglednem času. Pri sistemu JxyZET bomo videli, da s tem

načinom simulacije ne pridobimo veliko zaradi med sabo odvisnih funkcij, ki se izvajajo v eni časovni rezini.

3.2. Uporaba simulacij

Verjetno se sprašujete, ali moramo uporabiti simulacijo za študijo dinamičnih sistemov. Obstaja veliko metod modeliranja sistemov, ki ne vsebujejo simulacije, temveč temeljijo na reševanju zaprte oblike, kot na primer sistemi linearnih enačb. Simulacijo potrebujemo predvsem v naslednjih primerih:

- ko je model zelo kompleksen, z veliko spremenljivkami in medsebojno odvisnimi komponentami;
- ko so relacije med spremenljivkami modela nelinearne;
- ko model vsebuje naključne elemente in
- ko mora biti izhod modela vizualne narave kot v 3D računalniški animaciji.

Sistem JxyZET vsebuje veliko spremenljivk, medsebojno odvisne komponente, izhod modela pa mora biti računalniška 3D animacija.
Moč simulacije, tudi v enostavno rešljivih linearnih sistemih, je uporaba splošne tehnike izvajanja za reševanje velike množice različnih sistemov. Drugi pomemben vidik tehnike simulacije je gradnja simulacijskega modela, ki kar se da natančno predstavlja realen sistem. Če se lotimo reševanja zaprte oblike, moramo velikokrat model prilagoditi metodi reševanja zaprte oblike. Tak model odstopa od dejanskega sistema, kar pa vsekakor ni naš cilj.
Lahko sklenemo kompromis med metodami reševanja zaprte oblike in simulacijo kot na primer v sistemu v programu JxyZET. Najprej smo začeli z modeliranjem sistema z analizo zaprte oblike, nato smo pa nadaljevali s simulacijo.

3.3. Očitne prednosti simulacij

Ko realni svet nadomestimo z umetnim (modelom), nam simulacija na tem modelu nudi:

- prihranek pri vseh virih, ki bi jih dejansko potrebovali;
- izognitev potencialnim nevarnostim, ki nas čakajo v realnem svetu in
- zaustavitev ali zakasnitev simulacije, kar nam omogoča globlje razumevanje sistema in njegov značaj.

3.4. Uporaba simulacij za potrebe učenja

Splošno znano dejstvo je, da učenje, ki zajema čisto pomnjenje, temelji na ponavljanju s strani učenca. Takšno znanje ne more biti implementirano ali preneseno, temveč moramo za dolgotrajni učinek imeti sposobnost rekonstrukcije.

Simulacija nam nudi možnost interakcije in takojšnjega odziva na uporabnikove akcije, s tem pa razvoj raziskovalnega učenja. Možnosti za aktivno učenje so v primerjavi z navadnimi učbeniki ali s podobnimi neinteraktivnimi mediji dejansko povečane [4].
Znano je tudi, da se lahko naučimo skritih relacij različnih dejstev najboljše, če so ta dejstva predstavljena čim bolj povezano v prostoru in tudi v času. Dobro zasnovane simulacije nudijo natančno to možnost, ker dajejo takojšen odziv na vsako uporabnikovo akcijo. Izvajanje več simulacij istočasno z različnimi parametri nam omogoča doseči enak cilj.

3.4.1. Podpora samostojnemu učenju

Ni dvoma, da je učenje s pomočjo učitelja večinoma dragoceno, nepogrešljivo in nujno. Prav tako je pomembno tudi samostojno učenje in vaje, ker ima vsaka oseba različne navade učenja, hitrost učenja in interese. Lahko bi rekli, da s pomočjo samostojnega učenja kompenziramo učenje s pomočjo učitelja.
Simulacije, ki so enostavne za uporabo in prilagodljive posameznim potrebam, nudijo možnost uvedbe večje stopnje samostojnosti pri učenju [10].

3.4.2. Prilagajanje kompleksnosti

Stvarni poskusi so nepogrešljivi za dokazovanje pravilnosti in so temelj za konstrukcijo znanja, toda velikokrat so prezapleteni in niso zlahka razumljivi. Kompleksnost simulacije lahko prikrojimo tako, da ustreza potrebam posameznika.
Postopno povečevanje kompleksnosti je nujno za boljše razumevanje v primerih, kjer si lahko pomagamo s simulacijami. Take primere lahko najdemo v fiziki. Na primer pri matematično enostavnem problemu, masnem središču telesa, kjer najprej spoznamo masno središče togega telesa in nato še elastičnega. Tak pristop, iz nečesa matematično enostavnega v nekaj matematično zapletenega, verjetno za razumevanje problema ni optimalen.

3.4.3. Priložnost novih didaktičnih pristopov k abstrakciji in uporabi matematičnih modelov

Glavni problem učenja in poučevanja znanosti je zgodnja uporaba matematičnih simbolov in metod, ki predstavljajo prepreko za marsikaterega učenca. To ga vodi do večjega nerazumevanja, s tem se učenec vda v usodo in na koncu ga čaka neuspeh.
Računalniške simulacije predstavljajo enako stopnjo natančnosti kot matematične enačbe, vendar je uporabljeni "jezik" računalniška animacija, ki ponuja učencu takojšnje razumevanje. Elementi tega "jezika" lahko torej služijo kot vmesna faza k abstrakciji in omogočajo razumevanje večjega števila problemov, kot je to možno s tradicionalnimi matematičnimi metodami.

3.4.4. Različni pogledi na računalniške simulacije

Prav je tudi, da si ogledamo različna mnenja nekaterih strokovnjakov omenjenega področja iz celega sveta. Njihova mnenja so pomembna, saj se na njih lahko veliko naučimo. Tako na primer Laiz Castro Benito iz oddelka za kemijo in fiziko v Madridu meni, da je vse lepo in prav s sodobno tehnologijo, vendar ga moti preveliko vpletanje računalnikov v učne procese.

Nasprotuje oddaljevanju učitelja od učenca in s tem razhajanje misli obeh udeleženih. Seveda se z njim strinjamo in moramo poudariti, da naj kakršnakoli računalniška simulacija kakršnegakoli problema služi le kot pripomoček pri učenju ne pa kot osnovno sredstvo. Na mednarodni konferenci GIREP-ICPE-ICTP, leta 1996, je bila to edina izrazita kritika računalniškim simulacijam v pedagoških procesih.
Prav tako so kritični nekateri pedagogi [15], ki opozarjajo na zmerno uporabo simulatorjev fizikalnih pojavov v procesu učenja.

Drugi se zavedajo izpostavljenega problema, vendar celotno stvar jemljejo bolj pozitivno. Tako na primer Erika Mechlova iz univerze v Ostravi na Češkem meni, da so računalniške simulacije v pedagoških procesih zaželene, vendar ob asistenci profesorja.
Poglejmo v drugo skrajnost. Elena Sassi iz oddelka za fiziko na univerzi v Napoliju v Italiji meni, da zelo pogosto prihaja do konflikta med učenci in učitelji na nivoju razumevanja in sprejemanja različnih konceptov. Zato pričakuje od snovalcev kakršnihkoli računalniških učnih pripomočkov, da se bolj potrudijo v smeri razvoja močnih, zaznavnih in neposrednih orodij, ki bi pripomogli učencu do osvojitve zaželenega znanja brez preprek.

Lahko vidimo, da je strokovna javnost različno opredeljena glede uporabe simulacij. Moje mnenje je, da se moramo sprijazniti z dejstvom, da so postali računalniki del vsakdanjika in moramo najti kompromis ter optimalno območje interakcije med človekom (učiteljem) in računalnikom kot pedagoškim pripomočkom.

4. JxyZET

4.1. Splošno

Simulacijski program JxyZET nudi možnost vizualizacije animiranih objektov v fiziki v treh dimenzijah in omogoča uporabniku veliko mero samostojnega in ustvarjalnega dela. Razvit je v celoti s programskim jezikom Java in tako popolnoma platformsko neodvisen, kar omogoča delo tako v skupinah kot posameznikom.

