Електротехнички факултет Универзитета у Београду
Катедра за рачунарску технику и информатику

дипломски рад

Развој графичког импулсног симулатора динамике крутих тела

ментор

др Игор Тартаља

студент

Марко Тинтор
tintor@gmail.com

Београд
октобар 2007.

Садржај

Увод

Симулација динамике је једна од битнијих тема у рачунарској графици. Користи се у разноврсним областима за моделирање, забаву и обуку. Динамички објекти су свуда присутни, и могућност нумеричке симулације њиховог понашања је важна у индустријама као што су филмови, рачунарске игре, аутомобилска и машинска индустрија. Уочљива је разлика између тела која се могу знатно деформисати и оних код којих су деформације занемарљиве или небитне. У другом случају потреба за ефикасношћу доводи до ограничења на крута тела.

У овом раду ће бити приказан једноставан симулатор динамике крутих тела у тродимензионалном простору. Интерфејс је графички, а један од циљева је и могућност симулирања у реалном времену, што омогућава природну интеркцију са корисником за време трајања симулације.

Интеракције између тела биће симулиране импулсном методом, чија је предност једноставност у односу на остале методе. Импулсном методом се рачунају импулси који спречавају тела у додиру од међусобног пробијања. Одређивање тих импулса је једноставно и брзо. Импулсна метода је веома ефикасна, али и мање прецизна од других медота пошто се импулси одређују само за пар тела, уместо за сва тела одједном.

Симулираће се судари, статички контакти, спојеви, гомилање и линеарно трење (статичко и динамичко) конвексних тела на нивоу класичне механике. Угаоно трење (котрљање и обртање) се неће симулирати. Две врсте ограничења слободном кретању тела које ће бити имплементиране су забрана међусобног пробијања и разне врсте спојева (везе између тела које смањују број степени слободе).

Конвексна тела су изабрана ради упрошћавања и убрзавања подсистема за откривање контаката. Пар конвексних тела може имати највише један заједнички контакт који може бити тачка, дуж или полигон. Комплексна тела могу да се направе комбиновањем спојева и једноставнијих тела.

Захваљујем се свом ментору др Игору Тартаљи и мр Ђорђу Ђурђевићу на помоћи и сугестијама у изради овог рада.

октобар 2007.
Марко Тинтор

Преглед проблематике

Шта је све потребно да би се симулирала динамика крутих тела? Потребно је да за свако тело буде позната његова позиција, брзина и укупна сила која делује на њега. На основу ових вредности, и под претпоставком да се сила неће знатно променити, интеграцијом се може израчунати нова позиција и брзина тела после неког кратког интервала Δt. Укупна сила која делује на тело се састоји од екстерне и реакционе силе која потиче од разних ограничења (контаката и спојева).

Реакционе силе морају да одржавају сва ограничена која постоје у систему. Рачунање тих сила представља највећи проблем у симулирању динамике крутих тела, па постоји више различитих метода како се тај проблем може решити [1]:

- Казнене методе (енг. Penalty-based)
  Дозвољава се пробијања ограничења, а реакционе силе се једноставно рачунају као јаке еластичне силе које делују на тело тако да се пробијено ограничење смањи. Проблем је што такве силе повећавају нестабилност интеграције и што су пробијања пропорционална екстерим силама.

- Метода ограничења (енг. Constraint-based)
  Своди се на решавање великог система неједначина, па је компликована за имплементацију, али не дозвољава пробијања.

- Импулсна метода (енг. Impulse-based)
  Ова метода симулира све интеракције између тела као размене импулса. Једноставна је за имплементацију, а ефикасни итеративни начин рачунања импулса је погодан за симулирање у реалном времену.

- Синхронизација колизија (енг. Collision synchronization)
  Такође позната као и анимација заснована на оптимизацији проширује физички модел заснован само на позицијама и решава проблем малих временских корака на коме се базирају претходне три методе.

- Комбинација више метода где да слабост једне од поменутих метода допуњује друга
  На пример, интерне интеракције у комплексним телима са спојевима се могу симулирати методом ограничења, а екстерне интеракције са другим телима импулсном методом.

Други већи проблем је откривање парова тела која су у контакту или се пробијају. Подржавање конкавних тела може да усложи програм пошто су у општем случају пресеци између конкавних тела сложенији и теже се проналазе него код конексних. Проблеме могу да праве нумеричке непрецизности које проузрокују да темена тела која припадају истој страници више не буду у истој равни после ротирања, транслирања или скалирања. Од начина описивања контакта зависиће и реакције објеката при контактима. Имплементација алгоритма за откривање колизија мора бити пажљиво имплементирана како би правилно радила за разноразне међусобне положаје тела.

Предложено решење

Опис стања крутог тела

Стање крутог тела се састоји из позиције центра масе x, оријентације Ω, брзине центра масе υ и угаоне брзине ω [3]. Све ове величине осим оријентације су 3-елементни вектори.

Оријентација тела Ω се представља помоћу јединичног 4-елементног вектора кватернице (енг. quaternion) који описује ротацију [4] око јединичног 3-елементног вектора d за угао α: $\Omega = (d \sin\frac{\alpha}{2}, \cos\frac{\alpha}{2})$

Овакав начин представљања оријентације има више предности над ротационом матрицом. С обзиром на то да се код кватернице користе само 4 броја, у односу на 9 код матрице, нормализација је бржа и једноставнија.

Из кватернице се једноставно може израчунати и ротациона матрица [4] [5].

Интеграција

За интеграцију стања тела кроз време је изабран Ојлеров симплексички интегратор [6]. Његове предности су једноставност, природно уклапање у главну петљу симулатора и највећа стабилност у односу на интеграторе истог реда [7].

$$\dot{x}(t+\Delta t) = \dot{x}(t) + \ddot{x}(t)\,\Delta t$$
$$x(t+\Delta t) = x(t) + \dot{x}(t+\Delta t)\,\Delta t$$

Акумулирана грешка овог интегратора је реда величине Δt [6].

