UDC: 514.7:004.7 Info M: str. 41-47

VIZUELNI SIMULATOR KRETANJA ČESTICA POD UTICAJEM VEKTORSKIH POLJA
VISUAL SIMULATOR MOTION OF PARTICLES UNDER THE INFLUENCE OF VECTOR FIELDS

Vladimir Divjak, Dražen Drašković, Bojan Furlan, Boško Nikolić
Elektrotehnički fakultet, Univerzitet u Beogradu

REZIME: Vizuelni simulator kretanja čestica pod uticajem vektorskih polja realizovan je kao alat koji pomaže prilikom vizuelizacije kompleksnih interferencija vektorskih polja i rezultujućeg efekta nad česticama, odnosno boljem razumevanju naše fizičke okoline. U radu je objašnjen cilj implementiranog modela, način upotrebe aplikacije i predstavljeni su korišćeni API-ji HTML5 tehnologije. U simulatoru su korišćeni WebGL API za direktan pristup grafičkom adapteru, WebGL THREE. js biblioteka, koja apstrahuje operacije nad grafičkim adapterom i pruža pristup objektno-orijentisanom modelu sistema čestica i Web Worker API koji pruža multiprocesorsku podršku u okviru jezika JavaScript. Simulator koristi jednostavan otvoreni model čestica koji se može izmeniti ili proširiti, kao i otvorenu implementaciju matematičkog modela vektorskih polja, što dozvoljava proširenje skupa vektorskih polja, koja se mogu simulirati. Aplikacija je u potpunosti klijentska i izvršava se u bilo kom novijem veb pregledaču. Simulator se može koristiti u naučno-istraživačke i obrazovne svrhe. Na Elektrotehničkom fakultetu u Beogradu koristi se u okviru nastave iz Veb dizajna, na drugoj godini studijskog programa Softversko inženjerstvo, za prikazivanje mogućnosti HTML5 standarda.

KLJUČNE REČI: vektorska polja, HTML5 tehnologija, WebGL, vizuelni simulator, edukacija

ABSTRACT: Visual simulator motion of particles under the influence of the vector field is implemented as a tool that helps in the visualization of complex interference of vector fields and the resulting effect on the particles, and for a better understanding of our physical environment. The paper explains the goal of the implemented model, application usage and presents APIs of HTML5 technology. The simulator have been used WebGL API for direct access to the graphics adapter, WebGL library THREE.js, which abstracted operations on graphics adapter and provides access to the object-oriented model of the system of particles and Web Worker API that provides multiprocessor support within the JavaScript language. Simulator uses a simple open model of particles that can be easily modified or extended, as well as open implementation of the mathematical model of vector fields, which allows easy extension of the set of vector fields, which can be simulated. The application is completely client-based and executed in any new web browser. The simulator can be used for scientific-research and educational purposes. At the University of Belgrade, School of Electrical Engineering, this simulator is used in the teaching process for the Web design course, in the second year of the study program Software Engineering, to show the possibilities of HTML5 standards.

KEY WORDS: fields, HTML5 technology, WebGL, visual simulator, education

1. UVOD

Iz perspektive prirodnih nauka, čitav univerzum sastoji se od čestica, okarakterisanih atributima, poput mase, naelektrisanja, spina i drugih, koje se kreću pod uticajem vektorskih polja koja ih okružuju. Neke od ovih fundamentalnih modela stvarnosti koje spadaju u znanja stečena iz matematike i fizike, gotovo je nemoguće vizuelizovati tradicionalnim tehnikama. Zato je razvijen alat koji je prilagođen korisnicima i u mogućnosti je da grafički predstavi pomenute modele. Alat se može koristiti pri istraživanju matematičkih modela prirode, kako već postojećih implementiranih modela, tako i korisnički definisanih.

Pored samih funkcionalnosti, u ovom radu su opisane i upotrebljene nove HTML5 tehnologije, odnosno API-ji ovog standarda. Prednosti tih API-ja mogu se videti u naprednim aplikacijama [1], kao što je ovaj vizuelni simulator, jer se kombinuje: 1) složena interna logika i model podataka uz implementaciju strategije obrade podataka i upotrebu multiprocesiranja; 2) grafička vizuelizacija tj. upotreba naprednih komponenti računara; 3) složen korisnički interfejs.

Rad je struktuiran na sledeći način: u poglavlju 2 dat je pregled postojećih rešenja i dostupnih tehnologija, u poglavlju 3 opisana je arhitektura sistema, u okviru koga je opisan i izgled korisničkog interfejsa, kao i način realizacije simulatora. U poglavlju 4 prikazani su rezultati simulacije, a u poglavlju 5 dat je zaključak.

2. PREGLED POSTOJEĆIH REŠENJA I DOSTUPNIH TEHNOLOGIJA

Vizuelni simulatori danas se sve više koriste u edukaciji. Njihova upotreba se može uočiti i u predavanjima naprednih matematičkih studijskih programa, kao što su programi inženjerskih nauka. Hennig i njegovi saradnici [2] sa Univerziteta Paderborn su predložili pristup mešovitog scenarija učenja koji olakšava prenošenje matematičkih znanja i primenili su ga na primeru kursa iz Osnova elektrotehnike. Ključni element ovog pristupa je bogata upotreba interaktivnih veb aplikacija zasnovanih na 3D vizuelizaciji, što omogućava razjašnjavanje složenih matematičkih problema u vrlo kratkom vremenskom periodu. S druge strane, Xue sa saradnicima [3] predlaže vizuelizaciju panoramskog pogleda na geomagnetske linije magnetnog polja Zemlje upotrebom WebGL tehnologije u okviru HTML5 standarda. Naime, Zemljino magnetno polje je podeljeno na glavno i promenljivo magnetno polje. IGRF (*International Geomagnetic Reference Field*) predstavlja međunarodno priznat model glavnog magnetnog polja danas, a za promenljivi model magnetosfere se koriste *Tsyganenko* modeli. Upotrebom mapiranja geografske širine i dužine u prostornu lokaciju, predloženi pristup linearno kombinuje ova dva modela, dobijajući linije magnetnog polja i njihovu odgovarajuću snagu koje se koriste za vizuelizaciju. Yikang sa saradnicima [4] koristi WebGL tehnologiju i model prostora za vizuelnu simulaciju prostornog okruženja. Oliker sa saradnicima [5]

