Vizualizacija trkov osnovnih delcev v fiziki visokih energij na spletu

Sebastien Strban, Ciril Bohak, Matija Marolt

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
E-pošta: ss4774@student.uni-lj.si, {ciril.bohak, matija.marolt}@fri.uni-lj.si

Web-based visualization of high-energy physics particle collisions

In this paper we present a visualization system for particle collisions in high-energy physics. During the experiments in particle physics a large amount of data is acquired, used for event reconstruction. For better understanding suitable visualization is needed. This poses a big challenge due to large amount of data, especially when we want to achieve real-time interactive functionality. For wider accessibility we decided to implement the visualization in web-based technologies. We developed a visualization system using WebGL, optimized for fast and efficient use with GPUs. We focused on minimization of draw calls and minimization of data transfer to the the GPU. This proofs as an efficient solution confirmed by performance analysis of the developed system.

Povzetek

V članku predstavljamo sistem za vizualizacijo trkov delcev v fiziki visokih energij. Pri eksperimentih v fiziki osnovnih delcev raziskovalci zajamejo ogromno količino podatkov, na podlagi katerih poskušajo rekonstruirati dogajanje. Za boljše razumevanje dogajanja je potrebna ustrezna vizualizacija, kar v primeru velike količine podatkov predstavlja svojevrstni izziv. Še posebej v primeru, ko želimo takšno količino podatkov vizualizirati v realnem času in z visoko stopnjo interaktivnosti. Zaradi želje po čim širši dostopnosti smo se odločili za implementacijo v spletnih tehnologijah. V ta namen smo razvili sistem za vizualizacijo opisanih eksperimentov v spletnem brskalniku z uporabo tehnologije WebGL. Sistem je optimiziran za čim hitrejše in učinkovitejše delovanje na grafičnih procesorskih enotah. Pri tem smo se osredotočili na minimizacijo klicev za izris, in minimizacijo prenosa podatkov na grafično kartico. To se je izkazalo za zelo učinkovito rešitev, kar potrjuje performančna analiza razvitega sistema.

1 Uvod

Vizualizacija predstavlja zelo pomembno vlogo v znanosti. S primerno vizualizacijo omogočimo lažje razumevanje obravnavanega problema, spremljamo obnašanje določenega sistema, ali morda celo opazimo določene lastnosti, ki iz surovih podatkov niso opazne. Predvsem pomembna je vizualizacija na področju fizike, natančneje pri opazovanju dogajanja v kvantni mehaniki, kjer stvari postanejo hitro kontraintuitivne. Primerna vizualizacija omogoči pravilnejše, natančneje in boljše razumevanje delovanja tovrstnih sistemov in izpopolnjevanje znanstvene teorije.

Z napredkom v tehnologiji je vizualizacija postala zelo pomembna pri trkih osnovnih delcev. Tovrstni eksperimenti so nadzorovani z uporabo množice različnih detektorjev prilagojenih za zaznavanje izbranih tipov delcev. Pri trkih delcev z visokimi energijami pride do razpada v še bolj osnovne delce, ki pa jih želimo zaznati in zaradi boljšega razumevanja tudi vizualizirati.

Posamezni dogodek zajema trke okoli 50 parov delcev, ki razpadejo na množico osnovnejših delcev. Tisti, ki nosijo naboj se po trku pod vplivom močnega magnetnega polja odklonijo in zaidejo skozi množico detektorjev, kjer se izmerijo njihove lastnosti, kot so položaj, gibalna količina in energija. Takšen dogodek v eksperimentu je predstavljen z množico točk v prostoru, kjer so bili delci zaznani in njihove izmerjene lastnosti v teh točkah. Za nadaljnje študije je potrebno rekonstruirati poti vseh nastalih delcev, kar predstavlja zelo težak problem. Da pri rekonstrukciji lažje opredelimo pravilnost posamezne poti in morebitne napake (npr. zaradi prisotnosti šuma) potrebujemo primerno vizualizacijo.

Upodabljanje tovrstnih podatkov razdelimo na interaktivne vizualizacije podatkov dogodka (prikazi dogodkov), vizualizacije statističnih podatkov za namene analize podatkov in prostorsko neodvisne vizualizacije podatkov. Tovrstne vizualizacije so predstavljene v [2]. Zajemajo rešitve na namiznih napravah z uporabo vmesnika OpenGL [6] in prenosnih napravah z uporabo vmesnika WegbGL [4]. Primere statističnih 2D in 3D vizualizacij predstavlja spletno orodje JSRoot [1].