Главна петља симулације

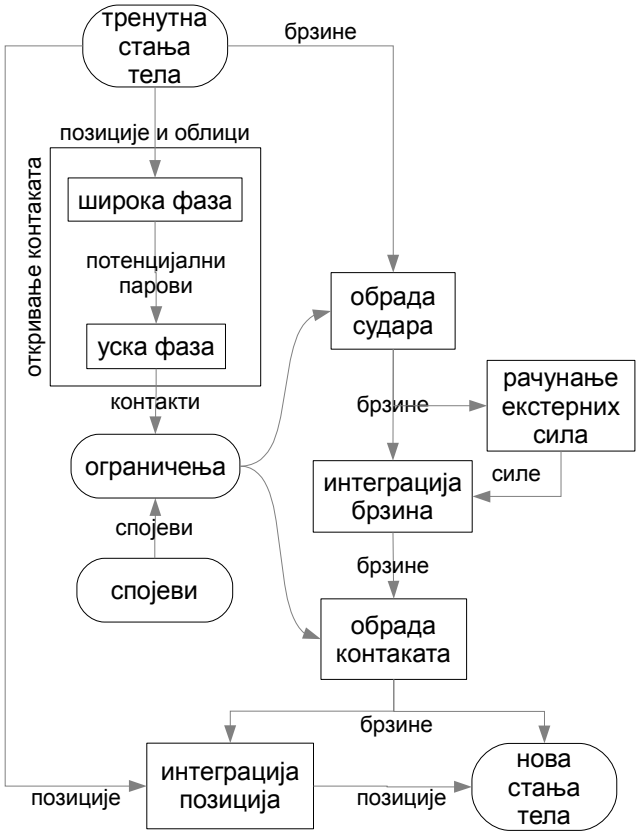
Главна петља симулира систем у корацима у трајању од Δt = k / f, где је k релативна брзина протока времена у симулацији, а f фреквенција освежавања (обично од 20 до 60Hz). Ако би се, на пример, симулирало кретање небеских тела k би било $>10^5$. Ако се жели постићи већа тачност корак од Δt се може поделити на n корака у трајању од Δt / n. Овиме се брзина израчунавања повећава n пута.

У свакој итерацији потребно је на основу стања тела на почетку корака одредити стања тела на крају корака (слика 1). Тело петље се састоји из следећих корака[8]:

1. откривање контаката (енг. collision detection)
2. обрада судара (размена импулса)
3. рачунање екстерних сила
4. интеграција брзина за Δt
5. обрада контаката (размена импулса)

6. интеграција позиција и оријентација за Δt

У кораку 1 се на основу тренутних позиција тела проналазе контакти који постоје између њих. У корацима 2 и 5 се итеративно рачунају импулси које тела размењују како би се спречила пробијања ограничења. У кораку 3 се акумулирају екстерне силе које делују на тела. Брзинама се додају деловања екстерних сила у трајању од Δt у кораку 4, а нове позиције и оријентације се рачунају у кораку 6.



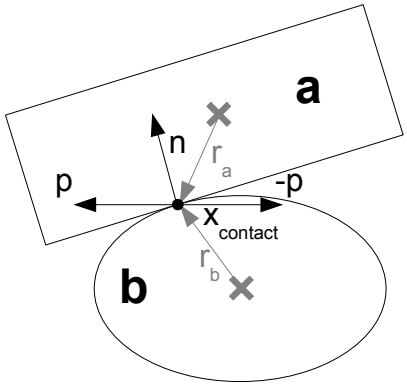
Слика 1: Дијаграм тока алгоритма симулације

Екстерне силе

Екстерним силама се у овом раду сматрају све силе које не потичу од контаката и спојева. Примери су површинска гравитациона сила $F = mg$ и сила отпора средине $F = -Av – Bv^2$. Екстерне силе могу деловати и у тачки која се не поклапа са центром масе тела. Ове силе могу бити произвољне, али једина претпоставка је да су константне за време Δt.

Судари са трењем

Посматра се судар између два тела (слика 2). Нека је $\mathbf{x}_{\mathbf{contact}}$ тачка контакта, а **n** нормала у тачки контакта од тела **b** ка телу **a**. На основу закона акције и реакције тело **a** прима импулс **p**, а тело **b** прима супротни импулс **-p** у тачки контакта.



Слика 2: Судар два тела

Зависност импулса **p** од промене релативне брзине **Δu** се изводи на следећи начин:

$$\Delta v_{\text{cm\_a}} = \frac{p}{m_a}$$

$$\Delta \omega_a = \boldsymbol{I_a^{-1}}(r_a \times p)$$

$$r_a = x_{contact} - x_a$$

$$\Delta v_a = \Delta v_{\text{cm\_a}} + \Delta \omega_a \times r_a = \frac{p}{m_a} + \boldsymbol{I_a^{-1}}(r_a \times p) \times r_a = \boldsymbol{K_a}\, p$$

$$\boldsymbol{K_a} = \frac{\boldsymbol{I}}{m_a} - \tilde{r}_a \boldsymbol{I_a^{-1}} \tilde{r}_a$$

$$\Delta u = \Delta v_a - \Delta v_b = \boldsymbol{K}\, p$$

$$\boldsymbol{K} = \boldsymbol{K_a} + \boldsymbol{K_b}$$

$$p = \boldsymbol{K^{-1}} \Delta u$$

Матрица **K** (3x3) зависи од распореда маса тела у односу на контактну тачку и представља инверзну масену матрицу судара. Нека је **t** јединична тангента релативне брзине у тачки контакта. Офсет брзина је $v_a = v_{\text{cm\_a}} + \omega_a \times r_a$. Тела се сударају ако и само се у тачки контакта крећу једно ка другом, тј. $\Delta v_n = (v_a - v_b) n < 0$. Ако овај услов није испуњен тела ће се раздвајати (>0) или остати непомична (=0) дуж нормале. У зависности која врста трења делује даље постоје два случаја [8]:

- Статичко трење (под условом $|p_t| \leqslant \mu_s p_n$)

$$p = \boldsymbol{K^{-1}} \Delta u = \boldsymbol{K^{-1}}[-e\, \Delta v_n n - \Delta v]$$

- Динамичко трење

$$p = p_n n - \mu_d\, p_n t = p_n (n - \mu_d t)$$

$$p_n = \frac{-(1+e)\, \Delta u_n}{n \boldsymbol{K} (n - \mu_d t)}$$

Прво се претпостави да важи статичко трење, па ако услов статичког трења није испуњен примењује се динамичко трење.