predstavio je simulacionu platformu za podršku obrazovanju plastične hirurgije. Ova platforma predstavlja sveobuhvatno virtuelno obrazovno okruženje koje omogućava kompleksnu, korak po korak simulaciju u realnom vremenu, uključujući interaktivnu 3D vizuelizaciju, kombinovanje hirurških snimaka uživo, tekstualnih i glasovnih oznaka, kao i stručno odobrenog načina testiranja.

Kao najvažniji kriterijumi prilikom izbora vizuelnih simulatora najčešće se pominju: pouzdanost, raspoloživost, prenosivost, skalabilnost i performanse. Simulator predstavljen u ovom radu namenjen je za izvršavanje na klijentskoj strani, odnosno u veb pregledaču, pa je interna logika aplikacije, kao i korisnički interfejs realizovan u jeziku *JavaScript*. Matematički modeli vektorskih polja su takođe realizovani u otvorenom formatu, kroz transparetnu implementaciju formule u *JavaScript* skripti, što omogućava izmene i proširenja skupa vektorskih polja, koje se mogu simulirati.

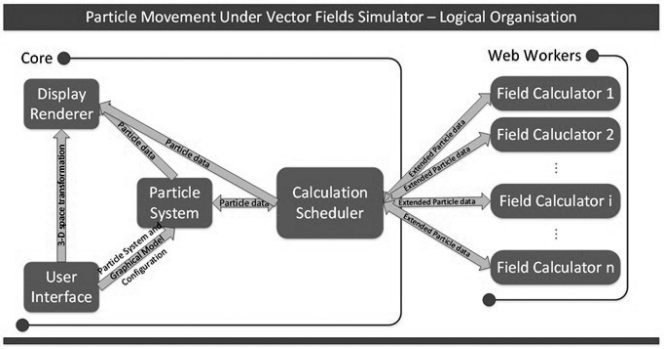
Kao jedan od zahteva prilikom izrade aplikacije, postavljen je uslov da na modernom četvorojezgarnom procesoru treba tečno simulirati kretanje do 100 hiljada čestica, a uz određene zastoje pri iscrtavanju, omogućiti simulaciju i kretanje i do milion čestica. Stoga, aplikacija za izvršavanje koristi sve dostupne resurse računara, odnosno sva jezgra procesora i grafički adapter, koliko to upotrebljeni interfejs dozvoljava. Za multiprocesorsku podršku u okviru jezika *JavaScript* upotrebljen je HTML5 *Web Worker API* [6][7].

Za iscrtavanje grafičkih elemenata upotrebljena je tehnologija WebGL, grafička biblioteka dostupnu u okviru HTML5 standarda, odnosno WebGL *wrapper* biblioteka THREE.js [8] koja dodatno olakšava instanciranje grafičkog konteksta, rad sa osvetljenjem i kamerom u okviru scene, kao i objektno-orijentisani model sistema čestica što dozvoljava veliku fleksibilnost pri dinamičkoj izmeni grafičkog modela i grafičkog konteksta.

Kako veb pregledači *Google Chrome* i *Mozilla Firefox* prednjače u implementaciji najnovijih standarda i tehnologija, konkretno HTML5 standarda i WebGL tehnologije, oni predstavljaju pogodne alate za razvoj aplikacije opisane u ovom radu. Pored implementacije pomenutih standarda, ovi pregledači se odlikuju odličnim razvojnim alatima koji se mogu koristi u samom pregledaču, konkretno *Google Chrome Developer Tools* i *Mozilla Firefox Web Developer Tools*, koji pružaju mogućnost pregleda učitanih resursa, kao i debagovanja i interpretiranog rada sa jezikom *JavaScript*. Za razvoj ovog vizuelnog simulatora korišćeni su veb pregledači *Google Chrome*, verzija 32 i *Mozilla Firefox*, verzija 26.

3. ARHITEKTURA SISTEMA I OPIS REALIZACIJE

Aplikacija na najvišem nivou apstrakcije podeljena je u dva logička segmenta, kao što je prikazano na slici 1. Prvi segment - *Core* je odgovoran za internu logiku aplikacije i interakciju sa korisnikom i kao takav sadrži podatke o sistemu čestica (*Particle System*), korisnički interfejs (*User Interface*), grafički displej (*Display Renderer*), kao i internu logiku sinhronizacije proračuna vektorskih polja i iscrtavanja podataka (*Calculation Scheduler*). Drugi logički segment - *Web Workers* sadrži implementacije matematičkih modela vektorskih polja (*Field Calculators*) koje se izvršavaju u posebnim nitima i koje su pod upravom interne logike sinhronizacije segmenta *Core*.



Slika 1: *Logička organizacija elemenata rešenja simulatora kretanja čestica pod uticajem vektorskih polja*

Particle System je komponenta koja sadrži celokupnu geometrijsku konfiguraciju čestica kao i grafički model koji se koristi pri iscrtavanju istih, implementiran kroz interfejs biblioteke *THREE.js*.

Geometrijska konfiguracija čestica predstavljena je kroz *THREE.Geometry* objekat koji sadrži sve podatke neophodne za opisivanje 3-D modela, kao i pomoćne geometrijske funkcije za izračunavanje tipičnih podataka potrebnih za geometrijske transformacije. Od najvećeg interesa za potrebe ovog rada jeste *vertices* polje objekta *THREE.Geometry* koje je implementirano kao niz *THREE.Vector3* objekata koji se koriste za reprezentaciju čestica. Ovi objekti sadrže koordinate čestice u trodimenzionalnom prostoru, odnosno koordinate čestice po x, y i z osi euklidskog koordinatnog sistema.