Implementacija programa JxyZET s programerskim orodjem Java nam omogoča lažje programiranje ter bolj varno izvajanje napisanih programov.
Ker je Java objektno usmerjen jezik, so v Javi vse programske komponente razredi in daljni potomci nekega razreda, očeta. Lastnost dedovanja s pridom izkoristimo tudi v sistemu JxyZET, kot bomo to spoznali kasneje.
Kot vemo nam pri programskem jeziku C/C++ največ težav povzročajo kazalci. Velika večina vseh programov s hrošči je posledica napačne uporabe kazalcev. Pri Javi moramo upoštevati, da je čisto vsak definiran objekt kazalec. S tem zablode o kazalcih odpadejo. Seveda je treba vsakemu na novo definiranemu objektu narediti primerek, da lahko kasneje z njim ravnamo.
Naslednja odlična lastnost javanske tehnologije je njeno upravljanje s pomnilnikom. Za čiščenje poskrbi t. i. javanski čistilec smeti. Ko prenehamo uporabljati objekt oziroma, ko nima več reference nanj, se v ozadju avtomatsko sproži proces čiščenja vseh resursov, ki jih je ta objekt zasegal. Ta lastnost nam pride še posebej prav pri iskanju in upodabljanju električnih silnic in ekvipotencialnih ploskev, saj je na primer velikost povprečno velike ekvipotencialne ploskve 2000 prostorskih točk in se dinamično spreminja. Taka stvar zahteva veliko programerskih spretnosti in discipline v ostalih programskih jezikih. Slaba lastnost čistilca pa je, da celotno izvajanje programa nekoliko upočasni [6].
Naslednja dobra lastnost je izvajanje javanskih programov v t. i. peskovniku, katerega vsebuje že vsak internetni brskalnik. V peskovniku je izvajanje programa popolnoma varno za gostujoči operacijski sistem, saj so izhodi iz peskovnika zelo omejeni in varnostno pogojeni.
Dotaknimo se še večnitne sposobnosti, saj se naš program JxyZET izvaja v več nitih. Java lepo poskrbi za njihovo uskladitev tako, da se lahko skoraj 100 % izognemo nevarnostim, ki jih prinašajo večnitni programi.

4.2. Znanstvena vsebina

Program JxyZET pokriva vrsto osnovnih konceptov mehanike in elektrike, ki skupaj tvorijo del vsakega interaktivnega tečaja fizike. Glavne koncepte lahko strnemo v:

- kinematiko;
- kinetično energijo;
- Newtonove zakone;
- ohranitev navora;
- ohranitev energije;
- Hookov zakon;
- gravitacijo;

- električni naboj in električno polje in
- ekvipotencialne ploskve.

4.2.1. Didaktični pristop

Klasičen pristop uvodnih tečajev v fiziko temelji na predstavitvah poskusov in neposredni primerjavi rezultatov z abstraktnimi matematičnimi izrazi skupaj z laboratorijskim delom, ki navadno ni časovno in zaporedno usklajeno.
JxyZET kot simulacijsko orodje ali medij ponuja vmesno stopnjo med pravimi poskusi in abstrakcijo le-teh. S pomikom matematičnega dela na računalnik so lahko rezultati upodobljeni na ekranu, kar služi učencu za izgradnjo potrebne baze znanja kvalitativnih konceptov.
Potrebo po matematičnih orodjih in integracijo kvalitativnih in kvantitativnih metod z namenom, da zmanjšamo stopnjo kompleksnosti, bomo prikazali v nadaljevanju.

4.3. Osnove programa JxyZET

4.3.1. Univerzum

Univerzum v programu JxyZET je zaradi lažje implementacije algoritmov zajet v kocko in je predstavljen v treh dimenzijah. Osnovni primitiv tega univerzuma je prostorska točka. Ta točka vsebuje le naslednje podatke:

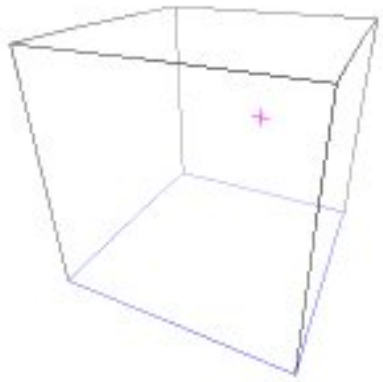
- x,y,z – vektor prostorskih koordinat.

4.3.2. Svet delcev

Naslednji korak je predstavitev posameznih delcev v našem univerzumu. Hierarhično gledano je vsak delec dedič prostorske točke in podeduje vse njene lastnosti. Delec je torej sestavljen iz:

- x,y,z – podedovano iz prostorske točke;
- vx,vy,vz – vektor hitrosti delca;
- fixed – je ta delec fiksiran ali ne;
- xforce,yforce,zforce – vektor sile, ki trenutno deluje na ta delec, v odvisnosti od okolja;
- charge – naboj delca;
- mass – masa delca;
- radius – polmer delca.

Poleg tega ima delec še mnogo drugih atributov, ki nam pridejo prav pri upodabljanju.



Slika 4.1: Upodobljen univerzum, ujet v kocko in en sam delec v njem

4.3.3. Povezave

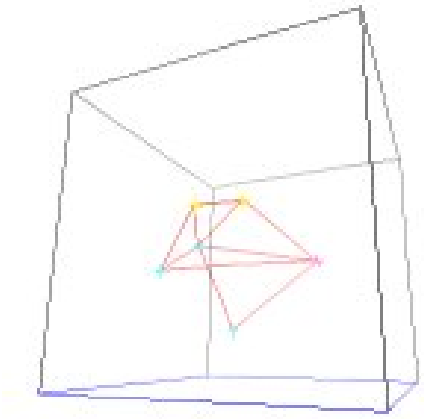
Naslednji primitiv, ki je pomemben za stabilnost sistema, je povezava. Razred povezave vsebuje naslednje spremenljivke:

- from,to – prostorski točki, ki sta med seboj povezani.

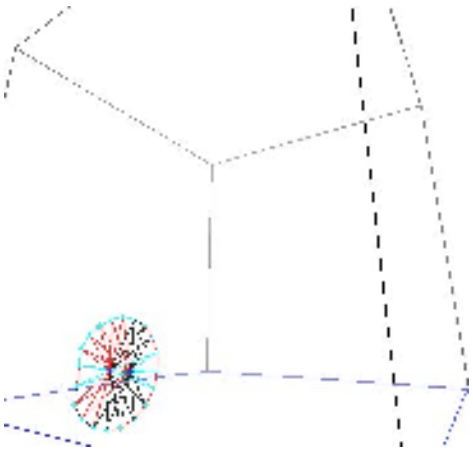
Definirana je tudi vzmet, ki podeduje lastnosti povezave in ima definiranih še nekaj spremenljivk:

- from,to – podedovano od povezave;
- length – dolžina vzmeti;
- const – konstanta vzmeti;
- force – sila v vzmeti.

Definirali smo primitive in nekaj njihovih naslednikov. Ti povsem zadoščajo, da lahko sestavimo bolj zapletene objekte, kot lahko vidimo na spodnji sliki.



Slika 4.2: Nekaj delcev, povezanih med seboj



Slika 4.3: Delci povezani z vzmetmi, ki predstavljajo kolo

4.3.4. Ostali elementi

Do sedaj smo opisali naš svet, primitive v njem in smo zgradili že nekaj kompleksnih objektov. Toda za čim boljše sovpadanje našega modela z realnim svetom je potrebna še množica dejavnikov, ki vplivajo na naš svet, na objekte v njem ter na njihovo obnašanje. Definirani so še:

- Coulombova konstanta;
- gravitacijska konstanta;
- zunanja gravitacija;
- zunanje električno polje;
- zunanje magnetno polje in
- viskoznost prostora.

Vsi ti dejavniki so izbirni, kar pomeni, da jih lahko tudi onemogočimo, če niso pomembni za naš model.

Ko smo zgradili naš model in smo podali lastnosti za vse naše delce, se lahko lotimo računalniške simulacije, s katero bomo pridobili potrebno znanje in izkušnje z interakcijo med izvajanjem modela.