Исте функције се могу користити за обраду контаката уз измену да је коефицијент еластичности нула.

Итеративна обрада судара и контаката

У случају да постоји више од два тела која интерагују, сва ограничења морају бити задовољена за сва тела одједном. Сваки пут када два тела размене импулсе постоји могућност да се направе нова кршења ограничена са суседним телима. Ако се више пута понови размена импулса за све парове тела, укупни импулс додат сваком телу ће конвергирати правом импулсу, који задовољава сва ограничења, ако решење постоји [9]. Метода је слична итеративном решавању система линеарних једначина. У прототипу је број итерација фиксиран, пошто се брзини, интерактивности и робустности даје предност у односу на прецизност. Брзина конвергенције обрнуто зависи од ширине контактног графа, јер у случају високих гомила (енг. stack) потребно је разменити више импулса како би се и тела на врху зауставила.

У обради судара и обради контаката се користе исте формуле за размену импулса, уз разлику што код контаката коефицијент еластичности линеарно расте од -1 до 0, од прве до последње итерације, како би се тела постепено успорила [8]. Код обраде контаката је потребна већа прецизност пошто тела могу да се гомилају. Коефицијент еластичности конвергира ка 0 пошто контактне силе нису еластичне.

Корекција позиција

Пошто се тела "слепо" померају на нове позиције могуће је пробијање ограничења до $v\Delta t$ (померај тела за време једног корака). Ради корекције се додаје мали корекциони импулс [11]:

$$\Delta u_e = \frac{b}{\Delta t} max(0, \delta - \delta_{max})$$

Дубина пробијања је означена са δ, а фактор корекције са b. Фактор корекције означава колики се проценат грешке исправи у једном Δt кораку. Вредности за b су обично у између 0,2 и 0,4.

Да би се ограничио утицај корекционих импулса на нестабилност тела брзине се деле на реалне и псеудо [12]. Обрада судара се обавља одвојено и на исти начин, само што на псеудо брзине делују још и корекциони импулси. Псеудо брзине се иницијализују на 0 на почетку сваког Δt корака. На овај начин корекциони импулси не могу директно да утичу на брзину тела. Позиције се интергрирају сумом реалне и псеудо брзине:

$$x_{new} = x_{old} + (v_{real} + v_{bias})\Delta t$$

Предност корекције је што смањује пробијање тела. Недостатак је што привремено уноси додатну енергију у систем, па судари више не могу бити апсолутно еластични, али то не представља проблем ако се симулирају реални материјали који не могу бити апсолутно еластични.

Спојеви

Улога споја (енг. joint) је смањивање степени слободе између два тела. Спојеви који ће бити описани су:

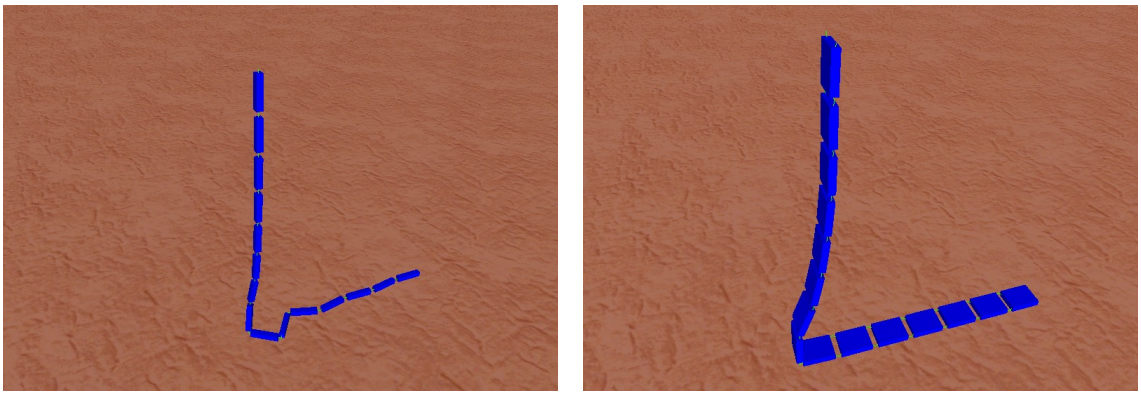
- шипкасти спој - енг. bar joint (растојање између једне тачке на једном и једне тачке на другом телу је фиксирано; нпр два возила спојена крутом везом)
- куглични спој - енг. ball joint (тела имају заједничку тачку око које могу слободно да ротирају)
- шарка - енг. hinge joint (тела имају заједничку осу око које могу слободно да ротирају, али не могу да се транслирају дуж осе)

Шипкасти спој

Нека су A_1 и A_2 тачке на једном и другом телу, респективно, преко којих су тела спојена. Пошто овај спој одржава ове везне тачке на фиксном растојању услов споја је $|\Delta x = A_1 - A_2| = \text{const}$. Квадрирањем и диференцирањем се добија $\Delta x \Delta v = 0$. Смисао овог услова је да везне тачке не могу да се приближавају и удаљавају, већ само да ротирају једна око друге. Промена брзине која поништава нормалну компоненту брзине је: $\Delta u = -\Delta v_n n$. (n је јединични вектор у смеру Δx)

Куглични спој

Куглични спој (слика 3) је специјални случај шипкастог споја када је растојање 0. Шипкасти спој се у овом случају не може користити. Услов кугличног споја је $\Delta x = 0$. Диференцирањем по времену се добија $\Delta v = 0$. Промена брзине која поништава релативну брзину је: $\Delta u = -\Delta v$, или једноставно речено брзине тела у везној тачки морају бити исте.



Слика 3: Примери кугличних (лево) и шаркастих (десно) спојева

Шарка

Овај спој се може једноставно направити употребом два куглична споја, јер ако два тела имају две различите заједничке тачке имаће и заједничку праву која пролази кроз те две тачке. Добро је да те две тачке буду што више удаљене, јер ако су сувише близу спој ће се понашати као куглични.