Grafički model čestica predstavljen je kroz *THREE.Material*, odnosno *THREE.ParticleSystemMaterial* objekat koji sadrži sve podatke neophodne za opisivanje detalja grafičkog iscrtavanja čestica kao što su veličina, boja, providnost itd. Konfiguracija geometrije i grafičkog modela moguća je kroz korisnički interfejs.

Podaci o geometriji sistema čestica se iz objekta *Particle System* prosleđuju *Display Renderer* komponenti samo pri iscrtavanju prve slike simulacije, odnosno neposredno nakon korisničke konfiguracije ili učitavanja postojeće konfiguracije, nakon čega se ovi podaci kopiraju u *Calculation Scheduler* komponentu. Ova komponenta je odgovorna za raspodelu podataka po nizu *Field Calculator* komponenti, nadležnih za transformaciju geometrije sistema čestica pod uticajem vektorskih polja, pa se prilikom početka simulacije podaci o geometriji prosleđuju *Display Renderer*-u direktno iz *Calculation Scheduler*-a radi uštede vremene i izbegavanja nepotrebnog kopiranja velikog broja podataka u *ParticleSystem* komponentu.

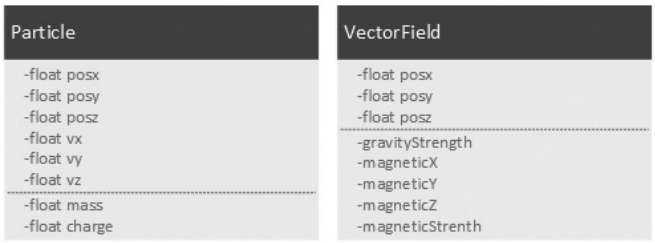
Obzirom da se atributi *verticesNeedUpdate* i *colorsNeedUpdate*, koji označavaju da je potrebno ažurirati podatke o geometriji i grafičkom modelu sistema čestica u memoriji grafičkog adaptera, postavljaju na vrednost *true* pri svakom pozivu funkcije za iscrtavanje, što rezultuje kopiranjem svih podataka iz atributa *vertices* i *colors*, a kako se radi o relativno

velikom broju podataka, ovo predstavlja vremenski zahtevnu operaciju. Radi unapređenja performansi aplikacije, funkcije kopiranja ovih podataka su u ovom radu izmenjene tako da kopiraju podatke na nivou celog niza i to iz programske strukture definisane u *Calculation Scheduler*-u koja nakon svake iteracije simulacije vektorskih polja čuva najsvežije podatake o poziciji i boji čestica.

Komponenta *Display Renderer*, koja sadrži funkcije potrebne za komunikaciju sa WebGL API-jem, čime se dobija kontrola grafičkog adaptera računara (GPU) visokog nivoa apstrakcije, implementirana je kroz *THREE.WebGLRenderer* objekat i sadrži sve funkcije neophodne za instanciranje grafičkog konteksta i podešavanje destinacije iscrtavanja.

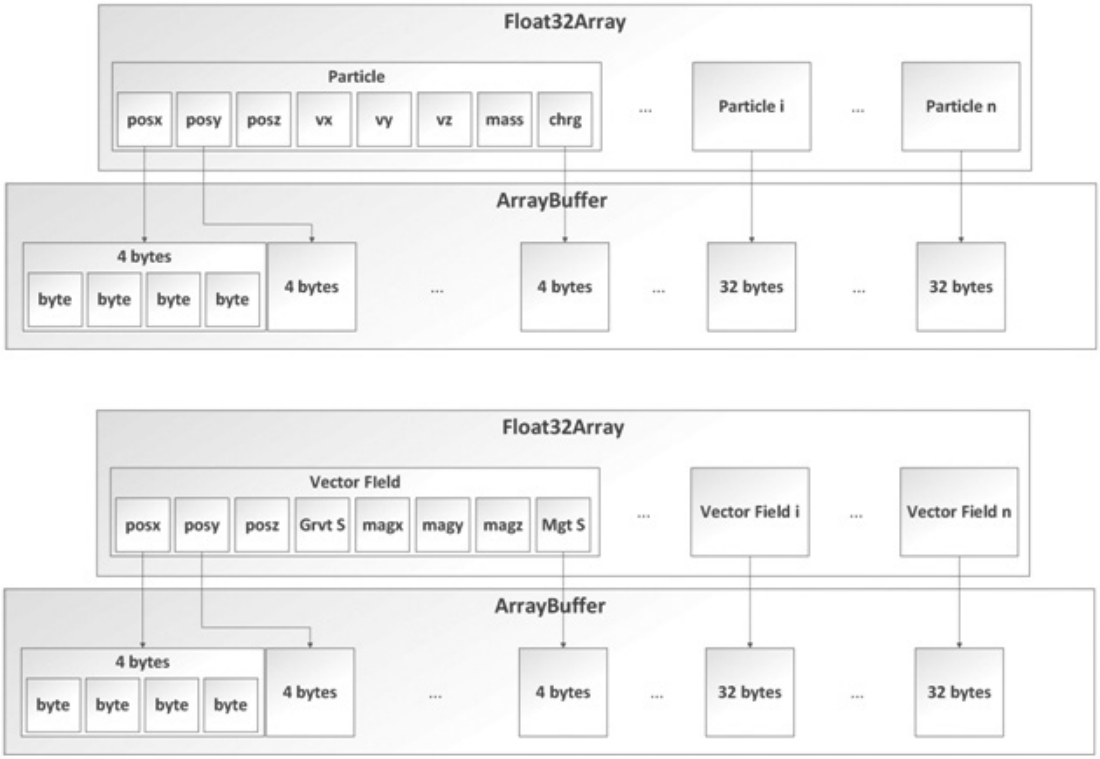
Komponenta *Calculation Scheduler* je segment koda interne logike aplikacije koji se koristi za raspodelu podataka o geometriji čestica po n *Field Calculator* komponenti, implementiranih kroz *Web Worker API*, kao i sinhronizaciju ovih komponenti, odnosno niti u kojima se kalkulacije izvršavaju, a zatim i konsolidaciju podataka u konsekutivan niz podataka pogodan za prosleđivanje *Display Renderer* komponenti.