4.4. Simulacija modela v programu JxyZET

Ko smo uspeli model načrtati, lahko sprožimo simulacijo. Simulacijo izvajamo po časovnih korakih s pomočjo dodeljevanja časa s časovnimi rezinami. Velikost rezin lahko spreminjamo in s tem vplivamo na hitrost izvajanja simulacije. Posledično z večanjem časovnih rezin se manjša natančnost simulacije, ker moramo upoštevati, da je natančnost števil s plavajočo vejico v računalnikih vedno omejena [12]. V našem svetu se v eni časovni rezni spremeni stanje sveta v odvisnosti od prejšnjega stanja in delovanja zunanjih vplivov. To lahko strnemo v naslednjo preslikavo:

$$f\left(\text{staro\_stanje}, \text{zunanji\_dejavniki}\right) \xrightarrow{\text{časovna rezina}} \text{novo\_stanje}$$

Poglejmo si bolj natančno, kaj se dogaja z objekti v našem modelu pri prehodu iz starega v novo stanje.
Recimo, da upoštevamo vse zunanje dejavnike, ki vplivajo na naš svet.

- Vsem delcem p_i ažuriramo njihove medsebojne sile v odvisnosti od zunanjega električnega polja extEF. Omeniti še velja, da operator * ne pomeni vektorskega množenja, ampak navadno množenje skalarnih vrednosti po komponentah.

$$\overrightarrow{p_i.force} = \overrightarrow{p_i.force} * \overrightarrow{extEF} \tag{4.1}$$

- Medsebojne sile delcev ažuriramo še glede na vpliv zunanjega gravitacijskega polja.

$$\overrightarrow{p_i.force} = \overrightarrow{p_i.mass} * \overrightarrow{extGravity} \tag{4.2}$$

- Za vsak par delcev p_i in pj izvedemo naslednje izračune:

  - najprej izračunamo vektorsko razdaljo med delcema p_i in p_j:
    $$\overrightarrow{dist} = \overrightarrow{p_j.pos} - \overrightarrow{p_i.pos} \tag{4.3}$$

  - izračunamo še kvadrat evklidske razdalje med obema delcema:
    $$sqrdist = (\overrightarrow{p_j.pos} - \overrightarrow{p_i.pos})^2 \tag{4.4}$$

  - izračunamo evklidsko razdaljo:
    $$dist = \sqrt{sqrdist} \tag{4.5}$$

  - normiramo vektor razdalje:
    $$\overrightarrow{dist} = \frac{\overrightarrow{dist}}{dist} \tag{4.6}$$

  - upoštevamo prispevek vzajemne gravitacije med delcema:
    $$f = \frac{mutGrav * p_i.mass * p_j.mass}{sqrdist} \tag{4.7}$$

  - upoštevamo še prispevek coulombove konstante:
    $$f = f + \frac{-coulConst * p_i.charge * p_j.charge}{sqrdist} \tag{4.8}$$

  - ažuriramo vrednosti na obeh delcih s prispevkoma vzajemne gravitacije in Coulombove konstante (4.8):
    $$\overrightarrow{p_i.force} = \overrightarrow{p_i.force} + f * \overrightarrow{dist} \tag{4.9}$$

    $$\overrightarrow{p_j.force} = \overrightarrow{p_j.force} - f * \overrightarrow{dist} \tag{4.10}$$

Upoštevali smo vse zunanje dejavnike, ki lahko prispevajo k spremembi medsebojnih sil pri delcih.

Naslednja faza so nasledniki veznih primitivov, vzmeti. Vzmeti prav tako prispevajo k spremembi medsebojnih sil med delci. JxyZET pozna več vrst vzmeti:

- Hookove;
- Hookove na potisk;
- Hookove na poteg;
- Van der Walls-ove in
- Lenard Jones-ove.

Ker so izračuni teh sil malce bolj zapleteni, si oglejmo le grob opis algoritma:

Za vsako od vzmeti {
upoštevaj tip vzmeti in izračunaj korekturni faktor f v odvisnosti od konstante vzmeti in njene dolžine.
Upoštevaj poleg konstante in dolžine še hitrosti obeh delcev, ki jih vzmet povezuje.
Ažuriraj medsebojne sile obeh delcev, ki jih vzmet povezuje.
}

Pri tem algoritmu so bile uporabljene naslednje fizikalne resnice:

- sila je enaka produktu mase in pospeška:
$$F = m * a \tag{4.11}$$

- pospešek je enak odvodu hitrosti po času:
$$a = \frac{dv}{dt} \tag{4.12}$$

- hitrost je enaka odvodu poti po času:
$$v = \frac{dx}{dt} \tag{4.13}$$

Sedaj smo dokončno opravili z medsebojnimi silami pri delcih in smo pripravljeni na izračun novih pozicij delcev v prostoru. Ta metoda je pri opisu z matematičnimi enačbami razmeroma nepregledna, zato bomo zopet podali psevdo algoritem, ki opisuje ta postopek:

Za vsakega od delcev {
shrani staro hitrost delca.
Izračunaj nove vrednosti hitrosti, upoštevajoč magnetno poljsko gostoto.
Pri hitrosti upoštevaj še viskoznost.
Izračunaj spremembe pozicije delca dx.
Preveri da nam pri novi poziciji ne uide iz našega univerzuma.
Če je delec znotraj kocke {
sprejmi njegovo novo hitrost,
}
če ne {
ga pa bodisi odbij od stene kocke, bodisi ga pusti na steni.
}
Sprejmi njegovo novo pozicijo.
}

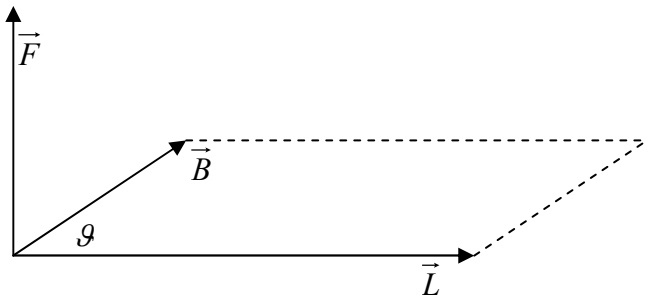
Pri premiku smo upoštevali tudi magnetno polje. Količina, ki jo lahko interaktivno spreminjamo v programu JxyZET, se imenuje magnetna poljska gostota [13]. Magnetna poljska gostota je vektor B, ki ima smer tangente na magnetno tokovnico. Magnetna sila deluje na nabite električne delce (npr. elektrone). Da bi spremenili pot nabitih delcev, mora magnetna sila delovati pravokotno na ravnino gibanja nabitih delcev. Oglejmo si naslednjo enačbo:

- Vektor sile je vektorski produkt med vektorjem dolžine in vektorjem magnetne poljske gostote, pomnožen s skalarjem toka:

$$\vec{F} = I * \vec{L} \times \vec{B} \tag{4.14}$$

- Če vektor in magnetna poljska gostota nista pravokotni, se vektorski produkt izračuna na način kot kaže 4.15.

$$\vec{F} = I * \vec{L} \times \vec{B} * \sin \vartheta \tag{4.15}$$



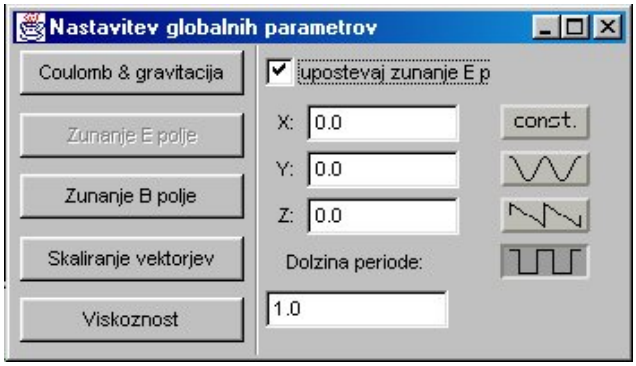
Slika 4.4: Vektorski produkt v primeru ko vektor dolžine in magnetna poljska gostota nista pravokotni.

Vse ostale uporabljene enačbe so podobne kot zgoraj.

Naslednji problem, ki se ga lotimo, je detekcija trkov delcev. Najbolj nazorno si bomo predstavljali problem, če implementacijo le-tega podamo v psevdo kodi:

Za vsak delec p_i {
za vsak delec p_j od i naprej {
izračunaj minimalno možno razdaljo med delcema p_i in p_j glede na njuno obliko.
Izračunaj dejansko razdaljo.
Če je dejanska razdalja < min. razdalje potem {
Imamo trk.
Ker vsak delec vsebuje tudi podatke o stari poziciji, ga postavimo nazaj.
Regresiramo vektor pozicije in upoštevamo odboj pri trku.
Popravimo vektor nove pozicije in vektor nove hitrosti.
}
}
}

Prišli smo že skoraj do konca našega simulacijskega koraka. Ostala nam je le še ena malenkost, ki je še nismo omenili.
Pri zunanjem električnem polju imamo tudi možnost izbire izmeničnega polja.