Откривање судара и контаката

Подсистем за откривање судара и контаката решава чисто геометријски проблем: пронаћи пресеке између n тела. Ово је компликовано урадити за неконвексна тела, па су у овом симулатору тела ограничена на конвексне облике. Проблем се дели на два мања подпроблема који се називају широка и уска фаза откривања судара [2]. Олакшавајућу околност представља то што је проблем који се решава сличан оном који је решен у прошлој итерацији (тела се мало померају између итерација), па не треба све поново да се рачуна.

Широка фаза (енг. broad phase)

У овој фази потребно је за датих n тела пронаћи парове тела која су потенцијално у контакту. Број парова тела је $O(n^2)$, а број контаката O(n). Ова фаза се може имплементирати на више начина:

- грубом силом (енг. brute force)
  предност – једноставност
  мана – временска сложеност $O(n^2)$

- просторно хеширање (енг. spatial hashing)
  Простор се подели у једнаке коцкасте ћелије. Свака ћелија се мапира хеш функцијом у једно поље хеш табеле. Свако тело се убаци у све ћелије које сече. Потенцијални парови су она тела која се налазе у истом пољу хеш табеле.
  предност – време O(n), ако је величина ћелије приближна величини тела
  мана – перформансе опадају ако су тела много већа или много мања од ћелија

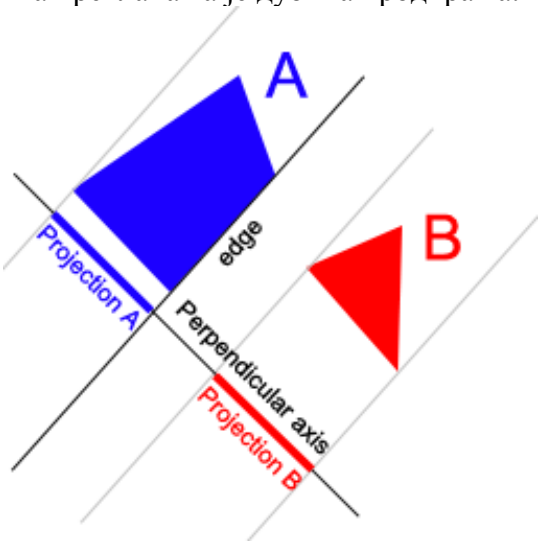
- обухватање и смањивање (енг. sweep and prune)
  Пројектовати тела на све 3 осе и одржавати сортираним крајње тачке интервала пројекција за све 3 осе. Пошто се тела мало померају за Δt време низови се могу сортирати алгоритмом уметања (енг. insertion sort) у O(n) времену. Потенцијални парови су она тела чије се пројекције преклапају на све три осе. Постоји и варијанта алгоритма која узима у обзир и брзине тела (енг. kinetic sweep and prune, [13]).
  предност – не зависи од величина тела као просторно хеширање
  мана – просторна комплексност $O(n^2)$; у случају да се пројекције великог броја тела преклапају на бар два осе перформансе опадају на $O(n^2)$

Уска фаза (енг. narrow phase)

У овој фази за парове потенцијалних тела из широке фазе треба одредити да ли су у контакту. Ако јесу треба одредити и тачку контакта, нормалу у контактној тачку и дубину продирања (дубина се користи само за корекцију). Брз тест који треба урадити на почетку је провера пресека описаних сфера. Ако се описане сфере не секу, тела не могу бити у контакту.

Коришћен је алгоритам заснован на теореми о раздвајајућим осама (енг. separating axis theorem) [14] за одређивање нормале и дубине, а за контактну тачку се узима геометријски центар пресека тела. Суштина SAT алгоритма је пројектовати тела на кандидат осе (слика 4). Кандидати за осе су нормале страница тела и векторски производи ивица једног и другог тела. Пројекција тела на оси је интервал. Ако се интервали не секу на једној оси

онда тела нису у контакту, иначе се памти оса са најмањом дужином преклапања. Та оса је нормала контакта, а дужина преклапања је дубина продирања.



Слика 4: Приказ једног корака SAT алгоритма.
Пошто се пројекције не секу, онда се ни тела не секу.

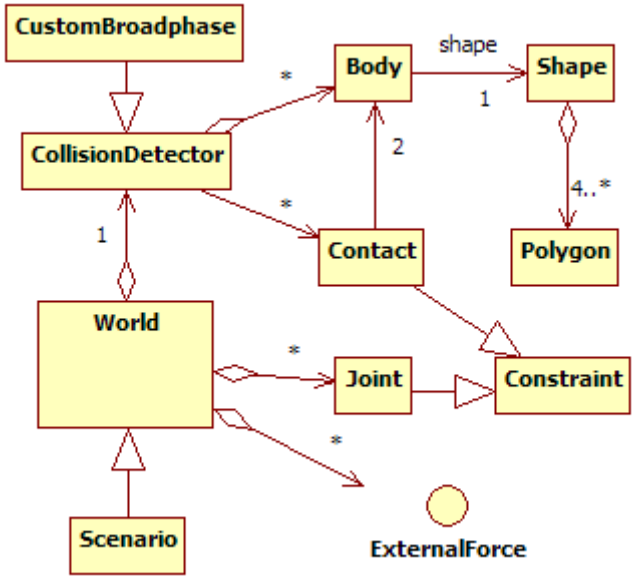
Имплементација

Прототип описаног симулатора је у потпуности имплементиран у програмском језику Јава и развојном окружењу Eclipse. Анимирање сцена је рађено коришћењем OpenGL-а.

	величина у КБ	број класа
језгро симулатора	53	21
цртање и управљање	27	7
сценарији	35	23
геометрија	115	29
помоћне класе	15	7
УКУПНО	245	87

Табела 1: Метрике имплементираног прототипа

Величине и број класа појединих компонената прототипа дати су у табели 1.