Za format poruka pri razmeni podataka između glavne programske i računarskih niti razvijene su struktura podataka za modelovanje čestica (*Particle*) i struktura podataka za modelovanje vektorskih polja (*VectorField*). Ove strukture osmišljene su tako da uključuju sve podatke neophodne za sprovođenje simulacije uticaja gravitacionog i magnetnog vektorskog polja. Strukture je moguće proširiti podacima neophodnim za implementaciju drugih vektorskih polja. UML dijagrami struktura *Particle* i *VectorField* dati su na slici 2.



Slika 2: *UML dijagrami Particle strukture podataka i VectorField strukture podataka*

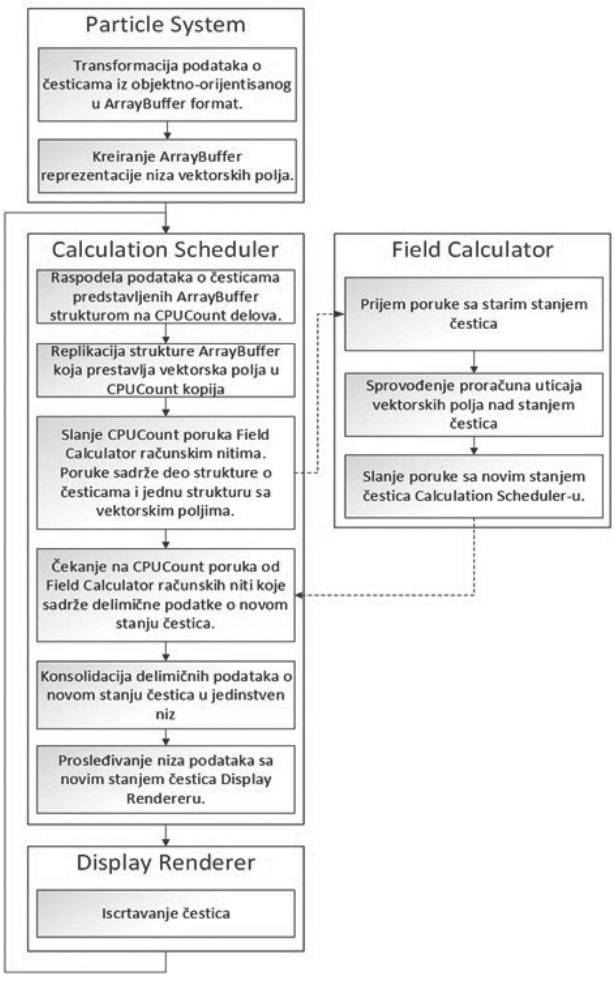
Ove strukture dizajnirane su tako da budu pogodne za čuvanje u *ArrayBuffer* objektima (slika 3), koji se koriste pri razmeni poruka uz upotrebu *TransferableObject* tehnologije, koja omogućuje prenos objekata iz jednog konteksta niti u drugi, bez kopiranja istih, što rezultuje znatno bržom komunikacijom. **TransferableObject** implementacija garantuje da se podaci pri transferu iz dokumenta u *Web Worker* skriptu i/ili u suprotnom smeru ne kopiraju iz jednog konteksta u drugi. Ovde se radi o operaciji predavanja kontrole, što u velikoj meri popravlja performanse razmene poruka sa *Web Worker* skriptama. Ova tehnologija je slična prosleđivanju po referenci koju srećemo u drugim programskim jezicima. Međutim kod prosleđivanja po referenci radi se o prosleđivanju pokazivača na strukturu podataka, dok se kod *TransferableObject* tehnologije sadržaj objekta prebacuje u kontekst niti koja je odredište poruke i referenca na objekat u niti pošiljaoca poruke nakon slanja poruke neće više biti validna. Ovim se ujedno postiže i međusobno isključivanje pri pristupu objektu, jer će u jednom trenutku referenca na objekat biti validna samo u jednoj od *JavaScript* skripti (bilo skripti HTML dokumenta ili *Web Worker* skripti).



Slika 3: *Float32Array i ArrayBuffer formati Particle strukture i VectorField strukture*

Kako se radi o aplikaciji koja sprovodi intenzivne proračune nad velikim brojem podataka (čestica), a obzirom da su proračuni nad svakom od čestica nezavisni od proračuna nad drugim česticama, radi se o računski zahtevnom problemu koji se može paralelizovati (eng. *embarrassingly parallel problem*).

Podaci su predstavljeni kroz *ArrayBuffer* strukturu koju je potrebno proslediti *Field Calculation* nitima, odnosno *Web Worker* skriptama. Zbog razmene poruka potrebno je pre raspodele podataka po nitima kreirati zasebne *ArrayBuffer* objekte koji će sadržati disjunktne podskupove podataka o stanju čestica, a zatim ih takve zajedno sa zasebnim *ArrayBuffer* objektima koji predstavljaju podatke o vektorskim poljima sistema, proslediti *Field Calculation* nitima. *ArrayBuffer* objekat, koji predstavlja stanja čestica u datom momentu se deli na n sličnih struktura, koje se koriste za izračunavanje uticaja vektorskih polja. Svaki od ovih *ArrayBuffer* objekata nakon raspodele sadrži jednak broj podataka o česticama, odnosno predstavlja podatke o ravnomerno raspoređenom broju čestica. Nakon upućivanja poruka *Field Calculation* nitima, *Calculation Scheduler* čeka na prijem poruka od strane *Field Calculation* niti. Ove poruke sadrže *ArrayBuffer* objekte koji predstavljaju stanja čestica nakon sprovedenih kalkulacija aktivnih vektorskih polja sistema, odnosno nova stanja čestica. Algoritam *Calculation Scheduler*-a dat je na slici 4.