Slika 4.5: Različne možnosti izbire izmeničnega električnega polja

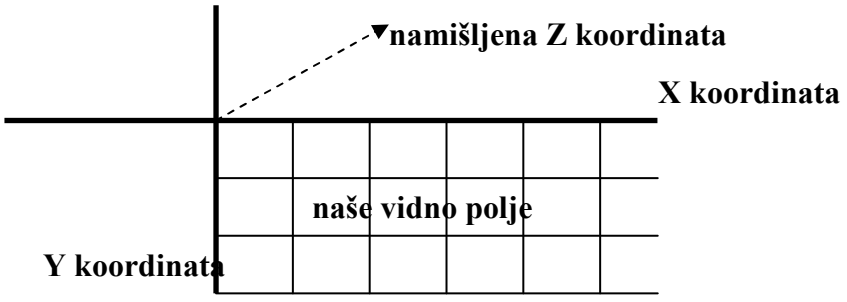
Na koncu ažuriramo še novo vrednost električnega polja glede na vrednost periode in obliko polja.
To je bil zadnji korak v eni časovni rezini naše simulacije. Kot smo že omenili, smo upoštevali vse dejavnike, ki vplivajo na potek simulacije. Sedaj so v vseh primitivih ali pa njihovih naslednikih že sveže vrednosti, ki čakajo na predstavitev uporabniku. Prišli smo torej do trenutka, ko bo potrebno celotno novo stanje upodobiti. Da bo uporabnik simulacijskega programa JxyZET čim bolje dojel predstavljeno informacijo, se bomo lotili 2.5D upodabljanja.

4.5. 3D upodabljanje v programu JxyZET

Kot osnovo za upodabljanje bomo vzeli našo kocko, ki omejuje naš univerzum. Na njej bomo izvedli vse potrebne operacije, katere nas bodo pripeljale do rezultata, ki je primeren za izris na zaslon.

4.5.1. Svet v treh dimenzijah

Kot smo omenili že v uvodnih poglavjih, dejansko vidijo človeške oči vedno le 2D. Z vrsto dodatnih informacij o sliki si zgradimo še tretjo, psevdo koordinato, kot to prikazuje spodnja slika [18].



Slika 4.5: Koordinatni sistem, ki si ga predstavljamo ljudje

4.5.2. Lokalni, globalni, opazovalni in zaslonski koordinatni sistem

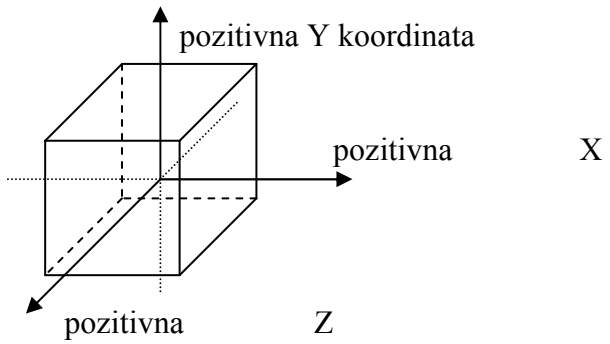
Če hočemo delati s tremi dimenzijami, se moramo prilagoditi na več koordinatnih sistemov. Pri tem ne mislimo na neke popolnoma drugačne koordinatne sisteme ampak na navadne desnoročno sučne sisteme z različnimi referenčnimi točkami. Še tako enostaven proces upodabljanja mora iti vsaj čez naslednje faze:

- preslikava 3D koordinat objekta v globalne 3D koordinate;
- preslikava globalnih 3D koordinat v opazovalne 3D koordinate kamere, ki opazuje naš svet in
- projekcija 3D opazovalnih koordinat v 2D zaslonske koordinate.

V tem trenutku je najboljše, da kar začnemo s praktičnim primerom, našo kocko, ki omejuje naš univerzum. Kocka je predstavljena z osmimi prostorskimi točkami v modelu, upodobljena pa bo z žičnim modelom [18].

Lokalne koordinate

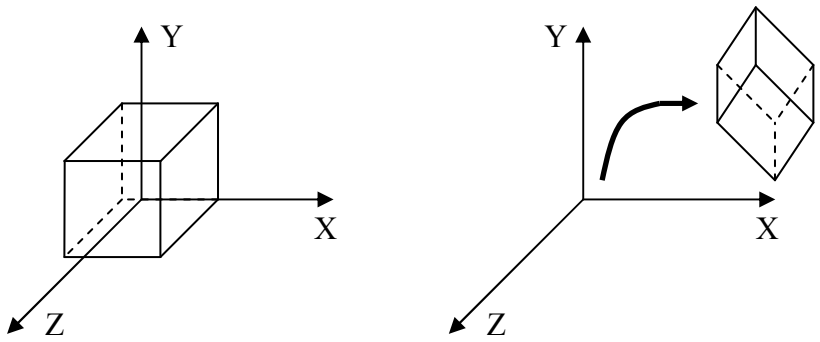
Osem oglišč kocke mora biti definiranih v nekem koordinatnem sistemu. Ker lahko kocka leži kjerkoli v prostoru, ki je večji od našega univerzuma, definiramo kocko v nekem lokalnem koordinatnem sistemu. Izhodišče koordinatnega sistema (0,0,0) postavimo v središčno točko kocke, kot kaže spodnja slika.



Slika 4.6: Naša kocka, centrirana v točko (0,0,0)

Globalne koordinate

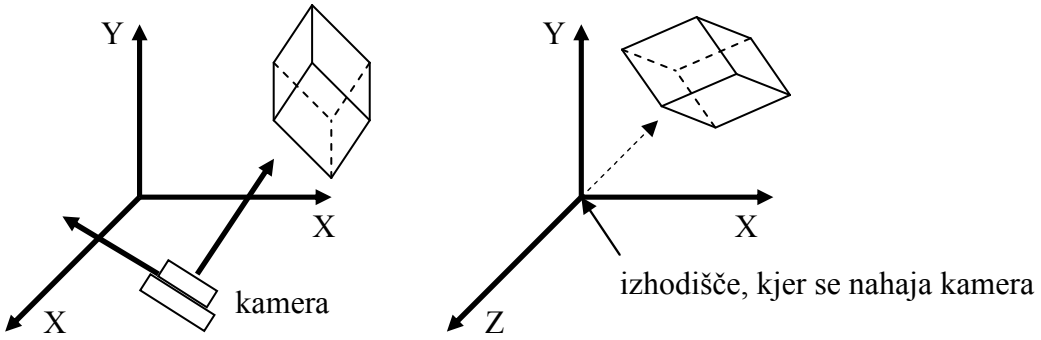
Predpostavimo, da se naša kocka nahaja nekje v svetu, na primer na lokaciji (15,100,-400). Predpostavimo še, da se je kocka zavrtela. Predstavljati si moramo, da kocko lahko vrtimo in premikamo okrog vsake od treh osi. Taka kocka mora biti zavrtena in premaknjena, če delamo preslikavo med lokalnim in globalnim koordinatnim sistemom, kot kaže spodnja slika:



Slika 4.7: Preslikava kocke iz lokalnih v globalne koordinate z vrtenjem in premikom

Opazovalne koordinate

Sedaj, ko kocka obstaja v globalnih koordinatah, bi jo želeli prikazati na ekranu v nekem oknu. Potrebujemo navidezno kamero, ki bo postavljena nekje v globalnih koordinatah in bo gledala v neko smer z nekim zornim kotom pogleda.
Ta postopek se navadno implementira v dveh korakih. Prvi korak je preslikava globalnih koordinat kocke v referenčni okvir kamere. Ta korak poenostavlja drugi korak, pri katerem projiciramo na 2D zaslon. Ustvarili bomo torej še opazovalni koordinatni sistem z izhodiščem, kjer se nahaja naša kamera, z *Z* osjo v smeri gledanja kamere. Sedaj lahko preslikamo globalne koordinate kocke v opazovalne koordinate kamere, kot to prikazuje spodnja slika.