Слика 5: Дијаграм класа језгра симулатора

Класе које чине језгро симулатора (слика 5):

- World – садржи сва тела (кроз CollisionDetector), спојеве и екстерне силе
- Scenario – иницијализује свет са телима, спојевима и екстерним силама
- CollisionDetector – проналази контакте између парова тела
- CustomBroadphase – имплементација алгоритма широке фазе
- Body – садржи облик тела и физичке променљиве које описују тело

- Constraint, Contact, Joint – ограничење између два тела које може да делује импулсима на њих; контакти су објекти привременог типа, а спојеви трајног
- ExternalForce – екстерна сила која делује на једно или више тела
- Shape – конвексни полиедар са центром масе у (0 0 0), описан својом површином
- Polygon – конвексни полигон у 3Д простору

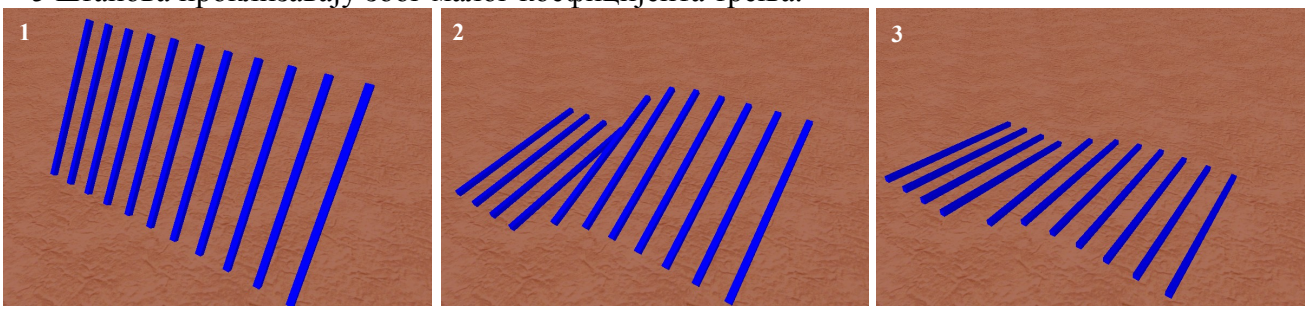
Сценарији су имплементирани као plugin-ови, па се проналазе и учитавају на почетку програма.

Већина геометријских објеката (вектори, кватернице, матрице, трансформације, праве и равни) је имплементирано користећи шаблон Immutable што је резултирало поједностављањем кода симулатора, пошто се операције над њима могу вршити у облику математичких израза, уместо наредби налик асемблерским које мењају стање објекта.

Примери који следе су симулирани у реалном времену на лаптоп рачунару са AMD Turion 64 мобилним процесором на 2.2GHz, 1GB RAM меморије и Sun-овој серверској јава виртуелној машини из пакета JDK 6.

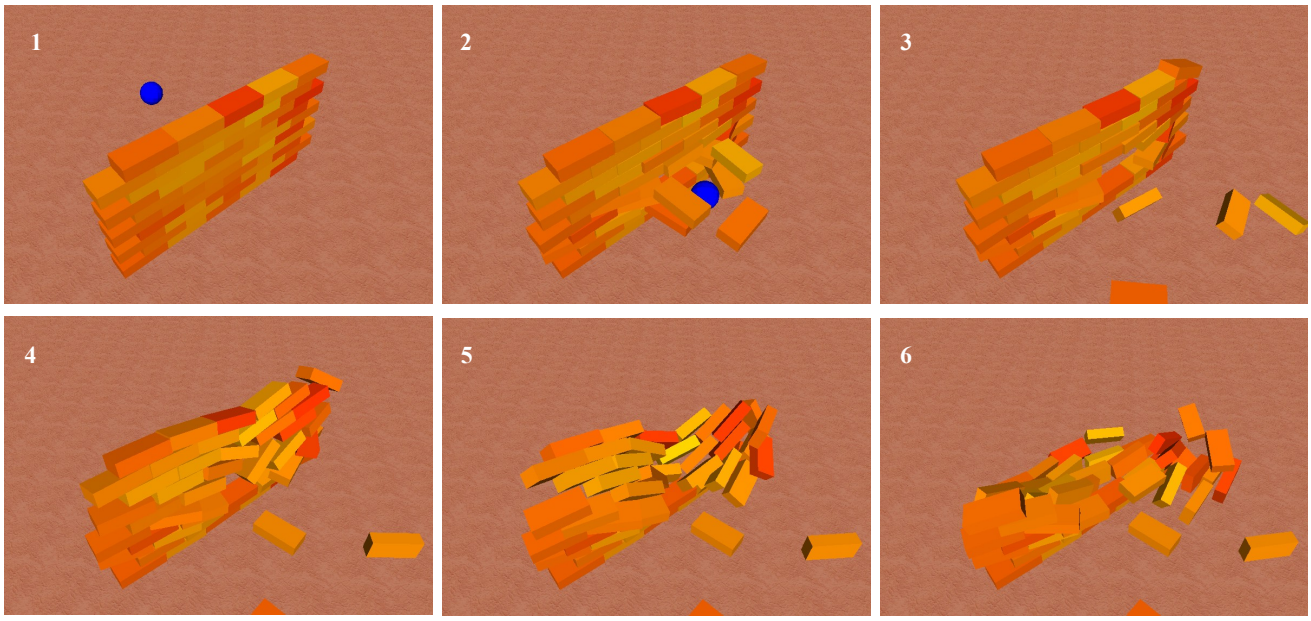
Примери симулација

На слици 6 је приказано деловање статичког трења на штапове у паду. Коефицијент статичког трења линеарно расте од 0 за први штап (са лева) до 1 за последњи штап. Првих 5 штапова проклизавају због малог коефицијента трења.