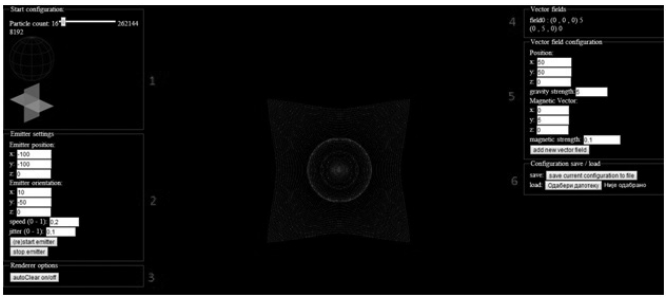


Slika 4: *Algoritam Calculation Scheduler sa komponentom Field Calculator*

Field Calculator je komponenta koja sprovodi matematičke kalkulacije vektorskih polja nad nizom podataka koji predstavljaju geometriju sistema čestica u datom koraku simulacije. *Field Calculator* komponenta se koristi u kombinaciji sa *Web Worker* tehnologijom, odnosno čini telo *Web Worker* skripte. Algoritam komponente *Field Calculator*-a dat je na slici 4.

A. OPCIJE VIZUELNOG SIMULATORA

Korisnički interfejs ovog simulatora predstavlja skup funkcija koje se koriste za potrebe konfiguracije geometrije i grafičkog modela sistema čestica, odnosno za kretanje po 3-D prostoru [9] u kome se sistem čestica nalazi. Ove funkcije su implementirane u jeziku *JavaScript*. Elementi korisničkog interfejsa prikazani su na slici 5.



Slika 5: *Korisnički interfejs simulatora: (1) konfiguracija broja čestica, (2) konfiguracija emitera čestica, (3) kontrola efekata, (4) i (5) konfiguracija vektorskih polja, (6) snimanje u fajl i učitavanje iz fajla*

Aplikacija je projektovana imajući u vidu operaciju nad velikim brojem čestica. Aplikacija za sprovođenje kalkulacija uticaja vektorskih polja koristi 16 računskih niti, što je broj koji je pri testiranju na različitim procesorima dao najujednačenije rezultate, ali se aplikacija može podesiti da koristi veći ili manji broj niti prema hardveru na kome se izvršava. Zbog ovog konstantnog parametra, broj čestica sa kojima aplikacija operiše je uvek multipl broja 16. Korisnički interfejs pruža mogućnost podešavanja broja čestica upotrebom HTML5 *input* elementa u vidu kontrole klizača u okviru *Start Configuration* dela korisničkog interfejsa, prikazanom kao segment 1 na slici 5. Nakon pomeranja klizača, broj čestica će se skalirati na najbliži multipl broja 16, nakon čega će simulacija krenuti iz trenutno aktivne konfiguracije, sa novopodešenim brojem čestica. Upotrebom korisničkog interfejsa moguće je odabrati broj čestica u rasponu od 16 do 262 144.

Korisnički interfejs pruža mogućnost konfiguracije početnog stanja (pozicije) čestica u sledećim oblicima: (1) čestice raspoređene u sferu radijusa 50 sa centrom u vektoru (0,0,0); (2) čestice raspoređene u ravni X; (3) čestice raspoređene u ravni Y; (4) čestice raspoređene u ravni Z. Pored ovih pozicija čestica, aplikacija pruža mogućnost postavljanja čestica u sistem u vidu simuliranog emitera čestica, prikazanog kao segment 2 na slici 5. Emiter čestica je modelovan skupom podataka: vektor početne pozicije čestice, vektor početne brzine čestice i *jitter*, odnosno varijansa pri emitovanju čestica. Konfiguracija emitera čestica može se izvršiti pomoću korisničkog interfejsa, u delu *Emitter settings*. Upotrebom interfejsa

moguće je: (1) definisati vektor pozicije emitera; (2) definisati vektor orijentacije emitera; (3) definisati brzinu emitovanja čestica; (4) definisati jitter; (5) startovati emiter, pri čemu se zanemaruje trenutna konfiguracija sistema i čestice počinju sa emitovanjem iz definisanog vektora, sa definisanim parametrima; (6) zaustaviti emiter čestica.

Korisnički interfejs daje pregled trenutno aktivnih vektorskih polja, u sekciji *Vector fields* korisničkog interfejsa. On takođe pruža mogućnost definisanja novog vektorskog polja. Pri tome je potrebno: (1) definisati vektor pozicije polja; (2) definisati gravitacionu konstantu; (3) definisati vektor magnetnog momenta; (4) definisati magnetnu konstantu; (5) dodati novo definisano polje u listu aktivnih polja simulacije.

Korisnički interfejs pruža i mogućnost snimanja trenutno aktivne konfiguracije sistema čestica i vektorskih polja u fajl ili učitavanje konfiguracije iz fajla. U ovom simulatoru, sistem čestica, odnosno geometrija implementirana je u trodimenzionalnom prostoru, pa je kroz korisnički interfejs omogućeno kretanje (kamere) po pomenutom prostoru, upotrebom miša. Radi jednostavnosti upotrebe interfejsa, kamera je uvek orijentisana ka korenu euklidskog sistema geometrije, odnosno ka vektoru (0,0,0). Pozicija kamere se u okviru interfejsa modeluje polarnim koordinatama, koje se potom transliraju u euklidske i kontroliše se na sledeći način:

- držanjem levog dugmeta miša i pomeranjem miša horizontalno, postiže se promena ugla θ, odnosno ugla inklinacije;
- držanjem levog dugmeta miša i pomeranjem miša vertikalno, postiže se promena ugla φ, odnosno ugla azimuta;
- okretanjem točkića miša postiže se promena radiusa.