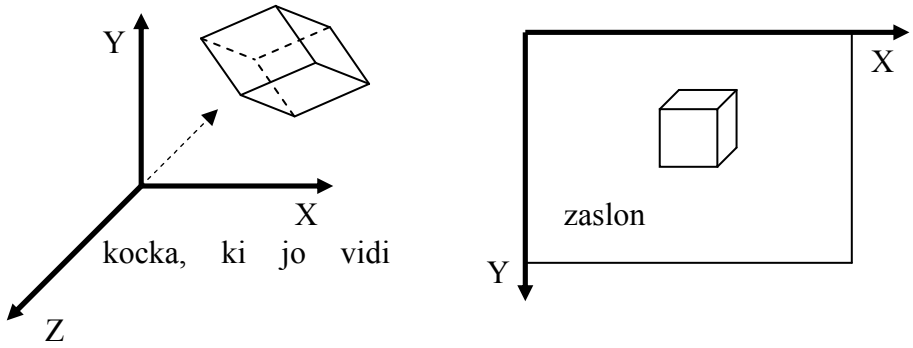


Slika 4.8: Preslikava iz globalnega koordinatnega sistema v opazovalnega

Zaslonske koordinate

Sedaj, ko imamo kocko v referenčnem koordinatnem sistemu kamere, je preslikava v zaslonske koordinate relativno enostavna. Tukaj je najpomembnejše deljenje koordinat *X* in *Y*

točk kocke z njihovo globino, koordinato *Z*. Pri tem moramo paziti na predznake pri računanju in na zasuk koordinate *Y*, katere pozitivna smer bo kazala v smeri navzdol na zaslonu, kot kaže spodnja slika.



Slika 4.9: Preslikava iz opazovalnih koordinat v zaslonske 2D koordinate.

Konceptualno smo opisali, kakšen problem nas čaka. Sedaj se lahko lotimo posameznih korakov in jih sproti matematično utemeljimo.

4.5.3. 2D upodobitev

Strnimo še enkrat vse potrebne korake, ki so potrebni za preslikavo v 2D zaslonske koordinate:

1. zasuk točke v želeno smer v globalnih koordinatah;
2. premik točke iz lokalnega koordinatnega sistema v globalnega;
3. premik točke iz globalnega koordinatnega sistema v opazovalnega;
4. zasuk točke okrog smeri kamere;
5. skaliranje točke glede na razmerje zaslonskih dimenzij in zornega kota kamere;
6. preslikava opazovalnih koordinat točke v zaslonske koordinate in
7. izris točke ali črte na ekran.

Prvi in drugi korak ustrezata transformaciji iz lokalnega v globalni koordinatni sistem. Za zasuk točke zgradimo sučno matriko (4.17), ki je sestavljena iz treh sučnih matrik okrog posameznih koordinatnih osi, kot kažejo enačbe (4.16).

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\sigma) & \sin(\sigma) \\ 0 & -\sin(\sigma) & \cos(\sigma) \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos(\nu) & 0 & -\sin(\nu) \\ 0 & 1 & 0 \\ \sin(\nu) & 0 & \cos(\nu) \end{bmatrix} \tag{4.16}$$

$$R_z = \begin{bmatrix} \cos(\vartheta) & \sin(\vartheta) & 0 \\ -\sin(\vartheta) & \cos(\nu) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{xyz} = \begin{bmatrix} \cos(\nu)\cos(\vartheta) & \cos(\nu)\sin(\vartheta) & -\sin(\nu) \\ \sin(\sigma)\sin(\nu)\cos(\vartheta) - \cos(\sigma)\sin(\vartheta) & \sin(\sigma)\sin(\nu)\sin(\vartheta) - \cos(\sigma)\cos(\vartheta) & \sin(\sigma)\cos(\nu) \\ \cos(\sigma)\sin(\nu)\cos(\vartheta) - \sin(\sigma)\sin(\vartheta) & \cos(\sigma)\sin(\nu)\sin(\vartheta) - \sin(\sigma)\cos(\vartheta) & \cos(\sigma)\cos(\nu) \end{bmatrix}$$

(4.17)

Za pomik pa uporabljamo navadno seštevanje vektorjev (4.18)

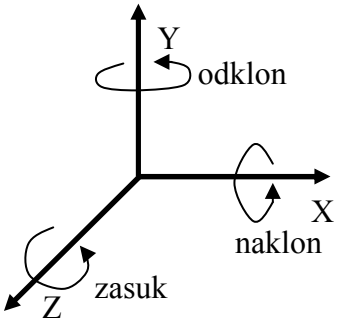
$$\begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} \tag{4.18}$$

pri čemer je vektor **r** rezultat, **p** pomični vektor in **s** vektor zasukane točke s pomočjo matrike (4.17). Pomični vektor **p** dobimo enostavno tako, da izračunamo razdaljo med izhodiščema lokalnega in globalnega koordinatnega sistema.

Koraki od 3 do 6 preslikujejo globalne koordinate v opazovalne koordinate in potrebujejo referenčni okvir kot temelj za opazovalne koordinate. Ker gledamo skozi “objektiv” kamere, je prav, da spregovorimo še nekaj besed o lastnostih naše navidezne kamere.

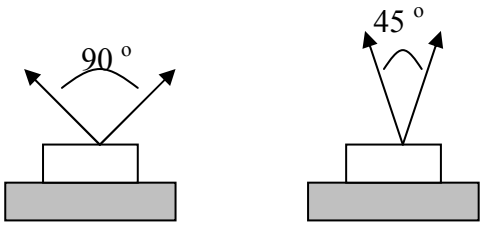
Navidezna kamera

Obstaja veliko načinov, s katerimi bi lahko pojasnili navidezno kamero. Kamera vsekakor potrebuje lokacijo v 3D globalnem koordinatnem sistemu tako, da ima relativno lokacijo glede na ostale objekte [18].
Prav tako kamera potrebuje smer. Smer gledanja kamere definiramo s pomočjo treh parametrov: naklona, zasuka in odklona. Naklon je vrtenje okrog *X* osi, odklon je vrtenje okrog *Y* osi in zasuk je vrtenje okrog *Z* osi. Slika 4.10 prikazuje našo navidezno kamero.



Slika 4.10: Navidezna kamera v globalnih koordinatah

S kamero še nismo povsem zaključili. Obstaja še parameter, ki ga bomo uporabili, in sicer zorni kot gledanja. Ta parameter nam pove, s kakšnim zornim kotom kamera opazuje svet. Pri večjih kotih hoče kamera preslikati na zaslon večji prostor (glej sliko 4.11).



Slika 4.11: Dve kameri z različnim zornim kotom gledanja

Sedaj se lotimo preslikave globalnih koordinat v opazovalne. Če se vrnemo nazaj na seznam sedmih korakov, postorimo naslednje:

- V tretjem koraku odštejemo koordinate lokacije kamere od naše prostorske točke, ki jo preslikujemo.
- V četrtem koraku zasukamo našo točko v nasprotni smeri, kot nam kažejo naklon, odklon in zasuk kamere. Ker smo tri sučne matrike združili, nam je potreben le en matrični račun.
- Peti korak transformira našo točko in vse druge, z upoštevanjem zornega kota kamere in razmerja koordinat *X* in *Y,* tako da ne dobimo popačenj slike vzdolž *X* ali *Y* osi.

Prišli smo končno do šestega koraka, ki preslika opazovalne koordinate v zaslonske. Naše platno, kamor bomo slikali, je definirano z višino in širino, tako da upoštevamo pri preslikavi tudi ta dva parametra. Ta korak je matematično preprost, saj enostavno delimo vsako *X* in *Y* koordinato točke s pripadajočo koordinato *Z*, katere namen je projekcija *X*, *Y* koordinat na ekran in povzroči, da so bolj oddaljeni objekti videti manjši.