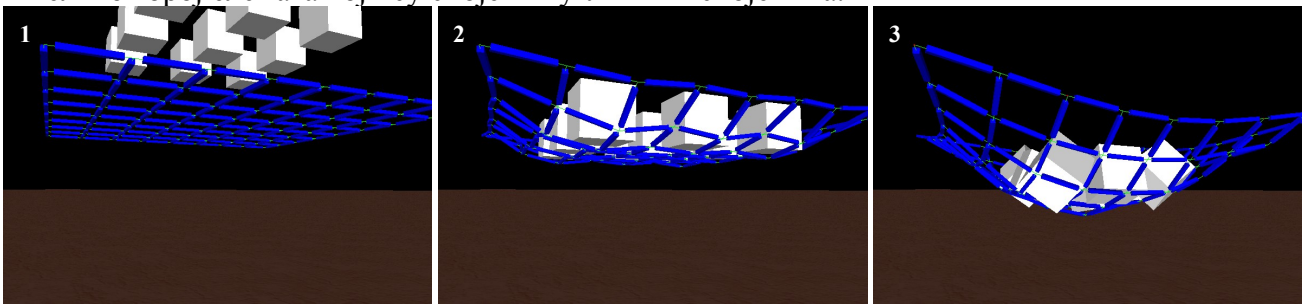
Слика 6: Проклизавање штапова са мали трењем

На слици 7 је приказан ефекат гомилања. Конфигурација зида од 50 цигала је стабилна (сем једне цигле у ћошку која проклизава). Импулси се пропагирају од подлоге нагоре.

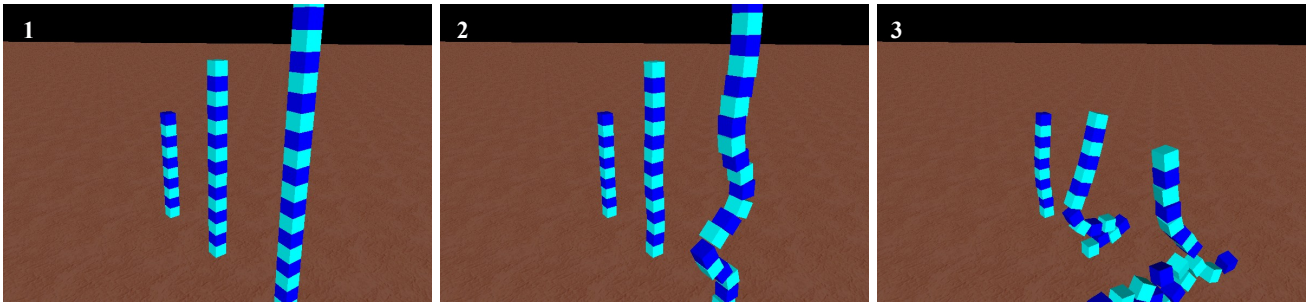


Слика 7: Пробијање зида куглом.

На слици 8 је приказан флексибилни комплексни објекат (мрежа) који се састоји од великог број елемата који су спојени кугличним спојевима.

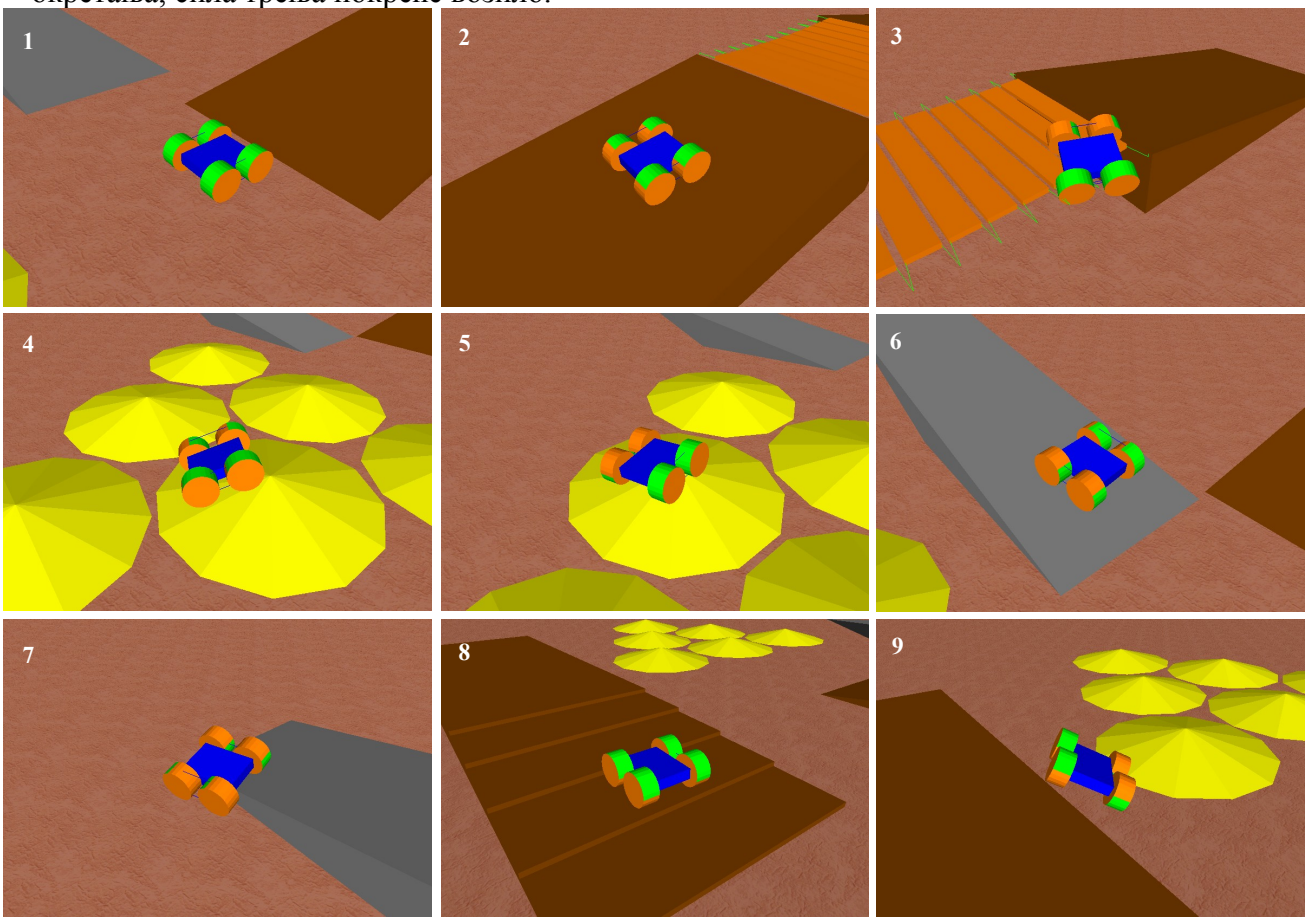


Слика 8: Мрежа која хвата коцке у паду.

