SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1092

SIMULACIJA SUDARA KRUTIH TIJELA

Igor Poljak

Zagreb, siječanj 2010.

1. Sadržaj

2. Uvod

Sve realniji grafički prikaz zahtjeva sve bolju i bržu fizikalnu simulaciju. Za simulacije koje se izvode u realnom vremenu brzina izvođenja važnija je od točnosti simulacije. Porastom brzine računala u računalnim igrama postaje moguća sve realnija fizička simulacija. Vremenski vrlo zahtjevan dio fizičke simulacije je detekcija sudara, te je na njoj poželjno obaviti razne optimizacije. U prvom dijelu rada opisane su metode detekcije sudara u dvije faze. U prvoj se primjenjuju jednostavni testovi koji u posebnim slučajevima mogu dati konačni rezultat. U drugoj fazi obavljaju se skupi i vremenski zahtjevni testovi. U drugom dijelu rada određene su formule za izračun jednostavne reakcije na sudar koristeći dobivene točke sudara i normalu, te početne brzine objekata. Na trodimenzionalnim primjerima ocijenjen je utjecaj različitih parametara.

3. Detekcija sudara

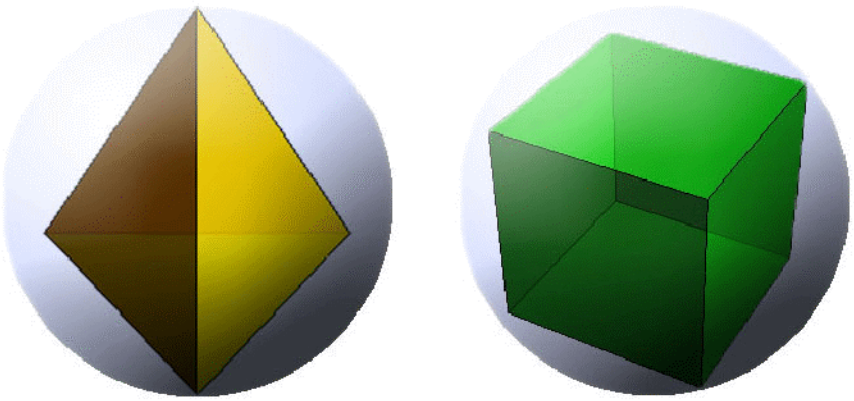
Detekcija sudara uključuje algoritme za određivanje presjeka dva dana objekta. Ti algoritmi moraju odrediti točke sudara i normalu između svih parova objekata koji se sudaraju. Kod složenih objekata detekcija sudara vrlo je vremenski zahtjevna. Zato se ona obično obavlja u dvije faze. Prvo se obavljaju brzi i jednostavni testovi kojima se eliminiraju potencijalni sudari, a zatim vremenski zahtjevni.

3.1. Široka faza

Glavni cilj široke faze detekcije sudara je izbjegavanje izvođenja skupih testova za tijela koja su ionako daleko jedno od drugog. Tijela se u ovoj fazi obuhvaćaju obujmicama. Primjeri obujmica su:

- Kugla
- Kvadar paralelan s koordinatnim osima
- Općeniti kvadar
- Konveksna ljuska

Obujmice služe za aproksimaciju složenog predmeta jednostavnijim predmetom radi lakšeg ispitivanja presjeka. Što je obujmica složenija to bolje aproksimira predmet, ali raste složenost testova presjeka. Ako se obujmice ne sijeku tada očito ne moramo provjeravati ni njihov sadržaj. U ovom radu kao obujmice su korištene kugle jer je test za njih najjednostavniji. Potrebno je samo izračunati udaljenost između dva objekta. Ako je ta udaljenost veća od zbroja polumjera dvaju obujmica, tada se obujmice ne sijeku. Na slici 2.1 prikazan je primjer aproksimacije tetrahedrona i kocke sfernim obujmicama.

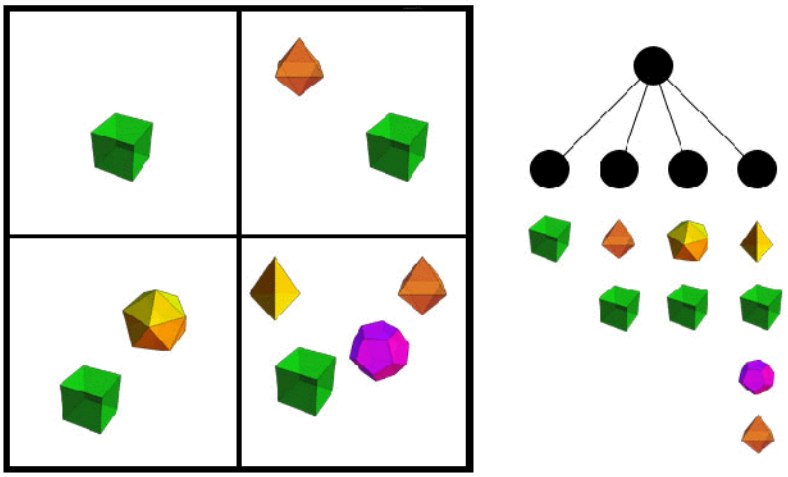


Slika 2.1 Jednostavni objekti aproksimirani obujmicama

Kod klasične metode detekcije sudara za svaki od n objekata u sceni provjerava se dali je u sudaru sa nekim od ostalih n-1 objekata. Ukupan broj potrebnih testova je $\frac{n(n-1)}{2}$. Čak i kada se koriste obujmice za aproksimaciju predmeta kod većeg broja objekata u sceni ova metoda

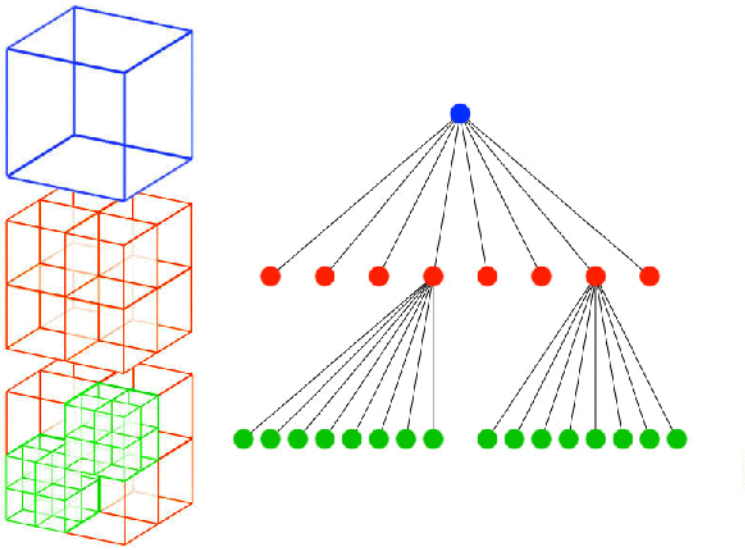
postaje neprimjenjiva za simulacije u realnom vremenu. Već kod 100 objekata u sceni potrebno je obaviti gotovo 5000 testova u svakom koraku simulacije. Kako ne bismo morali testirati sve parove objekata u sceni objekti se organiziraju u prostorne strukture podataka.

Na slici 3.2 na 2D primjeru prikazana je podjela prostora na četiri dijela. Klasičnom metodom za 9 objekata bilo bi potrebno obaviti 36 testova. Nakon podjele prostora za potencijalne parove sudara uzimaju se samo oni koji se nalaze u istom dijelu, te je sada potrebno obaviti samo 8 testova. Kada bismo donji desni dio još jednom rekurzivno podijelili, broj potencijalnih testova bio bi još manji. Ipak, kad u jednom dijelu ostane dovoljno mali broj objekata, jednostavnije je obaviti testove presjeka za sve objekte nego nastaviti dijeliti prostor.



Slika 3.2 Podijela 2D prostora

U ovom radu korišteno je oktalno stablo. Oktalno stablo rekurzivno dijeli 3D prostor na osam oktanata tj. čvorova. Svaki čvor u stablu predstavlja dio prostora, čvorovi bez djece su listovi i oni sadrže podatke o tome koji objekti se nalazi u tom dijelu prostora. Na slici 3.3 prikazan je primjer dijeljenja 3D prostora na oktane, i pripadajuće stablo. Ako neki objekt presijeca više oktanata informacija o njemu će biti spremljena u svaki od tih oktanata. Daljnji testovi presjeka vrše se samo za objekte koji se nalaze u istom oktanu, što znatno smanjuje broj potrebnih testova. Ipak prostor ne možemo dijeliti beskonačno, jer kad bi oktani postali manji od objekata, objekti bi se počeli pojavljivati u mnogo oktana, te bi zapravo počeli povećavati broj potrebnih testova. Zbog toga je potrebno ograničiti dubinu stabla.