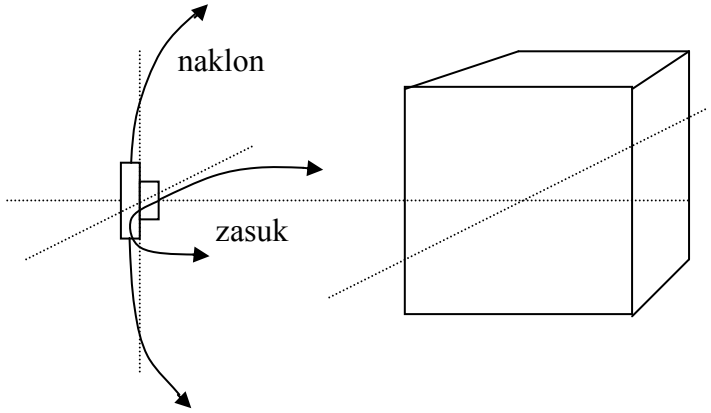
$$X_{zaslon} = \frac{X_{opazovalna}}{-Z_{opazovalna}} * širina_{platna} + \frac{širina_{platna}}{2} + 0.5$$

$$Y_{zaslon} = višina_{platna} - \frac{Y_{opazovalna}}{-Z_{opazovalna}} * višina_{platna} + \frac{višina_{platna}}{2} + 0.5 \quad (4.19)$$

Formula (4.19) je malce bolj zapletena, kot smo pričakovali. Predznak koordinate *Z* mora biti obrnjen, ker smo definirali negativni del *Z* osi pred kamero. *Y* koordinato pa moramo obrniti, da ustreza našemu zaslonskemu koordinatnemu sistemu.

Opozoriti je še treba na dogodek, ko imamo *Z* koordinato v opazovalnem koordinatnem sistemu večjo od 0, z drugimi besedami to pomeni, da se naša opazovana točka nahaja izza kamere. Preslikav takih točk se moramo enostavno izogniti. Če preslikujemo cel poligon ali samo črto, jo moramo prikrojiti na tistem mestu, kjer nam preostanek izgine za kamero.

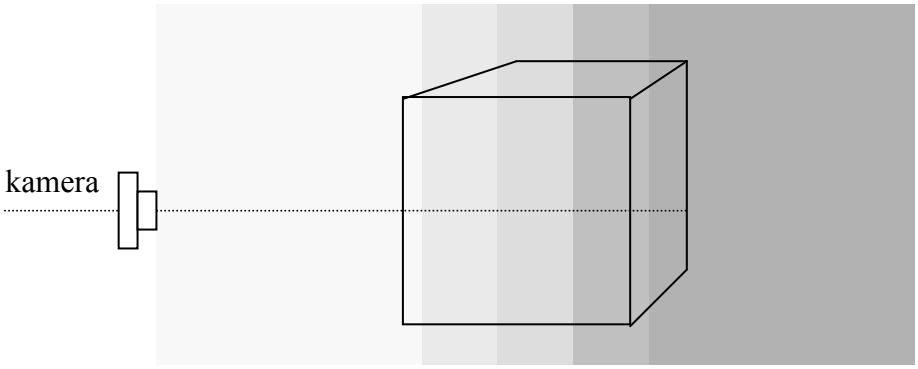
Naj omenimo še, da se v sistemu JxyZET gibljemo s kamero okrog kocke le v dveh smereh in nam s tem odpade potreba po vrtenju kamere z odklonom. Nazorno gibanje kamere je predstavljeno na spodnji sliki 4.12.



Slika 4.12: Pri JxyZET sta pri kameri možna le naklon in zasuk

4.5.4. 2.5D upodabljanje

Človek je torej sposoben iz slike v dveh dimenzijah iz realnega sveta zbrati toliko informacij, da si v mislih ustvari tretjo, navidezno dimenzijo. Pri računalniškem upodabljanju se prav tako skušamo približati preslikavi, ki bi bila za človeka čim bolj prostorska. S študijo človeškega dojemanja sveta smo že sposobni izvleči nekaj atributov pri sliki, ki pozitivno delujejo na predstavo prostora v 3D.
Pri zgornji upodobitvi v 2D smo že uporabili enega od pristopov, in sicer deljenje *X* in *Y* koordinat s koordinato *Z*. Pri tem postopku dobimo občutek, da so manjši objekti bolj oddaljeni, večji pa bližji. Pri JxyZET smo uporabili še en postopek, ki močno vpliva na človeško dojemanje projekcije našega univerzuma v kocki, to je senčenje z globino.
Našo kocko v opazovalnem koordinatnem sistemu razdelimo na pet segmentov senčenja. Segment, ki je najbližji kameri, dobi najsvetlejšo barvo. Intenzivnost barv potem z globino pada, tako da ima najbolj oddaljen segment najtemnejšo barvo. (slika 4.14)



Slika 4.13: Opazovalni koordinatni sistem razdeljen na pet segmentov intenzivnost barv.

Naš univerzum je sedaj upodobljen tako, da daje uporabniku dokaj pristen občutek globine. Naj omenimo še, da je JxyZET interaktivno orodje in uporabniku daje možnost gibanja po prostoru, kar je tudi pomemben prispevek k pravemu dojemanju prostora.

Naša kocka, točke in črte med njimi so sedaj upodobljene z možnostjo dojemanja v 2.5 dimenzijah. Lotili se bomo še dveh tem, ki sta obravnavani in implementirani v programu JxyZET in omogočata uporabniku profesionalen pristop do simulacije fizikalnih poskusov. To sta iskanje in upodabljanje silnic električnega polja ter iskanje in upodabljanje ekvipotencialnih ploskev in predstavljata matematično najzahtevnejši del tega produkta.

4.6. Silnice električnega polja

4.6.1. Teoretična podlaga

V vsaki točki prostora v okolici električnega naboja deluje električna sila na naelektrena telesa. Pravimo, da se v okolici električnega naboja razprostira polje električne sile ali električno polje. V vsaki točki električnega polja deluje električna sila na naboje, ki so v polju. Jakost električnega polja v neki točki definiramo s pomočjo sile, s katero polje učinkuje na pozitivni naboj, ki je v tisti točki. Električna sila na naboj je tem večja, čim večji naboj e vstavimo v polje [9, 13]:

$$\vec{F} = e * \vec{E} \tag{4.20}$$

Sorazmernostno konstanto med električno silo in nabojem e imenujemo jakost električnega polja.
Električno polje ponazorimo s silnicami. Silnice so črte, katerih tangente povedo smer delovanja električne sile na pozitivni naboj. Pozitivni naboji se pod vplivom električne sile pospešujejo v smeri silnic, negativni pa v nasprotni smeri. Električno poljsko jakost v okolici točkastega naboja lahko takoj določimo, ker poznamo Coulombov zakon (4.21).

$$\vec{F} = \frac{e_1 e_2}{4\pi\varepsilon_0 r^2} \tag{4.21}$$

$$\vec{E} = \frac{\vec{r}}{r} * \frac{e_1}{4\pi\varepsilon_0 r^2} \tag{4.22}$$

Smer električne sile **F** izrazimo z enotskim vektorjem $\frac{\vec{r}}{r}$.
Če imamo opravka z več naboji, na primer $e_1, e_2, ..., e_n$, ki so od točke, kjer določamo električno poljsko jakost **E**, oddaljeni za $r_1, r_2, ..., r_n$, moramo za vsak naboj posebej ugotoviti smer in velikost električne sile. Električne sile vseh nabojev vektorsko seštejemo in dobimo rezultanto **F**, ki je e**E**. Električna poljska jakost se torej izraža:

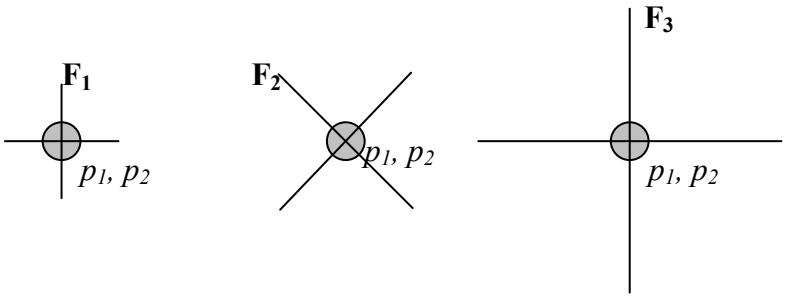
$$\vec{E} = \frac{1}{4\pi\varepsilon_0} \sum \frac{e_i \vec{r_i}}{r_i^3} \tag{4.23}$$

Toliko za zdaj o teoriji iz fizike. Sedaj se lotimo implementacije v sistem JxyZET.