Vizualizacija takšnih eksperimentov lahko hitro postane problematična zaradi velikega števila elementov, še posebej kadar želimo zagotoviti visoko stopnjo interaktivnosti. Pri razvoju takšnega sistema želimo razrešiti sledeče izzive:

- intuitivna vizualizacija prikaza podatkov osnovnih delcev in njihovih trajektorij v prostoru,
- realizacija platformno neodvisnega prikaza,
- visoka stopnja odzivnosti vizualizacije na interakcijo in

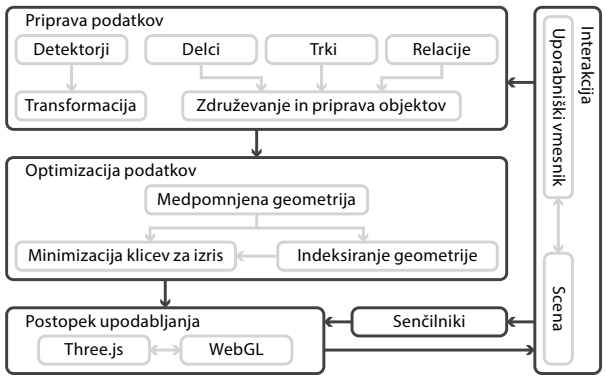
- možnost uporabe na vsakdanjih (tudi mobilnih) sistemih.

Cilj je izdelava sistema, ki omogoča interaktivno vizualizacijo geometrije eksperimenta in vizualizacijo dogodkov interakcije delcev s posameznim detektorjem, kar predstavlja veliko količine delcev znotraj različnih delov geometrije eksperimenta in rekonstruiranih trajektorij delcev. Hkrati želimo za posamezni delec intuitivno vizualizirati izbrane parametre. Z uporabo upodabljanja v brskalniku (z uporabo WebGL-a) želimo realizirati platformno neodvisni prikaz in zagotoviti možnost uporabe na vsakdanjih sistemih.

Takšen sistem predstavimo v naslednjem poglavju 2. V poglavju 3 sistem analiziramo in ovrednotimo. V zadnjem poglavju podamo tudi sklepne ugotovitve in možna izhodišča z nadaljnje delo.

2 Sistem za vizualizacijo trkov

Razviti sistem temelji na upodabljanju v spletnem brskalniku z uporabo tehnologije [4]. Zaradi enostavnosti smo uporabili programsko ogrodje Three.js [3], namenjeno upodabljanju interaktivnih 3D upodobitev v spletnih brskalnikih v realnem času in abstrahira nizkonivojske funkcionalnosti WebGL-a. Shema delovanja sistema je prikazana na sliki 1, v nadaljevanju pa je predstavljen posamezni korak.



Slika 1: Na sliki je prikazana shema razvitega sistema.

2.1 Priprava podatkov

Podatke o geometriji detektorjev in podatke o posameznih dogodkih v eksperimentih, smo pridobili s strani izziva TrackML[1] objavljenega na spletišču Kaggle. Podatki so na voljo v pomoč pri reševanju omenjenega izziva, namenjenega rekonstrukciji poti delcev razpisanega v letu 2018 v CERN-u.

Podatki o geometriji detektorjev so na voljo kot atributi v datoteki tipa CSV, ki jih je potrebno pred uporabo pretvoriti v primerno obliko za izris. Posamezni vnos v datoteki prestavlja centroid modula posameznega detektorja, ki jo moramo s primerno transformacijo, glede na tip detektorja, razširiti v pravokotni ali trapezoidni modul primernih dimenzij in postaviti na ustrezen položaj v prostoru. Končne detektorske module s primerno orientacijo in indeksacijo vozlišč shranimo v medpomnjeno geometrijo in za potrebe osvetlitve poračunamo normale v ogliščih.

Podatki o dogodkih eksperimenta znotraj detektorjev so prav tako zapisani datotekah tipa CSV. Posamezni dogodek zajema informacije o delcih ob trku znotraj detektorja, informacije o vseh interakcijah delcev z detektorji in informacije, ki povezujejo interakcije s posameznimi delci. Informacije povežemo in shranimo v obliki primerni za izris.

2.2 Optimizacija izrisa

Z optimizacijo izrisa zagotovimo potrebi po visoki stopnji interaktivnosti vizualizacije in s tem možnosti uporabe na vsakdanjih sistemih.

2.2.1 Medpomnjena geometrija

Z namenom učinkovite predstavitve geometrije (poligonov, črt ali točk) smo uporabili medpomnjeno geometrijo, ki zahteva nižjenivojsko razumevanje, a omogoča hitrejši izris. Vsi atributi so shranjeni v medpomnilnikih kar skrajša čas prenosa podatkov na grafično kartico (GPE).