Kombinacijom ove tri akcije, moguće je pozicionirati kameru u svaku od tačaka po sferi trenutnog radiusa sa centrom u vektoru (0,0,0) odnosno sagledati rezultate simulacije iz svih uglova.

Tok simulacije može se kontrolisati pritiskom na dugme *space*, koje rezultuje prekidom, odnosno nastavkom simulacije, u zavisnosti od trenutnog stanja toka simulacije. Korisnički interfejs takođe pruža mogućnost kontrole efekta ostavljanja tragova pri kretanju čestica, što omogućuje bolje praćenje vektora kretanja čestice.

B. REALIZACIJA SIMULATORA

Grafičke sposobnosti aplikacije realizovane su upotrebom HTML5 WebGL API-ja [10]. WebGL API omogućuje pristup grafičkom adapteru računara, odnosno korišćenje OpenGL API-ja za iscrtavanje, u okviru HTML5 dokumenta. Dodatno, u implementaciji ove aplikacije korišćenja je WebGL *wrapper* biblioteka *THREE.js*, koja dodatno apstrakuje operacije nad grafičkim kontekstom, rendererom, kamerom i ostalim tipičnim strukturama jezika OpenGL, i izlaže objektno orijentisani interfejs za manipulaciju tim strukturama.

Za modelovanje sistema čestica, u okviru ove aplikacije koristi se *THREE.ParticleSystem* objekat koji sadrži podatke o geometriji sistema, kao grafičkom modelu čestica. Objekat *THREE.ParticleBasicMaterial* sadrži osnovne podatke o grafičkom modelu čestica kao što su veličina grafičke reprezentacije čestice, providnost i podaci o bojenju čestica. Objekat *THREE.Geometry* sadrži osnovne podatke o geometriji, odnosno pozicije čestica, kao i podatke o bojama svakog od elemenata geometrije. Objekat *THREE.Vector3* sadrži podatke o poziciji jednog elementa geometrije sistema čestica, odnosno poziciji jedne čestice. Za potrebe ovog rada, klasa *THREE.Vector3* je morala biti proširena atributima za opisivanje brzine čestice u datom trenutku, kao i atributima za opisivanje mase i naelektrisanja čestice.

Aplikacija pruža mogućnost lakog konfigurisanja početnog stanja sistema čestica:

- sfera radijusa 50, sa početkom u koordinatnom početku - vektoru (0,0,0);
- čestice pravilno raspoređene u jednoj od ravni euklidskog koordinatnog sistema;
- čestice emitovane u jednoj tački.

Klasa razvijena za potrebe sprovođenja kalkulacije efekata vektorskog polja izlaže interfejs koji se sastoji od metode za prijem poruke, koja nosi *ArrayBuffer* struktuirane podatke o trenutnom stanju sistema čestica, metode u kojoj su implementirani matematički modeli vektorskih polja i metode za vraćanje novog stanja sistema kroz *messaging* sistem *JavaScript*-a. Implementacija ove klase je ujedno i celokupno telo jedne *Web Worker* skripte.

U ovom radu implementirana su dva vektorska polja, gravitaciono polje (G) i magnetno polje dipola (B), čiji su opisi dati u nastavku:

$$G = \begin{Bmatrix} -n * \dfrac{m * M}{|x|^2}, & eksterno \\ \dfrac{-n * m * M * |x|}{r^3}, & interno \end{Bmatrix}$$

Legenda:

G - vektor delovanja gravitacione sile na česticu
x - vektor položaja čestice u odnosu na gravitacioni centar
n - normalizovani vektor položaja čestice
m - masa čestice
M - masa gravitacionog centra
r - radius gravitacionog centra.

$$B = \begin{Bmatrix} \dfrac{pc * \mu_0}{4\pi} * \left(\dfrac{3x(m \cdot x)}{|x|^5} - \dfrac{m}{|x|^3} \right), eksterno \\ \dfrac{pc * \mu_0}{4\pi} * \left(\dfrac{3n(m \cdot x) - m}{|x|^3} + \dfrac{8\pi}{3} m\delta(x) \right), interno \end{Bmatrix}$$

Legenda:

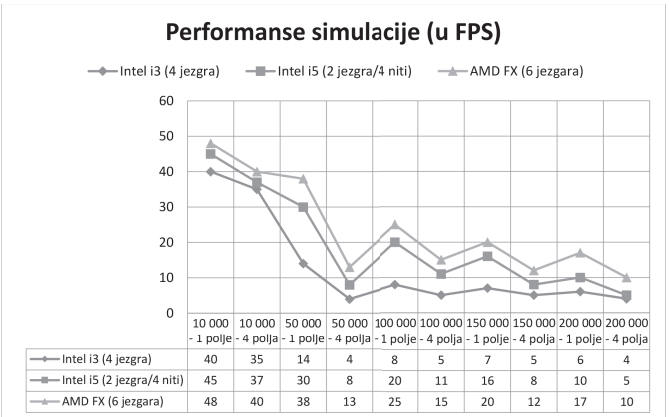
B – vektor delovanja magnetne sile dipola na česticu
pc – naelektrisanje čestice
μ_0 – magnetna konstanta
x – vektor položaja čestice u odnosu na vektor magnetnog momenta
n – normalizovan voktor položaja čestice
m – vektor magnetnog momenta
δ – Dirakova delta funkcija

Web Worker API se koristi pri rešenju problema obrade ogromne količine podataka. Podaci se pri početnoj konfiguraciji dele u posebne skupove podataka i zatim se mehanizmom prosleđivanja poruka jezika *JavaScript* prosleđuju *WebWorker* nitima na obradu, nakon čega se prikupljaju poruke od istih, a podaci iz poruka konsoliduju u jedinstven niz pogodan za prosleđivanje grafičkom renderer-u.