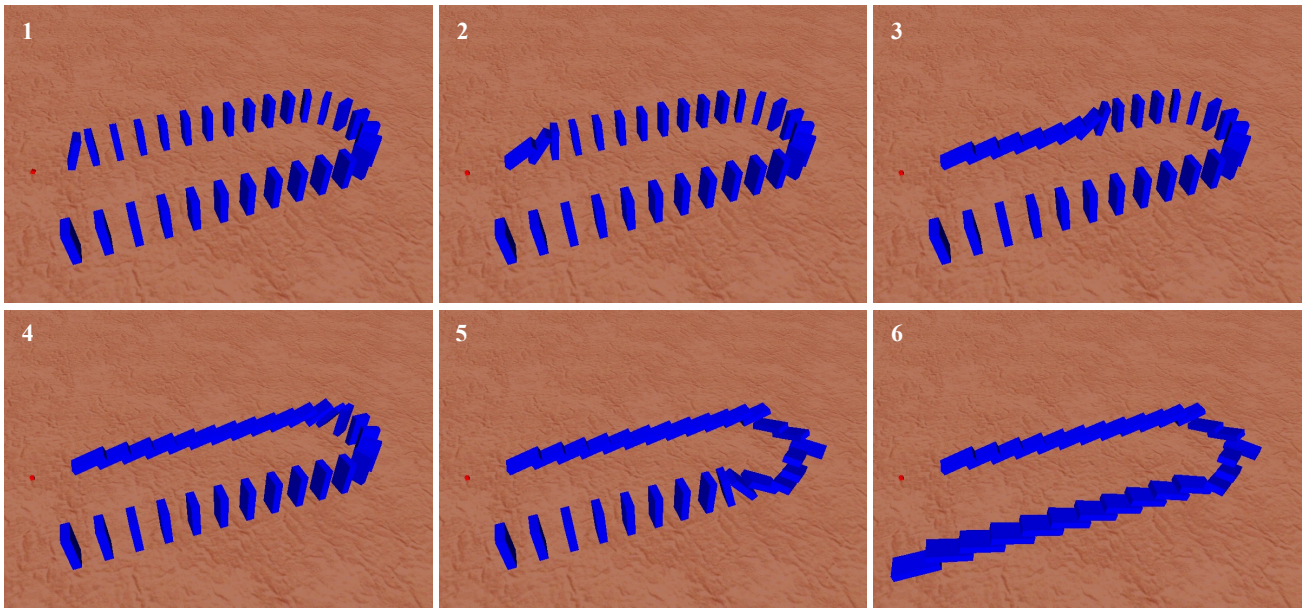


Слика 9: Торњеви од 10, 15 и 25 коцака

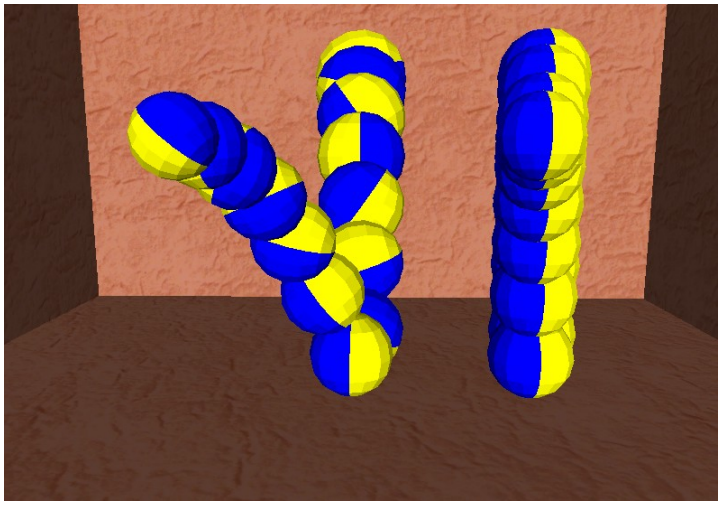
На слици 10 је приказан пример интеракције са корисником. Корисник у реалном времену управља возилом на терену са препрекама: коса раван, висећи мост, купе, покретна рампа, степениште. Висећи мост се састоји од шаркастих спојева. Применом момента силе у истом смеру на све точкове возило се креће напред и назад, а ако су моменти силе у супротним смеровима возило скреће. Леви и десни точкови су спојени са по два шипкаста споја како би се окретали у фази. Точкови притискају подлогу, па услед њиховог окретања, сила трења покреће возило.

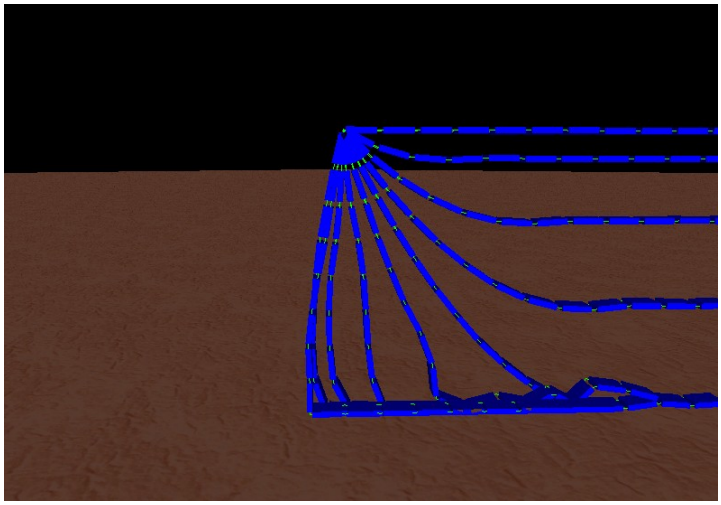


Слика 10: Возило на терену са препрекама.