Slika 3.3 Podijela 3D prostora

3.2. Uska faza

U uskoj fazi obavljaju se najskuplji i najpreciznij testovi, odnosno testovi presjeka za pojedine poligone objekata. Kruti objekti na računalu najčešće se prikazuju mrežom trokuta. Za svaki objekt definira se lista vrhova i lista trokuta. Pošto su obijekti sastavljeni isključivo od trokuta potrebno je obaviti testove presjeka za sve trokute objekata koji se potencijalno sudaraju. Test presjeka za trokute obavlja se pomoću sljedećeg algoritma. Trokut T_1 nalazi se u ravnini π_1 a trokut T_2 u ravnini π_2.

Ravnina π_2 ima sljedeću jednadžbu:

$N_2 \cdot X + d_2 = 0$

X je bilo koja točka na ravnini, a N_2 i d_2 se računaju po sljedećim formulama:

$N_2 = (V_{21}-V_{20}) \times (V_{22}-V_{20})$

$D_2 = -N_2 \cdot V_{20}$

Prvo se izračunava udaljenost svih vrhova T1 od ravnine π2.

Udaljenost točke od ravnine računa se po sljedećoj formuli:

$d_{vi} = N_2 \cdot V_{1i} + d_2$, i=0,1,2

Ako su sve 3 udaljenosti istog predznaka, i nijedna nije jednaka nuli sigurno nema presjeka jer ravnina trokuta T_2 uopće ne siječe trokut T_1. Detektor javlja da nema presjeka i ne izvršava daljnje testove.

Ako trokut T_1 siječe ravninu π_2 isti test ponavlja se za trokut T_2 i ravninu π_1.

Nakon toga izračuna se pravac L koji je presjek ravnina π_1 i π_2.

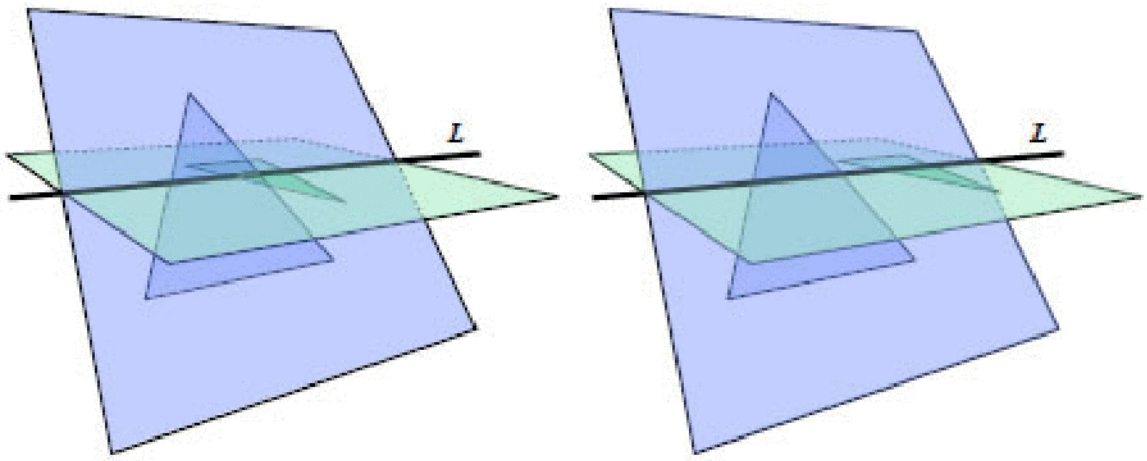
$L=O + tD$

Gdje je $D=N1 \times N2$, a O jedna točka na tom pravcu.

Radi ubrzavanja proračuna pravac L projecira se na os s najvećom komponentom. Vrhovi trokuta se zatim projeciraju na pravac L. Presjeci trokuta T_1 i T_2 sa linijom L pronalaze se iz sličnosti trokuta.

$$I_0=v_2+(v_0-v_2)*d_{v2}/(d_{v2}-d_{v0})$$
$$I_1=v_2+(v_1-v_2)*d_{v2}/(d_{v2}-d_{v1})$$

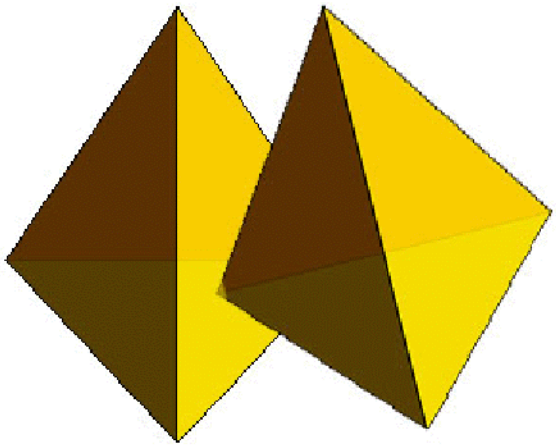
Gdje su v_0 i v_1 projekcije vrhova s jedne strane pravca L, a v_2 projekcija vrha sa suprotne strane. Pronalaze se intervali za oba dva trokuta, te ako se ta dva intervala preklapaju tada se dva trokuta sijeku. Na slici 3.4 prikazana su dva moguća slučaja. U prvom se intervali sijeku, a u drugom nema presjeka.



Slika 3.4 Trokuti sa pripadajućim ravninama

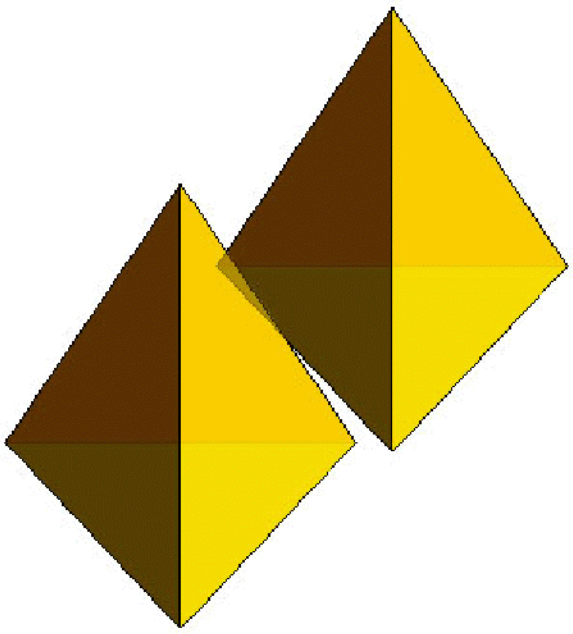
3.3 Točke sudara i normala

Koko bi mogli odrediti reakciju na sudar, u detekciji sudara moraju se odrediti dvije točke na objektima koje su se sudarile, te normala sudara. Zbog pojednostavljenja računa u simulaciji se zanemaruju slučajevi sudara koji su u realnim uvjetima gotovo nemogući, te se dešavaju jako rijetko ili se uopće ne dešavaju. To su slučajevi sudara dva vrha, vrha i brida, dva trokuta ili brida i trokuta. Ako se ti sudari i dogode u simulaciji će biti riješeni kao jedan od dva najčešća slučaja, a to su sudar vrha i trokuta, te sudar dva brida.



Slika 3.5 Sudar vrha jednog i trokuta drugog objekta

Na slici 3.5 prikazan je slučaj kada se vrh jednog objekta sudario sa stranicom drugog. Ovaj slučaj je detektiran kada se intervali jednog trokuta na pravcu L u potpunosti nalaze unutar intervala drugog trokuta. Kod sudara između vrha i trokuta točke sudara lako se određuju. Točka sudara na prvom tijelu je vrh koji se nalazi unutar drugog tijela, a na drugom tijelu to je točka na trokutu najbliža tom vrhu. Radi pojednostavljenja proračuna u simulaciji se za točku na trokutu uzimaju koordinate točke vrha koji probada tijelo. Normala sudara je normala tog trokuta.