2.2.2 Indeksiranje geometrije

Za dodatno pohitritev smo za predstavitev geometrije detektorjev in trajektorij uporabili indeksirano obliko zapisa. S tem zmanjšamo porabo pomnilnika kot tudi skrajšamo čas prenosa na GPE. Z uporabo indeksiranja na GPE omogočimo možnost uporabe predpomnjenja (Post Transform Cache). S tem omogočimo optimizacijo nad prekrivajočimi podatki na GPE.

2.2.3 Minimizacija klicev za izris

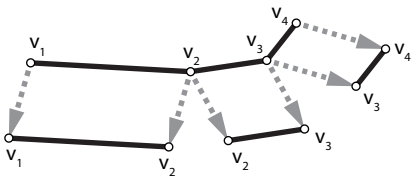
Ker v opisu posameznega dogodka trka nastopa velika količina informacij (npr. 12263 delcev, 120939 trkov in 10566 trajektorij), smo se pri optimizaciji najbolj osredotočili na minimizacijo klicev za izris (angl. draw calls), kar se je izkazalo za pomemben dejavnik in je razvidno iz rezultatov v poglavju 3.

Geometrijo posameznih detektorjev upodobimo z uporabo trikotniških primitivov, ki za skupke treh vozlišč celotne geometrije priredi trikotnike. To nam omogoča, da celotno geometrijo posameznega detektorja upodobimo z enim klicem za izris.

Podatke vseh delcev v začetnih položajih in položaje detekcij trkov upodobimo kot točke. Na ta način upodobimo celotno množico začetnih položajev delcev in vse detekcije trkov vsako z enim klicem za izris.

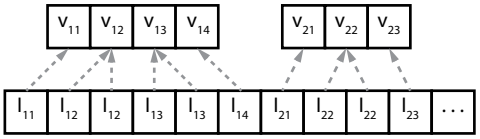
Posamezno trajektorijo razdelimo na segmente daljic (slika 2). Segmente daljic vseh trajektorij shranimo kot skupno geometrijo v medpomnilniku na način, ki nam omogoča ločevanje med trajektorijami, obenem pa povezuje daljice iste trajektorije (slika 3). Ker se vozlišča znotraj trajektorije podvajajo (glej sliko 3), izvedemo že prej omenjeno indeksiranje geometrije, pri čemer obdržimo

[1] https://www.kaggle.com/c/trackml-particle-identification



Slika 2: Na sliki je prikazana zgradba posamezne trajektorije iz posameznih daljic.

relacije daljic v posameznih trajektorijah. S tem pridobimo še dodatno pohitritev. Množico vseh trajektorij upo-



Slika 3: Slika prikazuje indeksiranje geometrije trajektorij.

dobimo kot verigo črt. Na ta način lahko vse trajektorije upodobimo zgolj z enim klicem za izris.

S predstavljenimi pristopi omogočimo izris z minimalnim številom klicev za izris in zagotovimo visoko stopnjo neodvisnega delovanja GPE (analiza v poglavju 3).

2.3 Scena

Za namene intuitivne vizualizacije smo za prikaz celotnega eksperimenta upodobili sceno, ki zajema:

- geometrijo eksperimenta z detektorji,
- začetne položaje delcev,
- zaznane trke delcev z detektorji in
- rekonstruirane trajektorije.

Po sceni navigiramo z uporabo miške in tipkovnice. Objekti so osvetljeni s šestimi lučmi, postavljenimi na ploskve navidezne omejevalne kocke.

2.3.1 Detektorji

Osvetljevanje geometrije detektorja izračunamo z uporabo Lambertovega modela osvetljevanja[2], senčenje pa izvedemo z Gouraudovo tehniko senčenja[3].

Ker se posamezni detektorji v prostoru prekrivajo, smo prosojnost simulirali tako, da jih najprej izrisujemo od zunanjih proti notranjim brez testa globine. S tem omogočimo prosojen pogled skozi detektorje iz središča navzven. Ker smo test globine pri tem onemogočili, so detektorji izrisani v obratnem vrstnem redu. Zato v drugem koraku detektorje izrišemo še obratno (od notranjega proti zunanjemu) s testom globine. S tem omogočimo še prosojen pogled v notranjost, zagotovimo pa pravilen izris geometrije glede na globino.