Слика 11: Домино ефекат


Слика 12: Тангенциони импул при судару услед ротације лопте и трења са подлогом.


Слика 13: Ланац са шаркастим спојевима у слободном паду.

Закључак и даљи правци развоја

У овом раду је приказан импулсни метод симулирања динамике крутих тела. Применом те методе имплементиран је прототип симулатор. Тестови са прототипом су показали да се интерактивне брзине симулације могу достићи са умерено комплексним физичким системима и да на перформансе највише утиче модул за откривање колизија.

Могући даљи правци развоја су:

- Додавање угаоног трења (котрљање и обртање)
  Без угаоног трења кугле и ваљци који се котрљају без проклизавања се не могу зауставити, пошто је релативна брзина у тачки контакта са подлогом 0.
- Конкавни облици тела
  За тела која су природно конкавна није практично делити их на велики број конвексних.
- Оптимизација редоследа обраде судара и контаката
  Ово може да доведе до убрзавања конвергенције и побољшању перформанси, као што је shockwave propagation метод [8].
- Спречавање тунелирања брзих објеката
  Пошто се откривање контаката одвија у дискретним тренутцима тела која су довољно брза могу потпуно да прођу кроз друга тела у једном кораку симулације.
- Прецизнија интеграција
  Интеграторима као што је Рунге Кута реда 4 може се смањити акумулирана грешка интеграције на $O(\Delta t^4)$ [15].
- Мотори на спојевима са серво контролерима
  Са серво моторима се могу симулирати возила и машине.
- Имплементација за процесоре са више језгара / мултипроцесорске системе / графичким процесорима
  Импулсни метод је веома погодан за паралелизацију пошто импулси који се размењују делују локално и зависе само од пара тела. Проблем скалабилности евентуално може представљати широка фаза откривања контаката.

Прилог - Рачунање масених особина полиедара

У овом раду, полиедар представљамо скупом троуглова који чине његову површину. Темена троуглова, гледано споља, су оријентисана супротно од казаљке на сату. Претпоставља се да је густина полиедра хомогена и једнака 1. Полиедар не мора да садржи тачку (0 0 0) и може да буде конкаван.

Запремина и центар масе

Идеја алгоритма је поделити полиедар на тетраедре, израчунати запремину и центар масе појединачних тетраедара и на крају сумирати израчунате величине за све тетраедре.

Тетраедри се праве тако да се састоје од три темена (a b c) из једног троугла полиедра и тачке 0. Запремина и центар масе тетраедра су:

$$V_i = \frac{a \cdot (b \times c)}{4}$$
$$X_i = \frac{a + b + c}{4}$$

Запремине и центри маса тетраедара се сумирају следећим формулама:

$$V = \sum V_i$$
$$\vec{X}\, V = \sum \vec{X}_i V_i$$

```
VolumeAndCenterOfMass(Triangle[] P)
     C = Vector(0 0 0)
     Volume = 0
     for each Triangle(a b c) in P do
          v = a dot (b cross c) / 4
          C += v * (a + b + c) / 4
          Volume += v
     end
     CenterOfMass = C / Volume
end
```

Тензор момента инерције

Момент интерције се рачуна преко матрице коваријансе [16]. Полиедар се дели на тетраедре и за сваки се рачуна коваријанса трансформацијом из каноничног тетраедра. Коваријанса се акумулира за сваки тетраедар и на крају се претвара у тензор момента инерције. Претпоставља се да је центар масе полиедра у (0 0 0).

```
InertiaTensor(Triangle[] P)
                        2 1 1
     canonical = Matrix(1 2 1) / 120
                        1 1 2
     C = Matrix3(0)
     for each Triangle(a b c) in P do
                       a
          A = Matrix(b)
                       c
          C += det(A) * transpose(A) * canonical * A
     end
     InertiaTensor = trace(C) * IdentityMatrix - C
end
```

Литература

[1] Kenny Erleben, *"Stable, Robust, and Versatile Multybody Dynamics Animation"*, PhD Thesis

[2] Brian Mirtich, *"Efficient Algorithms for Two-Phase Collision Detection"*, 1997,
Mitsubishi Electic Research Laboratory

[3] David Baraff, *"An Introduction to Physically Based Modeling:
Rigid Body Simulation I - Unconstrained Rigid Body Dynamics"*
Robotics Institute, Carnegie Mellon University, 1997

[4] Енглеска Википедија, *"Quaternions and spatial rotation", 2007*

[5] више аутора, *"The Matrix and Quaternions FAQ"*, 2003

[6] Енглеска Википедија, *"Symplectic Euler method"*, 2007

[7] Енглеска Википедија, *"Geometric integrator"*, 2007

[8] Eran Guendelman, Robert Bridson, Ronald Fedkiw,
"Nonconvex Rigid Bodies with Stacking", Stanford University, SIGGRAPH 2003

[9] Jan Bender and Alfred Schmitt, *"Fast Dynamic Simulation of Multi-Body Systems Using Impulses"*, Virtual Reality Interactions and Physical Simulations (VRIPhys), Madrid, November 6-7, 2006

[10] Kevin Egan, *"Techniques for Real-Time Rigid Body Simulation"*, Thesis
Brown University, 2003

[11] Erin Catto, *"Fast and Simple Physics using Sequential Impulses"*
GameDevelopers Conference 2006

[12] Erin Catto, *"Modeling and Solving Constraints"*, GameDevelopers Conference 2007

[13] Coming, Staadt, *"Kinetic Sweep and Prune for Multi-body Continuous Motions",*
in "Virtual Reality Interaction and Physical Simulation", Computers & Graphics,
Vol. 30, No. 3, 2006.

[14] David Eberly, *"Intersection of Convex Objects: The Method of Separating Axes"*
Geometric Tools, Inc. 2003

[15] Енглеска Википедија, *"Runge-Kutta method"*, 2007

[16] Jonathan Blow, Atman Binstock, *"How to find the inertia tensor (or other mass properties) of a 3D solid body represented by a triangle mesh", 2004*