Slika 3.6 Sudar bridova objekata

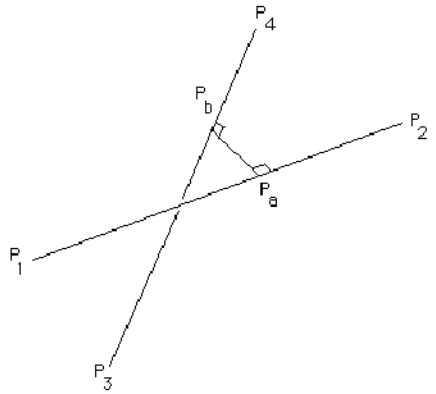
Kod sudara dva brida točke sudara nešto je kompliciranije odrediti. Potrebno je na 2 brida koja su se sudarila pronaći najbliže točke. To radimo pomoću sljedećeg algoritma:

Prvi brid definiran je sa dvije točke P1 i P2, a drugi sa P3 i P4, kao na slici 3.7. Točka na bridu a ima sljedeću jednadžbu:

$P_a = P_1 + mu_a (P_2 - P_1)$

Slično tome točka na bridu b ima jednadžbu:

$P_b = P_3 + mu_b (P_4 - P_3)$



Slika 3.7 Najbliže točke na bridovima

Vrijednosti mu_a i mu_b kreću se od $-\infty$ do ∞, a za segmente pravaca između točaka P_1 P_2 i P_3 P_4 imaju iznose između 0 i 1.

Pravac koji spaja dvije najbliže točke na linijama bit će okomit na oba pravca:

$(P_a - P_b)\cdot(P_2 - P_1) = 0$

$(P_a - P_b)\cdot(P_4 - P_3) = 0$

To proširujemo jednadžbama:

$(P_1 - P_3 + mu_a (P_2 - P_1) - mu_b (P_4 - P_3))\cdot(P_2 - P_1) = 0$

$(P_1 - P_3 + mu_a (P_2 - P_1) - mu_b (P_4 - P_3))\cdot(P_4 - P_3) = 0$

Računanjem skalarnog produkta dobivamo:

$d_{1321} + mu_a d_{2121} - mu_b d_{4321} = 0$

$d_{1343} + mu_a d_{4321} - mu_b d_{4343} = 0$

Gdje je:

$d_{mnop} = (P_m - P_n)\cdot(P_o - P_p)$

S tim da vrijedi: $d_{mnop} = d_{opmn}$

Rješavanjem se dobiva da je:

$mu_a = (d_{1343} d_{4321} - d_{1321} d_{4343}) / (d_{2121} d_{4343} - d_{4321} d_{4321})$

$mu_b = (d_{1343} + mu_a d_{4321}) / d_{4343}$

Točke sudara dobivamo iz početnih jednadžbi, a za normalu sudara uzima se vektorski produkt vektora smjera dva sudarena brida. Orjentacija normale uvijek je od objekta B prema objektu A.

Sudar je detektiran samo ako se dobivene točke sudara približavaju jedna drugoj. Ako je njihova relativna brzina manja od nule, znači da su se već odbile jedna od druge, te na njih ne primjenjujemo impuls sudara.

4. Reakcija na sudar

4.1 Prikaz tijela

Kod simulacije gibanja krutog tijela koristi se sličan model kao i kod simulacije gibanja čestica. Za simulaciju gibanja čestice potrebna su dva parametra, a to su vektor pozicije i vektor brzine. Na promjenu pozicije ne djeluje se direktno nego preko promijene brzine pomoću postupka integracije. Položaj krutog tijela u prostoru određen je pozicijom centra mase te orjentacijom. Orjentacija se u 3D prostoru određuje pomoću kvaterniona ili pomoću 3D matrice rotacije. Kod simulacije krutih tijela kvaternioni su bolji način za prikaz orijentacije zbog toga što su manje podložni greškama zaokruživanja koje se akumuliraju kod rotacije od matrica.

4.2 Određivanje reakcije na sudar

Jednom kada je sudar detektiran, potrebno je odrediti reakciju na sudar. Očita stvar koju trebamo napraviti jednom kad je sudar detektiran bila bi primjeniti sile na oba objekta, ali to nebi bilo dobro riješenje. Sila neće spriječiti objekte da prođu jedan u drugi jer sila ne može trenutno promijeniti brzinu. Sila mijenja brzinu integracijom kroz vrijeme, a pošto se u trenutku kad je detekiran sudar tijela već dodiruju nemamo dodatno vrijeme u kojem sila može promijeniti brzinu. Potrebno je trenutno promijeniti brzinu, a to se postiže uporabom impulsa. U stvarnom sudaru sudaru događaju se komplicirane stvari koje ne možemo direktno simulirati. Zato se proces sudara aproksimira idealiziranim modelom. Impuls možemo promatrati kao jako veliku silu koja dijelu jako kratak vremenski period. Impuls mijenja moment trenutno što rezultira trenutnom promjenom brzine, te sprječava objekte da prodru jedan u drugi. Središnji problem reakcije na sudar je računanje tog impulsa.

4.3 Određivanje impulsa sudara

Postoji mnogo načina na koji se računa vrijednost i smjer impulsa. Mi u našoj simulaciji pretpostavljamo da u sudaru ne postoji trenje, pa je jedina sila tijekom sudara ona u smjeru vektora normale **n**. Trenje bi uzrokovalo i silu okomitu na normalu.

Početne brzine prije sudara točaka sudara određuju se iz sljedećih izraza:

$$\mathbf{v}_{ap1} = \mathbf{v}_{a1} + \omega_{a1} \times \mathbf{r}_{ap} \quad (4.1)$$

$$\mathbf{v}_{bp1} = \mathbf{v}_{b1} + \omega_{b1} \times \mathbf{r}_{bp} \quad (4.2)$$

gdje su: $\mathbf{v}_{a1}$ i $\mathbf{v}_{b1}$ brzine centra mase 2 tijela prije sudara, ω_{a1} i ω_{b1} kutne brzine 2 tijela prije sudara, a $\mathbf{r}_{ap}$ i $\mathbf{r}_{ap}$ vektori od centra mase tijela do točke sudara.

Na sličan način određuju se brzine točaka nakon sudara:

$$\mathbf{v}_{ap2} = \mathbf{v}_{a2} + \omega_{a2} \times \mathbf{r}_{ap} \quad (4.3)$$

$$\mathbf{v}_{bp2} = \mathbf{v}_{b2} + \omega_{b2} \times \mathbf{r}_{bp} \quad (4.4)$$

Sada pronalazimo relativne brzine točaka koje se sudaraju.

$\mathbf{v}_{ab1} = \mathbf{v}_{ap1} - \mathbf{v}_{bp1}$
$\mathbf{v}_{ab2} = \mathbf{v}_{ap2} - \mathbf{v}_{bp2}$

Gdje je $\mathbf{v}_{ab1}$ početna, a $\mathbf{v}_{ab2}$ konačna relativna brzina točaka sudara.

Koristeći izraze 4.1 i 4.2, odnosno 4.3 i 4.4 za brzine točke na tijelu ove formule možemo proširiti na sljedeće:

$\mathbf{v}_{ab1} = \mathbf{v}_{a1} + \omega_{a1} \times \mathbf{r}_{ap} - \mathbf{v}_{b1} - \omega_{b1} \times \mathbf{r}_{bp}$ (4.5)
$\mathbf{v}_{ab2} = \mathbf{v}_{a2} + \omega_{a2} \times \mathbf{r}_{ap} - \mathbf{v}_{b2} - \omega_{b2} \times \mathbf{r}_{bp}$ (4.6)

Ako uzmemo da je $\mathbf{n}$ normala sudara, a e koeficijent elastičnosti sudara gornja dva izraza možemo povezati sljedećom formulom:

$\mathbf{v}_{ab2} \cdot \mathbf{n} = -e\, \mathbf{v}_{ab1} \cdot \mathbf{n}$ (4.7)

Impuls sudara je $j\,\mathbf{n}$, gdje je j parametar koji moramo odrediti. Objekt A osjeća impuls $j\,\mathbf{n}$, dok na objekt B djeluje po iznosu isti, ali smjerom suprotni impuls $-j\,\mathbf{n}$. Impuls je promjena inercije, te ako ga podijelimo sa masom dobivamo promjenu brzine:

$\mathbf{v}_{a2} = \mathbf{v}_{a1} + j\,\mathbf{n} / m_a$ (4.8)
$\mathbf{v}_{b2} = \mathbf{v}_{b1} - j\,\mathbf{n} / m_b$ (4.9)