U ovoj aplikaciji se naizmenično odvija računanje efekata vektorskih polja u *Web Worker* nitima i iscrtavanje grafičkog modela sistema čestica na ekran. Pri iscrtavanju čestica na ekran potrebno je kopirati ogroman broj podataka iz operativne memorije računara u memoriju grafičkog adaptera. Ovo se odvija automatski pozivanjem funkcije *render()* WebGL renderera. Međutim, način na koji se ovo izvršava nije optimalan, s obzirom da se radi o velikom obimu podataka. Zarad poboljšanja performansi simulacije, pri realizaciji ove aplikacije primenjena je tehnika izmene za specifične potrebe (eng. *patching*) *THREE.js* biblioteke kako bi bolje odgovarala konkretnim potrebama simulatora. Zbog obfuskacije koda biblioteke *THREE.js* za ove potrebe je primenjivan *debugger*, kako bi se utvrdilo deo koda koji je odgovoran za kopiranje podataka sistema čestica u grafičku memoriju.

4. REZULTATI SIMULACIJE

Simulator je projektovan sa namerom da radi nad velikim brojem čestica i pri tom ispuni minimalne kriterijume simulacije: da zadrži odziv korisničkog interfejsa i da broj slika simulacije u sekundi bude minimalno pet. Aplikacija je testirana sa brojem čestica od 16 do 256K, sa 1-4 vektorskih polja, na procesorima sa 2, 4 i 6 jezgara. Rezultati simulacije prikazani su na slici 6.



	10 000 - 1 polje	10 000 - 4 polja	50 000 - 1 polje	50 000 - 4 polja	100 000 - 1 polje	100 000 - 4 polja	150 000 - 1 polje	150 000 - 4 polja	200 000 - 1 polje	200 000 - 4 polja
Intel i3 (4 jezgra)	40	35	14	4	8	5	7	5	6	4
Intel i5 (2 jezgra/4 niti)	45	37	30	8	20	11	16	8	10	5
AMD FX (6 jezgara)	48	40	38	13	25	15	20	12	17	10

Slika 6: *Performanse simulatora po broju slika u sekundi na tri procesora (Intel i3, Intel i5 i AMD FX) Legenda: FPS - frejmova po sekundi*

Rezultati simulacije su izmereni upotrebom internog merača performansi u okviru pregledača *Google Chrome*, pa kao takvi u izvesnoj meri odstupaju od realnog broja izračunatih slika simulacije. Stoga, za pretpostavku se uzima da dobijeni rezultati oslikavaju performanse sa određenim konstantnim odstupanjem.

Analizom rezultata sa dobijenog grafika se može zaključiti sledeće:

1. Simulator zadovoljava kriterijum tečne simulacije sa brojem čestica i do 200 000 na sva tri test procesora.
2. Skaliranje performansi pri porastu broja čestica i/ili vektorskih polja je bolje od linearnog.

Neka od ograničenja predloženog rešenja su usled implementacije *JavaScript* interpretera pri veb pregledačima, koji nemaju API za legitimno prikupljanje podataka o sistemskom okruženju, te je nemoguće sigurno utvrditi broj procesora na računaru. Jedini način da se podatak o broju procesora pribavi od sistema jeste tehnika procene koja se svodi na postepeno povećanje broja niti i odabir onog broja niti sa optimalnim vremenom obrade. Obzirom da je pomenuta tehnika van teme ovog rada, kao i da je eksperimentalno utvrđeno da broj niti od 16 daje vrlo dobre i ujednačene rezultate simulacije na spektru procesora sa 2 do 6 jezgara, za broj procesora, odnosno niti pri simulaciji se koristi konstantan broj 16.

Zbog sigurnosnih ograničenja kako *Mozilla Firefox*-a, tako i *Google Chrome*-a, konfiguracione fajlove je moguće sačuvati na lokalnom računaru samo za manji broj čestica. Naime, kada se pokuša čuvanje konfiguracionog fajla za veći broj čestica, veb pregledač će prekinuti ovu akciju kako bi zaustavio naizgled malicioznu radnju - preuzimanje velikog fajla sa lokalnog domena kroz veb stranicu. Broj čestica za koji se konfiguracija može snimiti iznosi približno 20 hiljada.

Na kraju, jedno od planiranih poboljšanja performansi izvršavanja simulatora je upotreba *transform feedback* funkcionalnosti OpenGL ES 3.1 standarda kada isti bude implementiran u sledećoj verziji WebGL API-ja [11]. *Transform feedback* funkcionalnost će omogućiti prenos podataka iz memorije grafičkog adaptera u memoriju procesora, čime će biti omogućen scenario sprovođenja kalkulacija uticaja vektorskih polja na samom grafičkom adapteru što će u velikoj meri poboljšati performanse simulatora zbog visokog stepena paralelizma kalkulacija koji grafički adapter pruža.

5. ZAKLJUČAK

U ovom radu predstavljen je vizuelni simulator kretanja čestica pod uticajem vektorskih polja, koji demonstrira mogućnosti novih veb standarda, u kontekstu kompleksnih proračuna. Implementirana su dva vektorska polja: Njutnovo gravitaciono polje i magnetno polje dipola, uz izvesne aproksimacije, kao što su granica unutrašnjosti i spoljašnosti gravitacionog i magnetnog polja, preciznost proračuna i slično.