4.6.2. Realizacija pri simulaciji v JxyZET

Najprej se moramo odločiti, kako bomo iskali silnice električnega polja, katere od njih bomo izbrali, ker jih je neskončno tudi samo za eno samo vrednost električne sile **F,** in s kakšno vrednostjo silnice bomo sploh iskali.
Najprej razložimo zamisel, katere silnice bomo računali. Iskali bomo 12 silnic, od katerih imajo po 4 silnice enako velikost sile **F**. Iskali bomo torej silnice za 3 različne vrednosti (**F**$_1$,

F_2, F_3) velikosti sile **F**. Vzemimo si za primer 2 delca p_1 in p_2. Na spodnji sliki je ideja že bolj jasna. Skupaj bomo imeli 12 silnic, ki jih predstavljajo črte, ki sekajo delce. Sorazmerna je tudi velikost sile.



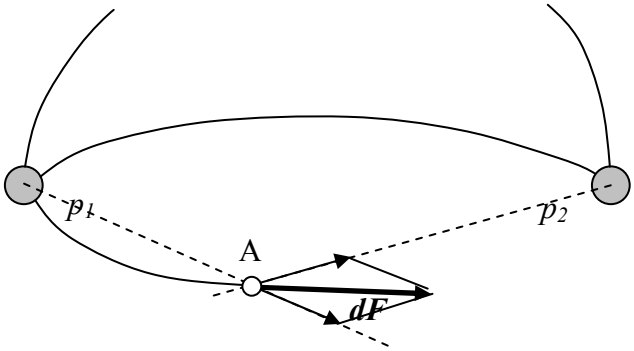
Slika 4.14: 12 silnic električnega polja, ki bodo upodobljene.

Naj opozorimo, da se delca p_1 in p_2 prekrivata in med njima obstaja neka razdalja, večja od 0.

Ker je algoritem zelo zapleten, ga bomo spet razložili le s psevdo kodo:

Za vsakega od delcev {
izračunaj začetne točke začenši pri trenutnem delcu za vseh 12 silnic.
V zanki računaj črto silnice.
}

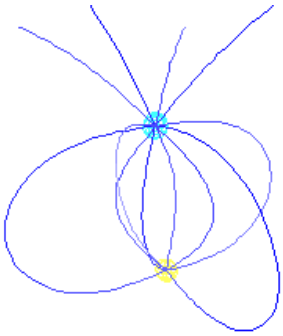
Kako pridemo do začetnih točk naših silnic, nam je jasno že iz slike 4.14 Čaka nas izračun celotne silnice, ki poteka tako kot to vidite na sliki 4.15.



Slika 4.15: Izračun silnice električnega polja.

V točki A izračunamo prispevke sil vseh delcev, izračunamo njihovo rezultanto, seveda vektorsko. Sila **F** kaže v smeri tangente na silnico in po tej tangenti se premaknemo samo za **dF**. Potem postopek ponovimo in, ker se pomikamo za **dF,** lahko zletimo dol iz prave silnice s pravo vrednostjo. Zato moramo pri vsakem koraku tudi preveriti, do kakšne napake je prišlo. Če je ta napaka manjša od nekega ε, nadaljujemo, sicer pa z metodo bisekcije spet najdemo pravo pot.

Ko izračunamo vseh 12 silnic za vse delce, jih lahko upodobimo na že opisan način in dobimo za rezultat nekaj takega:



Slika 4.16: Upodobljene silnice za dva delca enako velikih, a nasprotno predznačenih nabojev.

4.7. Ekvipotencialne ploskve

4.7.1. Teoretična podlaga

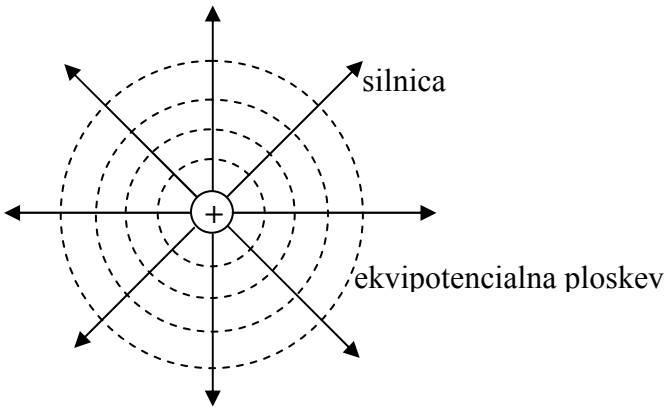
Spoznali smo, da električna sila premika proste naboje v električnem polju in pri tem opravlja delo. Zanima nas, koliko dela opravi električna sila **F** = *e***E**, če premakne pozitivni naboj iz točke 1 do točke 2 [13]:

$$A = \int_1^2 \vec{F} ds = e \int_1^2 \vec{E} ds \tag{4.24}$$

Delo električne sile je odvisno le od začetne in končne lege naboja v polju, ne glede na obliko pretečene poti med obema legama. To pomeni, da je električna sila konservativna sila, električno polje pa potencialno polje.
Električni naboj ima v električnem polju določeno potencialno energijo (električna energija W_e) in ta električna energija je premosorazmerna naboju delca:

$$W_e = eV \tag{4.25}$$

Sorazmernostni faktor V imenujemo električni potencial in je odvisen od mesta v električnem polju. Električni potencial v določeni točki polja pove, koliko električne energije (W_e) ima enota pozitivnega naboja v tisti točki.
Električna sila opravi pri premiku naboja delo na račun zmanjšanja električne energije naboja. Pozitivni naboj se pod vplivom električne sile premakne na mesto z manjšim električnim potencialom. Sledi, da se električni potencial v smeri silnic zmanjšuje.
Če povežemo sosednje točke, ki imajo enak potencial dobimo ploskev, ki jo imenujemo ekvipotencialna ploskev. Električna energija naboja se vzdolž ekvipotencialne ploskve ne spreminja torej je električno delo premikanja naboja vzdolž ekvipotencialne ploskve enako nič. Prav tako sledi, da so električne silnice pravokotne na ekvipotencialno ploskev, kot to kaže slika 4.17.



Slika 4.17: Ekvipotencialne ploskve, pravokotne na električne silnice

4.7.2. Realizacija v JxyZET

Problem upodabljanja ekvipotencialnih ploskev razdelimo podobno kot pri električnih silnicah na dva koraka:

- iskanje začetne točke in
- risanje ekvipotencialne črte, ki je del ploskve.

V JxyZET si delo olajšamo na ta način, da za razliko od električnih silnic celoten prostor razdelimo na ploskve po vseh treh ravninah (XY, XZ in YZ). Interaktivno lahko izbiramo, kako na gosto želimo imeti te ploskve, v katerih ravninah ter kakšen potencial nas zanima.
Začetni algoritem ob predpostavki, da hočemo upodobiti ekvipotencialno ploskev v vseh treh ravninah, je videti tako:

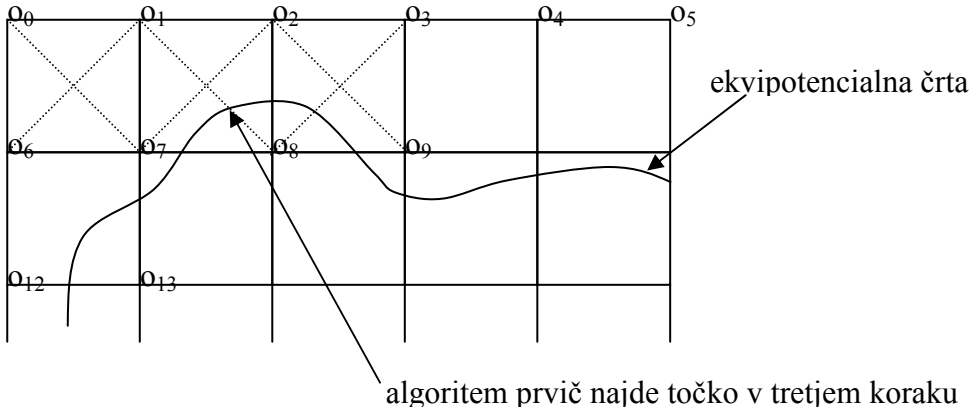
Za vsako od ravnin r_j, j=0..2{
razdeli kocko na n ploskev v ravnini r_i.
Za vsako ploskev p_{ij}, i=0..n {
izračunaj začetno točko in
povleci črto, ki je del ekvipotencialne ploskve
}
}

Iskanje začetne točke

Ko smo izbrali ploskev p_{ij} v ravnini r_j, lahko začnemo z iskanjem začetne točke, ki bo vsebovala želeni potencial V. Ploskev p_{ij} zopet razdelimo na šahovnico 5 krat 5 polj in tako dobimo 6 krat 6 oglišč šahovnice o_k, k=0..35. Sedaj postopamo na naslednji način:

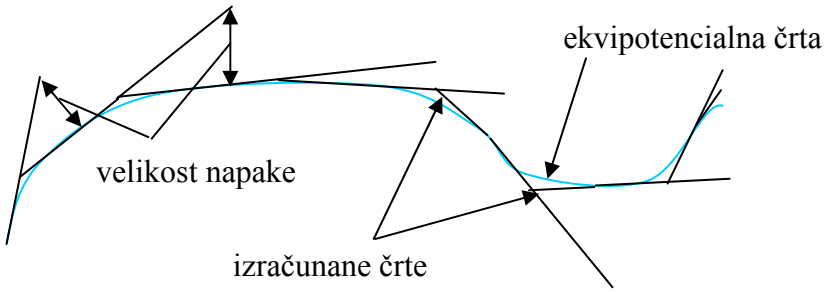
Za vsako oglišče o_k, k=0..29 {
preveri če obstaja želeni potencial na diagonali med točkama o_k in o_{k+7} ter med točkama o_{k+1} in o_{k+6} z uporabo hitre bisekcije.
Če najdeš začetno točko potem {
povleci ekvipotencialno črto v eno smer iz začetne točke in nato še v drugo smer.
}
}

Spodnja slika prikazuje princip iskanja začetne točke za začetek upodabljanja črt, ki sestavljajo ekvipotencialno ploskev.

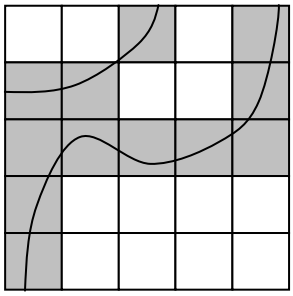


Slika 4.18: Prikaz delovanja algoritma iskanja začetne točke.

Ko smo našli začetno točko, se lotimo upodabljanja ekvipotencialne črte najprej v eno smer in nato še v drugo. Tvorba črte poteka podobno kot pri električnih silnicah in se pomikamo vzdolž ploskve za majhen Δ v smeri tangente na ploskev. Ob napaki, večji od nekega ε, korigiramo s pomočjo bisekcije, kar lahko vidimo na sliki 4.20. Ko z našo črto prečimo posamezne dele šahovnice, jih sproti označujemo, tako da nam ni treba iskati iste črte še enkrat ali večkrat (slika 4.21).



Slika 4.19: Risanje ekvipotencialne črte s prikazom napak pri računanju.

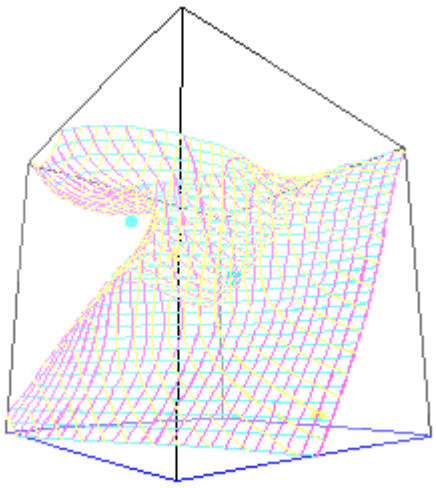


Slika 4.20: Računanje ekvipotencialne črte in označevanje šahovnice za kasnejše iskanje začetnih točk.

Ko izračunavamo naslednje točke, vzamemo zelo majhen korak. Zato slika 4.21 niti približno ne sovpada z dejanskim stanjem. Njen namen je bralcu prikazati, kaj se lahko zgodi pri tangentnih metodah in na kakšen način se temu izogniti.
Zgoraj je omenjena tudi hitra bisekcija, katere delovanje lahko prikažemo na primeru:

- leva meja je 1, desna pa 1000, iščemo pa točko kjer je vrednost 990;
- če bi delali po klasični bisekciji, bi trajalo zelo dolgo, da bi prišli do rešitve;
- pri hitri bisekciji pa razdelimo interval v sorazmerju z levo in desno mejo ter iskano vrednostjo in
- če je interval linearen, imamo zadetek že v prvem koraku.

Iskanje in upodabljanje ekvipotencialnih je zelo računsko intenzivno in časovno potratno. Tu se pokaže slabost izvajanja javanskih programov, ker imajo pod sabo navidezni stroj in vsak ukaz se tolmači, ne pa prevaja. Primer upodobljene ekvipotencialne ploskve v programu JxyZET najdemo na spodnji sliki.



Slika 4.21: Primer 6 delcev in ekvipotencialne ploskve s potencialom 0.

5. Zaključek

Na projektu CoLoS (Conceptual Learning of Science ali konceptualno učenje znanstvenih ved) sem sodeloval več let. Predstaviti sem hotel zaključeno in celovito orodje pri učenju fizike, ki je zlahka prenosljivo in zraven nudi širok spekter poskusov iz mehanike in elektrike. Kot sem že omenil, so v okviru projekta CoLoS pripravljeni celoviti tečaji iz omenjenih dveh poglavij iz fizike. Diplomska naloga skuša bralcu predstaviti teoretično ozadje simulacij in upodabljanja ter v razlagi samega produkta JxyZET, celotno stvar še podkrepiti z matematičnimi in fizikalnimi modeli.

JxyZET ni zaključena celota, ampak sistem, ki je potreben vzdrževanja v smislu nadaljnjega razvoja. V svetu poznamo veliko ekip, ki se ukvarjajo s podobnimi problemi, zato bi bilo smiselno iskanje skupnih značilnosti. Ker gre za akademski projekt, ne bi bilo odveč produkta licencirati kot odprtokodnega pod npr. GNU GPL (General Public License) licenčnim modelom.

Poudariti je potrebno uporabnost samega sistema JxyZET. Želja vsakega snovalca nekega produkta je, da bi se produkt s pridom uporabljal v praksi. JxyZET bi bilo potrebno predstaviti širši strokovni javnosti na različnih dogodkih kot npr. poletna šola, CoLoS dogodki, ipd. Prav tako bi bilo treba prikrojiti vsebino posameznih tečajev vsebini, ki je trenutno aktualna na naših šolah.

Seznam slik

Viri

[1] The Oxford English Dictionary, letnik 1989

[2] Human Depth Perception, http://www.cartage.org.lb/en/themes/arts/painting/principl-tech/human-depth/humdepth.htm, enciklopedija Cartage, Libanon

[3] Barta B.Z., Eccleston j., Hambush R., IFIP Transactions A-35 Computer mediated Education of Information technology professionals and advanced End-Users, Amsterdam, 1993

[4] Christian W., Esquembre F., Modeling Physics with Easy Java Simulations, THE PHYSICS TEACHER, Vol. 45, November 2007

[5] Cunningham S. Hubbold R.J., Interactive Learning through Visualisation, Springer, 1992

[6] Daconta M., Java 1.2 and JavaScript for C and C++ programmers, John Willey, 1998

[7] Dix A., Time, Space and Interaction, www.comp.lancs.ac.uk/~dixa/papers/FADIVA/ FADIVA, School of Computing, Staffordshire University, 1998

[8] Fazarinc Z., Computer Simulation in Physics, IEEE potentials, 1990

[9] Fazarinc Z., Computation of Electric Fields from Electric Charges: Coulomb's Law or Poisson's Equation, CAEE, 1994

[10] Fishwick P., Theory of Simulations, http://www.cis.ufl.edu/~fishwick/, 1995

[11] Härtel H., Martin E., New path for teaching electricity. Proceedings of the "International Conference on Compuetr Based Learning, CBLIS, Dunaj, 1993

[12] Kodek D., Arhitektura računalniških sistemov, Bi-TIM, 1994

[13] Kladnik R., Osnove Fizike 2, DZS, 1994

[14] Kononenko I., Strojno učenje, FE in FRI, 1997

[15] Kožuh V., Simulacije z računalnikom pri pouku fizike v osnovni šoli, Diplomska seminarska naloga, Univerza v Mariboru, 1999

[16] Lichtfeld M., Ideen fuer den Physikunterricht, Berlin, 1993

[17] New Network-Based Media in Education, Proceedings of the International CoLoS Conference, Maribor 1998

[18] Willey J., Three-Dimensional Math and Polygon Rendering, John Willey, 1996