Promjena zakretnog momenta objekta A uzrokovana impulsom $j\,\mathbf{n}$ dana je izrazom $\mathbf{r}_{ap} \times j\,\mathbf{n}$. Promjena u zakretnom momentu dijeli se sa momentom inercije kako bi dobili promjenu u kutnoj brzini. Kutne brzine nakon sudara dobivaju se iz izraza:

$\omega_{a2} = \omega_{a1} + [I_a]^{-1}(\mathbf{r}_{ap} \times j\,\mathbf{n})$ (4.10)
$\omega_{b2} = \omega_{b1} - [I_b]^{-1}(\mathbf{r}_{bp} \times j\,\mathbf{n})$ (4.11)

Koristeći gornje jednadžbe možemo riješiti impulsni parametar j . Krećemo od izraza 4.7 te ga proširujemo izrazom 4.6:

$\mathbf{v}_{ab2} \cdot \mathbf{n} = -e\, \mathbf{v}_{ab1} \cdot \mathbf{n}$
$(\mathbf{v}_{ap2} - \mathbf{v}_{bp2}) \cdot \mathbf{n} = -e\, \mathbf{v}_{ab1} \cdot \mathbf{n}$
$(\mathbf{v}_{a2} + \omega_{a2} \times \mathbf{r}_{ap} - \mathbf{v}_{b2} - \omega_{b2} \times \mathbf{r}_{bp}) \cdot \mathbf{n} = -e\, \mathbf{v}_{ab1} \cdot \mathbf{n}$

Zatim tu jednadžbu proširujemo jednadžbama 4.8 – 4.11 te dobivamo:

$((\mathbf{v}_{a1} + j\,\mathbf{n} / m_a) + (\omega_{a1} + [I_a]^{-1}(\mathbf{r}_{ap} \times j\,\mathbf{n})) \times \mathbf{r}_{ap} - (\mathbf{v}_{b1} - j\,\mathbf{n} / m_b) - (\omega_{b1} - [I_b]^{-1}(\mathbf{r}_{bp} \times j\,\mathbf{n})) \times \mathbf{r}_{bp}) \cdot \mathbf{n} = -e\, \mathbf{v}_{ab1} \cdot \mathbf{n}$

Pošto lijeva strana sadrži izraze za $\mathbf{v}_{ab1} \cdot \mathbf{n}$ dane izrazom 4.5 prebacujemo ih na desnu stranu te dobivamo:

$$(j\,\mathbf{n} / m_a + [I_a]^{-1}(\mathbf{r}_{ap} \times j\,\mathbf{n}) \times \mathbf{r}_{ap} + j\,\mathbf{n} / m_b + [I_a]^{-1}(\mathbf{r}_{bp} \times j\,\mathbf{n}) \times \mathbf{r}_{bp}) \cdot \mathbf{n} = -(1 + e)\,\mathbf{v}_{ab1} \cdot \mathbf{n}$$

$\mathbf{n}$ je normaliziran pa vrijedi da je $\mathbf{n} \cdot \mathbf{n} = 1$. Daljnjim sređivanjem dobivamo:

$$j\,(1 / m_a + 1 / m_b + (\mathbf{r}_{ap} \times \mathbf{n}) \cdot ([I_a]^{-1}(\mathbf{r}_{ap} \times \mathbf{n})) + (\mathbf{r}_{bp} \times \mathbf{n}) \cdot ([I_b]^{-1}(\mathbf{r}_{ap} \times \mathbf{n})) = -(1 + e)\,\mathbf{v}_{ab1} \cdot \mathbf{n}$$

Dijeljenjem dobivamo konačan izraz za j:

$$j = \frac{-(1 + e)\,\mathbf{v}_{ab1} \cdot \mathbf{n}}{1 / m_a + 1 / m_b + (\mathbf{r}_{ap} \times \mathbf{n}) \cdot ([I_a]^{-1}(\mathbf{r}_{ap} \times \mathbf{n})) + (\mathbf{r}_{bp} \times \mathbf{n}) \cdot ([I_b]^{-1}(\mathbf{r}_{bp} \times \mathbf{n}))}$$

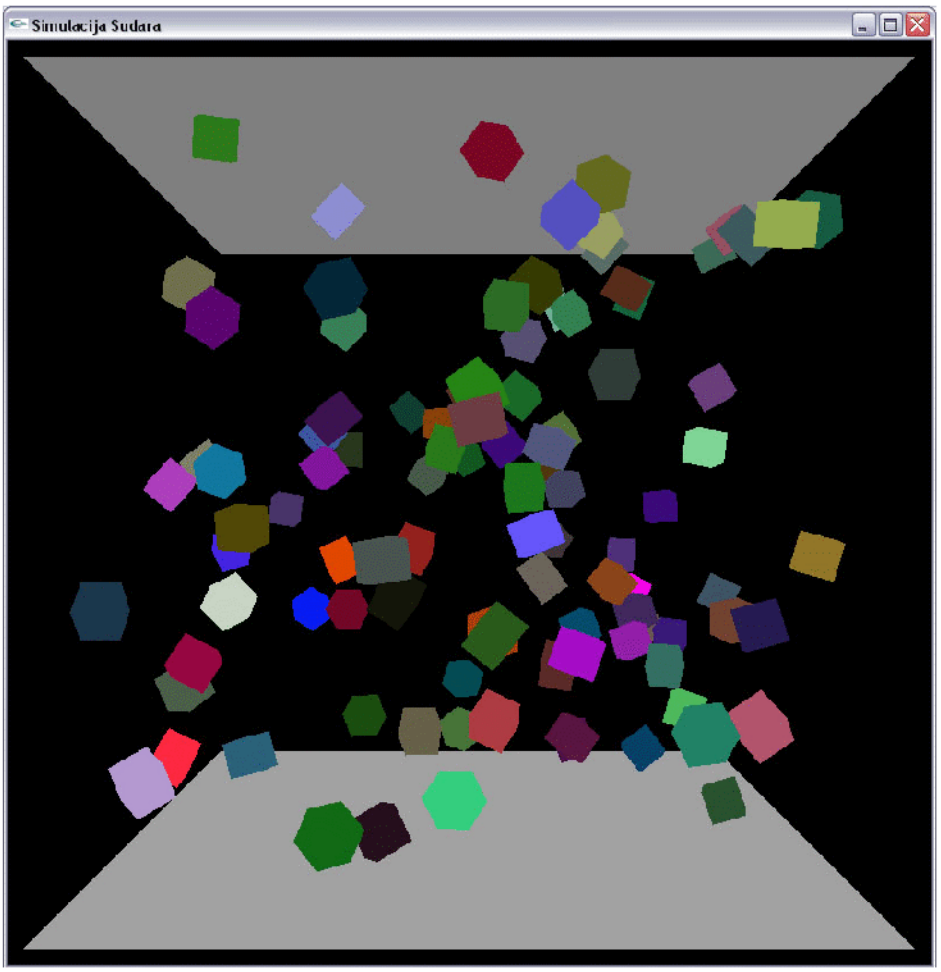
Istu jednadžbu možemo koristiti i za sudar između objekta i zida koristeći pretpostavku da je masa zida beskonačna, pa imamo $m_b \to \infty$ i $[I_b]^{-1} \to 0$ te jednadžba poprima sljedeći oblik:

$$j = \frac{-(1 + e)\,\mathbf{v}_{ap1} \cdot \mathbf{n}}{1 / m_a + (\mathbf{r}_{ap} \times \mathbf{n}) \cdot ([I_a]^{-1}(\mathbf{r}_{ap} \times \mathbf{n}))}$$