Korišćenje novih veb tehnologija, kao što je predlog standarda HTML5 i veliki broj API-ja za njihovo korišćenje, pogodno je za razvoj novih složenih aplikacija [12][13], pa čak i aplikacija za naučne proračune koje zahtevaju pristup naprednim hardverskim mogućnostima, kao što su multiprocesorska obrada i rad sa grafičkim adapterom.

Ovaj vizuelni simulator može se koristiti u obrazovne i istraživačke svrhe. Na Elektrotehničkom fakultetu u Beogradu

koristi se u okviru nastave iz Veb dizajna, na drugoj godini studijskog programa Softversko inženjerstvo, za prikazivanje mogućnosti HTML5 standarda. U planu je dalji razvoj simulatora koji podrazumeva proširenje modela čestica i skupa vektorskih polja koja se mogu simulirati, unapređenje preciznosti simulacije rafinisanjem modela, odnosno preciznijom implementacijom matematičkih modela vektorskih polja, i upotrebu grafičkog adaptera (GPU) u svrhu paralelnih proračuna, kada se tehnologije potrebne za ovakve proračune nađu u impelementaciji nekog od popularnih veb pregledača. U planu je i rešavanje problema snimanja konfiguracije u fajl pri velikom broju čestica, ili metodom kompresije podataka pri snimanju ili pružanjem mogućnosti da se fajl otpremi na neki od internet servisa za čuvanje podataka. Još jedna od planiranih mogućnosti je i proširenje korisničkog interfejsa simulatora, kako bi se korisniku omogućilo dinamičko upravljanje modelom čestica i vektorskih polja, kroz minimalnu izmenu modela opisanog u XML-u. Na ovaj način alat bi se mogao koristiti pri testiranju novih matematičko-fizičkih modela prirode.

LITERATURA

[1] Garaizar, P.; Vadillo, M.A.; Lopez-de-Ipina, D., "Benefits and pitfalls of using HTML5 APIs for online experiments and simulations," 9th International Conference on Remote Engineering and Virtual Instrumentation (REV), 2012, Bilbao, Spain, on pages: 1-7, DOI: 10.1109/REV.2012.6293120, Print ISBN: 978-1-4673-2540-0

[2] Hennig, M.; Gaspers, D.; Mertsching, B., "Interactive WebGL-based 3D visualizations for situated mathematics teaching," International Conference on Information Technology Based Higher Education and Training (ITHET), 2013, on pages: 1-6, DOI: 10.1109/ITHET.2013.6671038

[3] Xue, L.; Yangming, Q.; Lei, L., "Visualization of Geomagnetic Environment Based on WebGL," Fifth International Symposium Computational Intelligence and Design (ISCID), 2012, on pages: 274-277, DOI: 10.1109/ISCID.2012.220

[4] Yikang, Y.; Xinxing, L.; Lei, L., "A WebGL - Based Method for Visualization of Space Environment," Fifth International Symposium Computational Intelligence and Design (ISCID), 2012, on pages: 331-334, DOI: 10.1109/ISCID.2012.234

[5] Oliker, A.; Napier, Z.; Deluccia N.; Qualter J.; Sculli F.; Smith B.; Stern C.; Flores R.; Hazen A.; McCarthy J., "Step-based cognitive virtual surgery simulation: an innovative approach to surgical education," Medicine Meets Virtual Reality, Vol. 19, 2012, on pages: 325 - 327, DOI: 10.3233/978-1-61499-022-2-325

[6] The Basics of Web Workers – HTML5Rocks: http://www.html5rocks.com/en/tutorials/workers/basics/, accessed on March 7th, 2014

[7] Using web workers – Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/Web/Guide/Performance/Using_web_workers, accessed on March 7th, 2014

[8] THREE.js: http://threejs.org/docs/, accessed on March 3rd, 2014

[9] Kwangseob, Kim; Sanggoo, Kang; Kiwon, Lee, "Mobile 3D fusion application based on HTML5 WEBGL," IEEE International Geoscience and Remote Sensing Symposium (IGARSS), July 2013, Melbourne, VIC, on pages 1700-1702, Print ISBN: 978-1-4799-1114-1

[10] A vocabulary and associated APIs for HTML and XHTML http://www.w3.org/html/wg/drafts/html/master/, accessed on March 1st, 2014

[11] WebGL Specification - Khronos Group: http://www.khronos.org/webgl/, accessed on March 3rd, 2014

[12] Weigang Zhang; Hao Yuan; Jiangong Wang; Zhonghua Yan, "A WebGL-based method for visualization of intelligent grid," 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), 2011, Weihai, Shandong, on pages 1546-1548, Print ISBN: 978-1-4577-0364-5

[13] Cui, Peng, "The research and design Of 3D Web guide system based On WebGL," The 26th Chinese Control and Decision Conference, June 2014, Changsha, China, on pages: 4052-4054, Print ISBN: 978-1-4799-3707-3

Vladimir Divjak, BSc SE – Univerzitet u Beogradu, Elektrotehnički fakultet
Kontakt: dv060105d@student.etf.rs
Oblast interesovananja: veb dizajn i programiranje

Dražen Drašković, MSc SE – Univerzitet u Beogradu, Elektrotehnički fakultet
Kontakt: drazen.draskovic@etf.bg.ac.rs
Oblast interesovananja: internet programiranje,veštačka inteligencija i inteligentni sistemi, softversko inženjerstvo, testiranje softvera, edukacija

Bojan Furlan, PhD – Univerzitet u Beogradu, Elektrotehnički fakultet
Kontakt: bojan.furlan@etf.bg.ac.rs
Oblast interesovananja: obrada prirodnih jezika, veštačka inteligencija, distribuirano procesiranje

Boško Nikolić, PhD – Univerzitet u Beogradu, Elektrotehnički fakultet
Kontakt: bosko.nikolic@etf.bg.ac.rs
Oblast interesovananja: veštačka inteligencija, internet programiranje, informacioni sistemi, edukacija