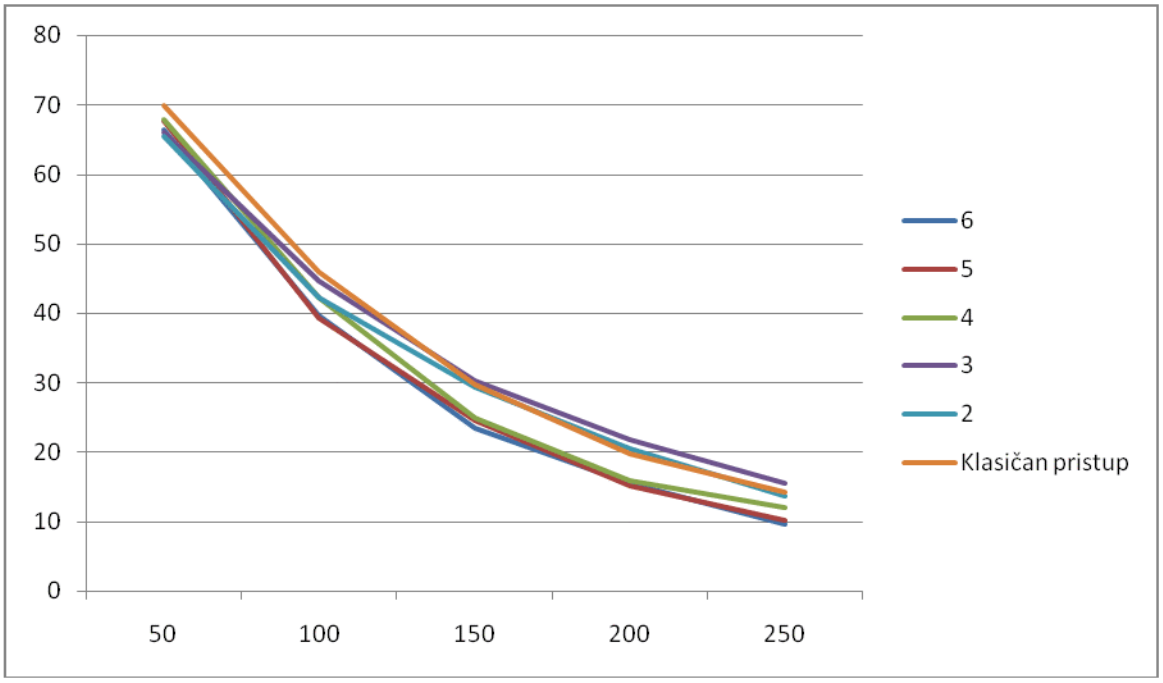
5. Rezultati simulacije

Tijekom simulacije ispitano je kako različiti parametri utječu na brzinu iscrtavanja kod različitog broja objekata u sceni. Sva ispitivanja obavljena su za slučajan raspored objekata unutar kocke duljine stranice 20.0.

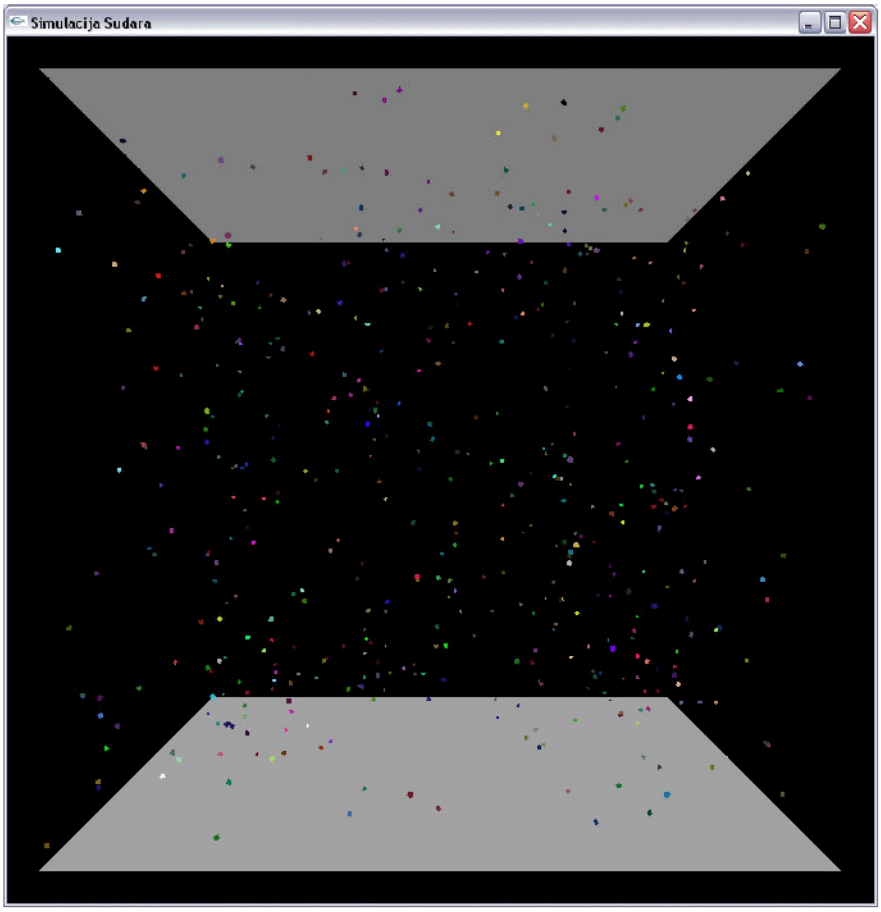
5.1 Utjecaj različite maksimalne dubine oktalnog stabla



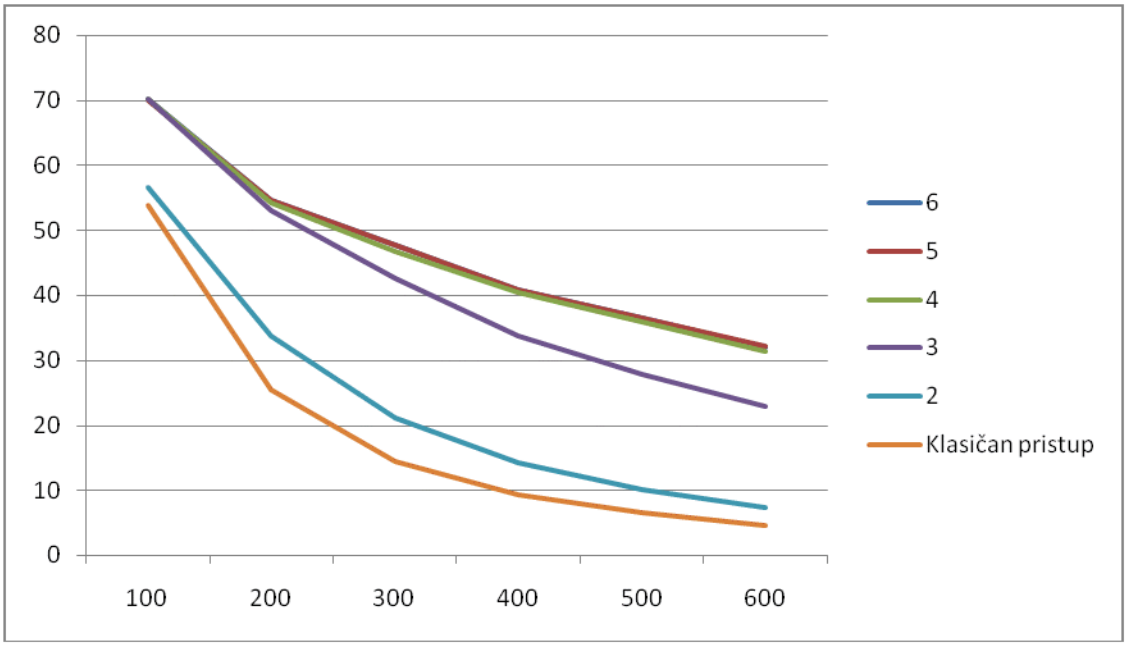
Slika 5.1 Izgled scene sa 100 slučajno raspoređenih kocaka duljine stranice 1.0



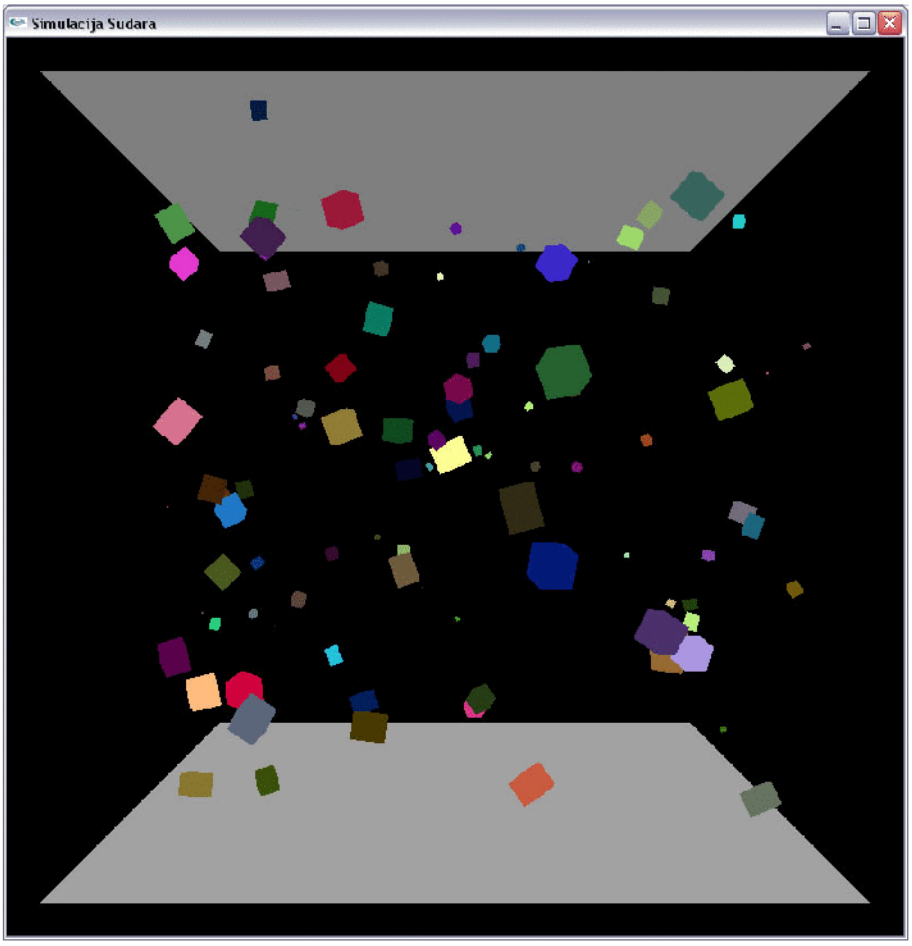
Slika 5.2 Graf zavisnosti brzine iscrtavanja i broja objekata kod različitih maksimalnih dubina oktalnog stabla za kocke duljine stranice 1.0



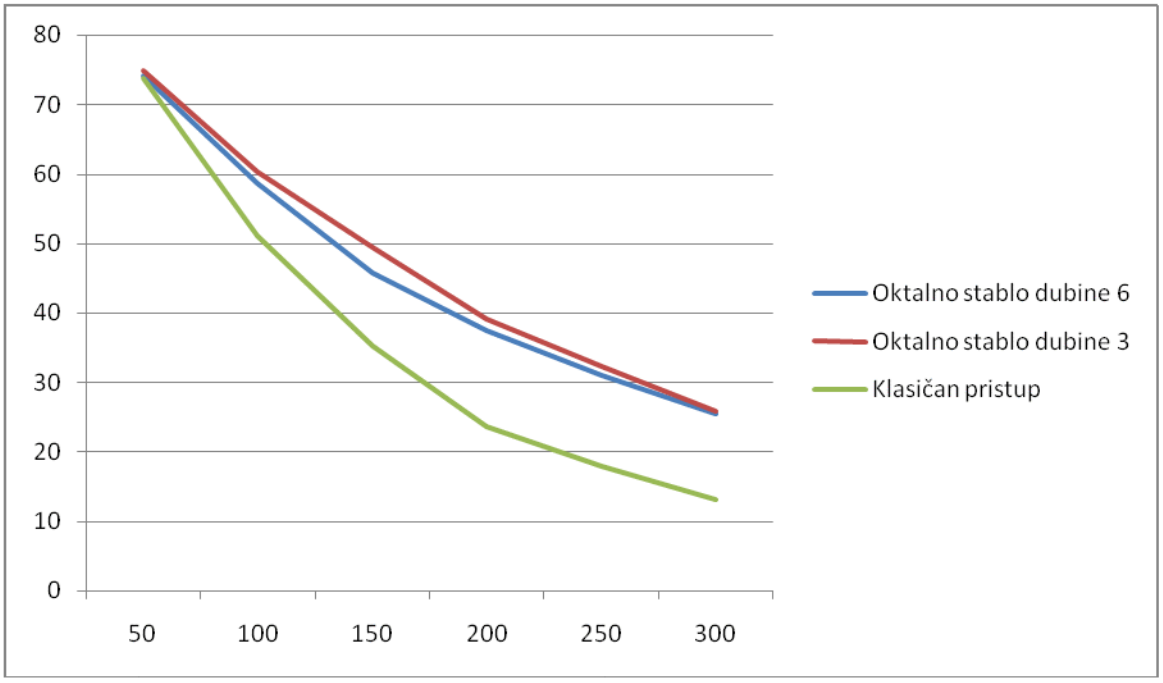
Slika 5.3 Izgled scene sa 500 slučajno raspoređenih kocaka duljine stranice 0.1



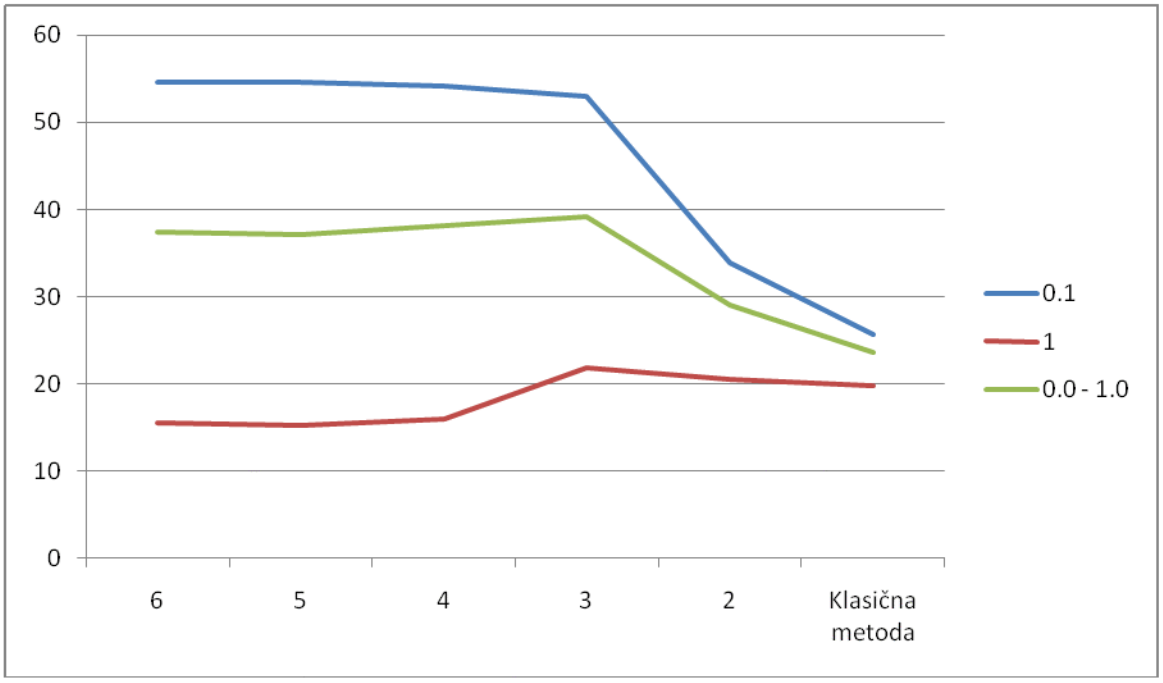
Slika 5.4 Graf zavisnosti brzine iscrtavanja i broja objekata kod različitih maksimalnih dubina oktalnog stabla za kocke duljine stranice 0.1



Slika 5.5 Izgled scene sa 100 slučajno raspoređenih kocaka nasumičnih duljina stranica između 0.0 i 1.0



Slika 5.6 Graf zavisnosti brzine iscrtavanja i broja objekata kod različitih maksimalnih dubina oktalnog stabla za kocke nasumičnih duljina stranica između 0.0 i 1.0

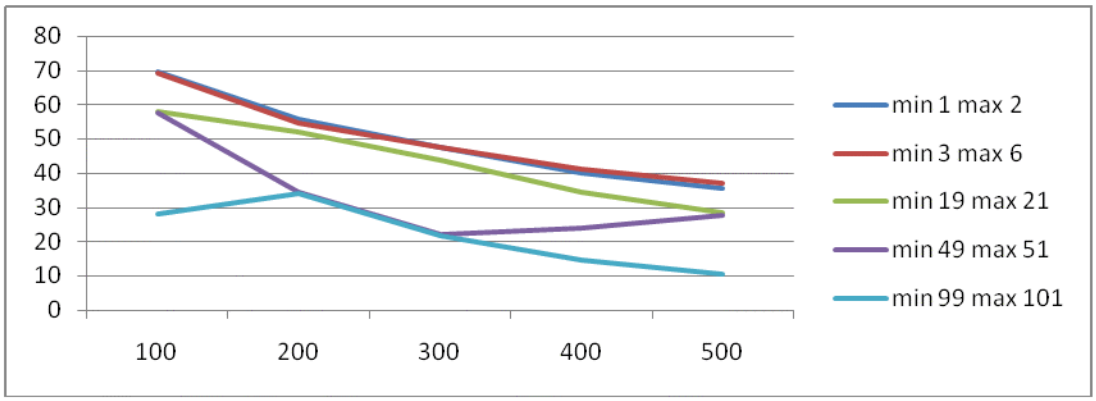


Slika 5.7 Graf zavisnosti brzine iscrtavanja i maksimalne dubine oktalnog stabla kod različitih duljina stranica objekata za 200 objekata na sceni

Dok su objekti na sceni relativno mali u odnosu na veličinu scene oktalno stablo pokazuje se kao dobar izbor. Međutim za veće objekte stablo postaje kontraproduktivno, jer se objekti pojavljuju u mnogo oktana te korištenjem stabla zapravo povećavamo broj potrebnih

testova. Za objekte veličine 1.0 oktalno stablo manje dubine pokazalo je nešto bolje rezultate, ali ti rezultati nisu se pokazali kao znatno ubrzanje u odnosu na klasičnu metodu.

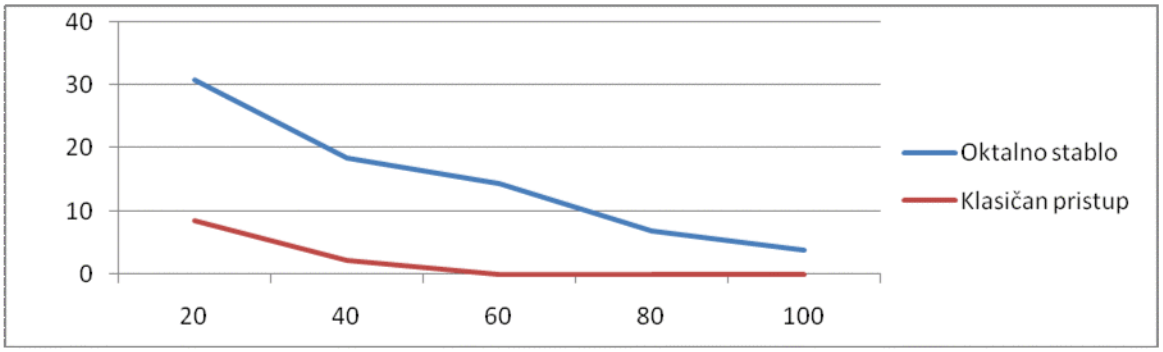
5.2 Utjecaj različitih granica broja objekata po oktanu



Slika 5.8 Graf zavisnosti brzine iscrtavanja i broja objekata za kocke duljine stranice 0.1 uz različite granice broja objekata po oktanu

Najbolje rezultate pokazala je granica od 3 do 6 objekata po oktanu. Manje granice nisu više pokazivale poboljšanje u performansama. Neočekivani rezultati pojavili su se kod većih granica. Kod 49 do 51 objekata po oktanu program je pokazao bolje performanse sa 400 i 500 objekata u sceni nego sa 300, dok je kod granice od 99 do 101 objekata po oktanu program pokazao bolje performanse za 200 nego za 100 objekata u sceni. Razlog tome je što je kod većeg broja objekata došlo do dodatne podjele oktana, te se smanjio broj potrebnih testova. Korištenje većih granica nije se pokazalo kao dobro riješenje zbog povećanja broja testova unutar oktana.

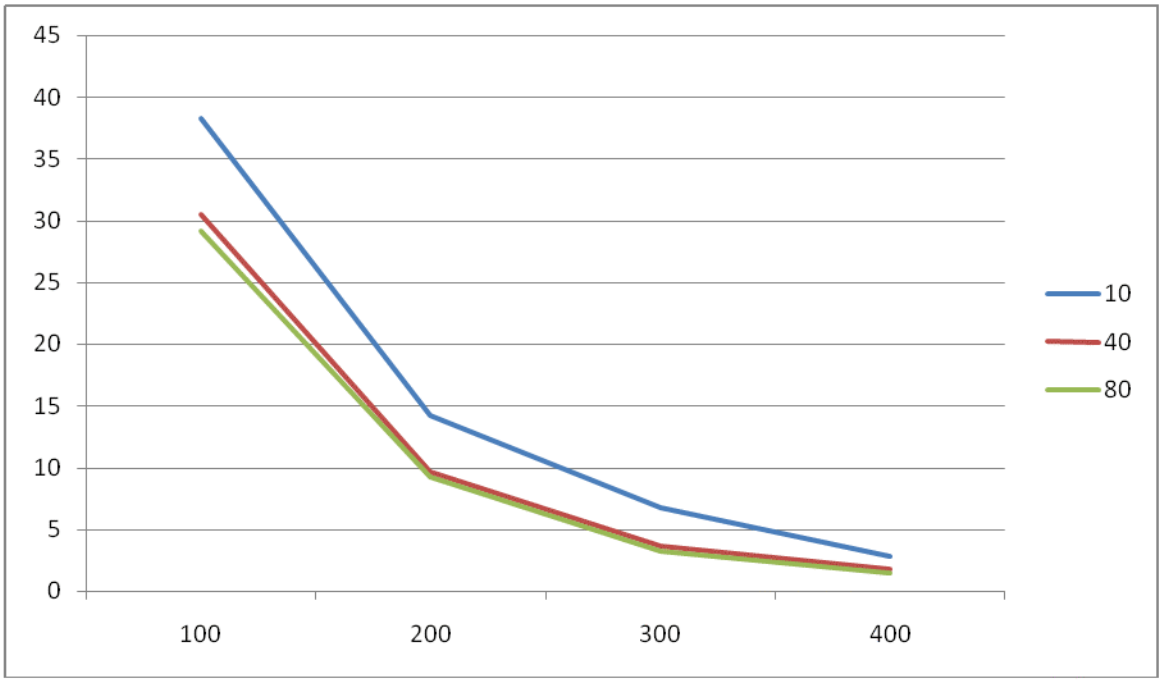
5.3 Utjecaj korištenja obujmica



Slika 5.9 Graf zavisnosti brzine iscrtavanja i broja objekata za kocke duljine stranice 1.0 bez korištenja obujmica sa i bez korištenja oktalnog stabla

Korištenjem obujmica očekivano su dobivena velika poboljšanja u performansama. Već kod relativno malog broja objekata u sceni bez obujmica rezultati su bili jako loši. Razlog tome je što su detaljni testovi u uskoj fazi detekcije vremenski vrlo zahtjevni.

5.4 Utjecaj različite brzine objekata

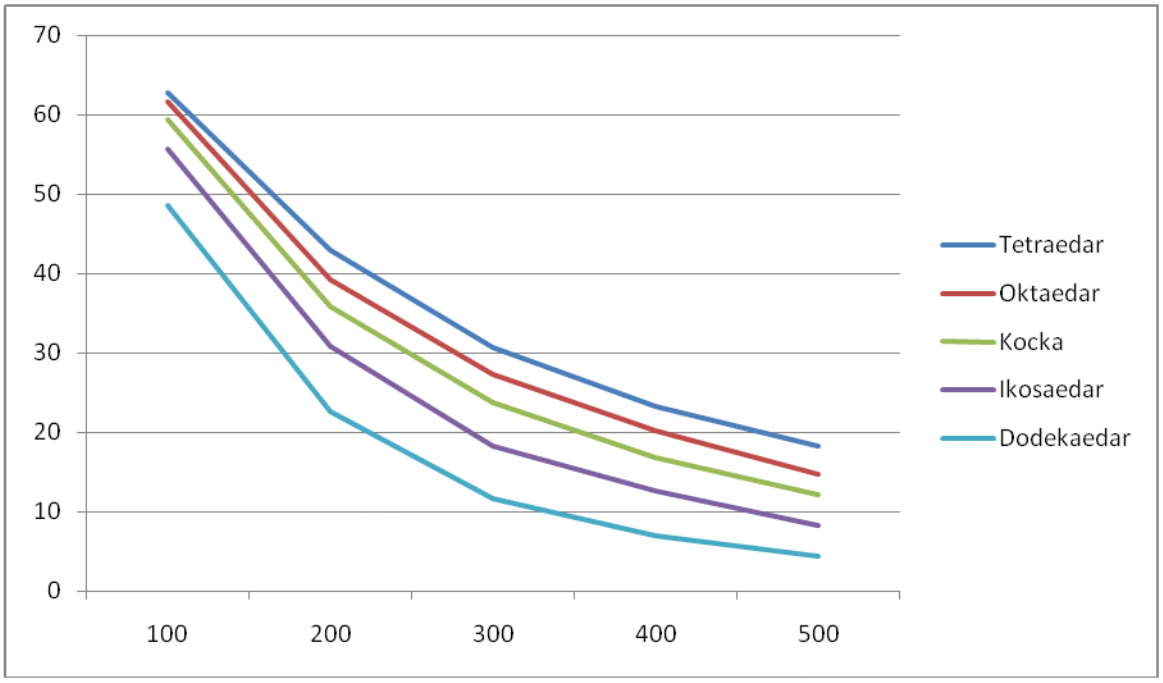


Slika 5.10 Graf zavisnosti brzine iscrtavanja i broja objekata za kocke duljine stranice 1.0 različitih početnih brzina

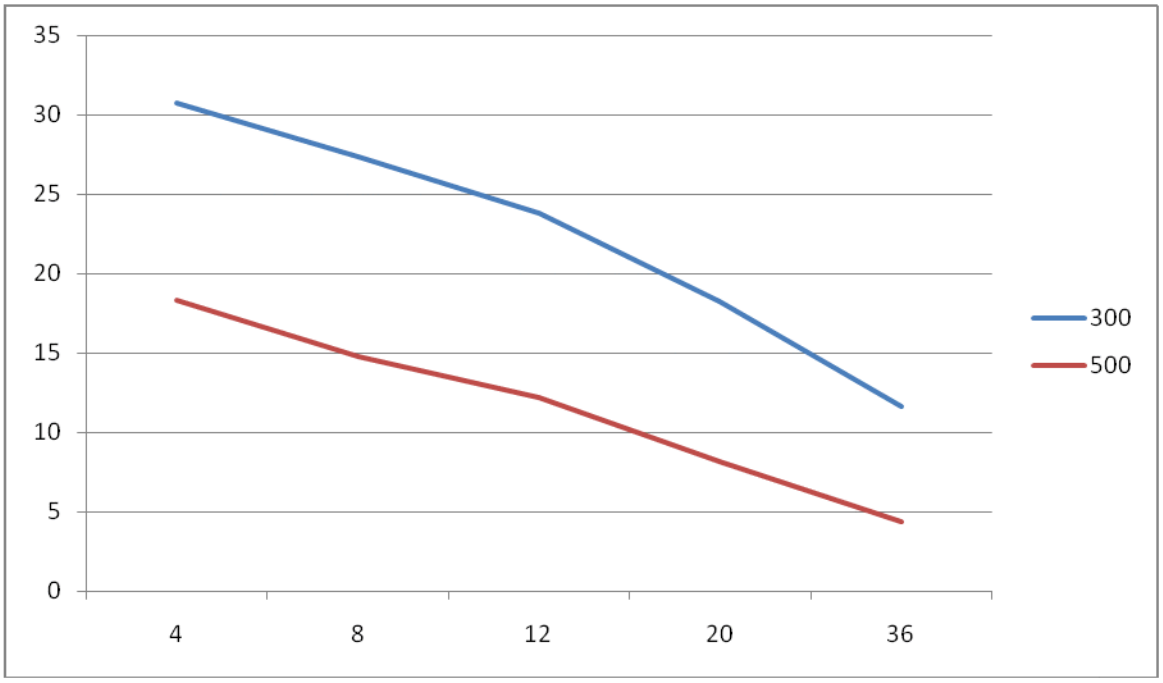
Povećanje brzine objekata dovelo je do malog pada u brzini iscrtavanja. Razlog toga može biti veći broj sudara u jedinici vremena, međutim kod većih brzina rezultati se znatno ne razlikuju.

5.5 Utjecaj različitog broja vrhova i trokuta po objektu

Slika 5.11 Različiti objekti



Slika 5.12 Graf zavisnosti brzine iscrtavanja i broja objekata za različite objekte promjera obujmice 1.0



Slika 5.13 Graf zavisnosti brzine iscrtavanja i broja trokuta od kojih su sastavljeni objekti za 300 i 500 objekata na sceni

Iako se testovi za pojedine trokute obavljaju tek u uskoj fazi detekcije povećanje broja trokuta po objektu znatno smanjuje brzinu iscrtavanja.

6. Zaključak

Rezultati simulacija većinom su potvrdili pretpostavke. Obujmice su se pokazale kao odlično riješenje bez kojih bilo koja simulacija u realnom vremenu ne bi bila moguća. Zbog toga što su detaljni testovi presjeka vremenski zahtjevni korištenje obujmica čak i kod malog broja objekata u sceni rezultiralo je znatnim ubrzanjem izvođenja. Oktalno stablo za objekte koji su dosta maleni u odnosu na veličinu scene pokazalo je odlične rezultate u eliminiranju potencijalnih sudara. Međutim kod nešto većih objekata oktalno stablo pokazalo je dosta loše rezultate koji su samo nešto bolji od rezultata klasične metode detekcije i to tek kod vrlo velikog broja objekata u sceni. Uska faza detekcije presjeka trokuta pokazala se relativno sporom, te bi ju trebalo dodatno optimizirati.

7. Literatura

[1] Chris Hecker, Rigid Body Dynamics, 6. 1997.,

http://chrishecker.com/Rigid_Body_Dynamics , 13.1.2010.

[2] Tomas Möller, A Fast Triangle-Triangle Intersection Test, 1997.,

http://www.cs.lth.se/home/Tomas_Akenine_Moller/pubs/tritri.pdf , 13.1.2010.

[3] Paul Bourke, The Shortest Line Between Two Lines In 3D, 4. 1998.,

http://local.wasp.uwa.edu.au/~pbourke/geometry/lineline3d/ , 13.1.2010.

[4] Erik Neumann,Rigid Body Collisions, 2004.,

http://www.myphysicslab.com/collision.html , 13.1.2010.

[5] Bill Jacobs, Collision Detection OpenGL tutorial,

http://www.videotutorialsrock.com/opengl_tutorial/collision_detection/text.php , 13.1.2010.

8. Sažetak / Abstract

SIMULACIJA SUDARA KRUTIH TIJELA

U ovom radu opisane su tehnike pogodne za izradu simulacije sudara krutih tijela. U prvom dijelu opisana je detekcija sudara u dvije faze. U sklopu široke faze opisano je oktalno stablo kao struktura podataka za ubrzavanje detekcije, te obujmice koje se koriste za aproksimaciju objekata radi ubrzavanja testova. U sklopu uske faze opisani su testovi za brzo određivanje presjeka dva trokuta, te način na koji se određuju točke sudara i normala. U drugom dijelu rada opisano je na koji se način iz dobivenih točaka sudara i normale određuje reakcija na sudar. Opisane strukture podataka i algoritmi implementirani su u programskom jeziku C++ korištenjem OpenGL grafičkog sučelja. Na trodimenzionalnim primjerima ocijenjen je utjecaj različitih parametara.

Ključne riječi: simulacija sudara, sustavi za simulaciju fizike, detekcija sudara, oktalno stablo, obujmice, dinamika krutih tijela

SIMULATION OF COLLISIONS BETWEEN RIGID BODIES

This paper describes the development of techniques suitable for the simulation of rigid body collisions. The first part described the collision detection in two phases. In the broad phase an octree is described as data structure to speed up detection, and bounding volumes used to approximate objects to speed up tests. The narrow phase described the tests to quickly determine the cross section of two triangles, and the way in which determine the collision point and normal. The second part of the work describes in which way are collision points and normal used to determine collision reaction. Described data structures and algorithms are implemented in C + + using the OpenGL graphics interface.
Effects of different parameters were compared using various 3D examples.

Keywords: collision simulation, physics engine, collision detection, octree, bounding volumes, rigid body dynamics