[2] `https://en.wikipedia.org/wiki/Lambertian_reflectance`

[3] `https://en.wikipedia.org/wiki/Gouraud_shading`

2.3.2 Delci

Senčenje delcev smo izvedli z nadzorom nad velikostjo, barvo in prosojnostjo. Pri tem lahko barvo določimo glede na atribut gibalne količine delca ali pa priredimo enotno barvo vsem delcem. Velikost upodobljenih točk, s katerimi ponazorimo delce izračunamo glede na oddaljenost od kamere. Tako delce, ki so bližje kameri izrišemo večje kot bolj oddaljene. V primeru, da smo v sceni izbrali določen element dogodka za opazovanje, to dodatno poudarimo s spremembo velikosti točke delca, barve in prosojnosti.

2.3.3 Trki

Tudi senčenje zaznanih trkov delcev smo izvedli z nadzorom nad velikostjo, barvo in prosojnostjo. Barvo določimo glede na tip detektorja, kjer je bil delec zaznan, ali pa priredimo enotno barvo. Za boljšo predstavo razporeditve položajev detekcij izrisujemo tiste, ki so bližje kameri z večjimi točkami. Izbiro elementa dogodka v sceni poudarimo s spremembo velikosti točke ki ponazarja trk, barve in prosojnosti.

2.3.4 Trajektorije

Senčenje rekonstruiranih trajektorij smo realizirali z nadzorom nad barvo in prosojnostjo. Barvo določimo glede na atribut gibalne količine v delu trajektorije ali pa priredimo enotno barvo. Zaradi velikega števila trajektorij smo pri senčenju trajektorij barvo in prosojnost dodatno utežili glede na število vseh trajektorij in oddaljenostjo od kamere. Pri izbiri trajektorije le-to poudarimo s spremembo debeline, barve in prosojnosti črt.

Pri spremembi debeline trajektorije smo testirali naslednje rešitve: *metodo WebGL API-ja* `glLineWidth` - kjer z `lineWidth()` določimo širino rasteriziranih linij. Ta pristop ni podprt na vseh platformah in ne bi zadostili cilju platformne neodvisnosti.

enakomerno porazdeljene točke - kjer poskušamo linijo simulirati z enakomerno porazdeljenimi točkami. Pri tem pristopu, glede na velikost točke in dolžino posameznega segmenta trajektorije, izračunamo položaje točk za zapolnitev prostora med dvema vozliščema in s tem ponazorimo linijo. To se je izkazalo za izvedljiv pristop, vendar pa take linije niso zvezne v vseh primerih.

izris s trakom trikotnikov - kjer krajišči vsake linije razdelimo na pare točk in jih pravokotno odmaknemo v smeri normale za želeno debelino (slika 4). Pri tem upoštevamo položaj kamere in dimenzije platna. Dobljene trakove nato izrišemo. To je tudi končna uporabljena rešitev.

Slika 4: Ponazoritev izračuna debeline za posamezni segment trajektorije.

2.4 Uporabniški vmesnik

Za lažjo interakcijo smo implementirali uporabniški vmesnik z uporabo knjižnice dat.GUI[4].

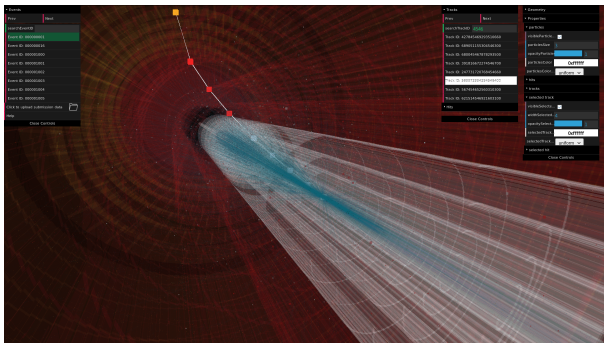
Ob zagonu sistema se v sceni prikaže geometrija detektorja in uporabniški vmesnik. Premikamo se lahko z orbitalno kamero. Uporabniški vmesnik nam ponuja nastavitve posameznih atributov (rezalne ravnine, vidljivost, barva, prosojnost, velikost, tip obarvanja), prikaza detektorjev, delcev, njihovih trkov z detektorji in rekonstruiranih trajektorij. Posamezni dogodek za opazovanje lahko izberemo iz prikazanega seznama. Ob izbiri dogodka se v sceno naloži delce, trke in trajektorije.

Ker je elementov veliko, smo implementirali možnost iskanja preko identifikacijske številke. Za opazovanje lahko izberemo določen trk ali trajektorijo delca, pri čemer se kamera postavi ob izbrani element. V primeru izbire trajektorije se kamera postopoma pomakne med začetno in končno vozlišče trajektorije (odmaknjeno v smeri normale) ter se usmeri v njen centroid. Ob izbiri trka se kamera postopoma pomakne med središče sistema in trk ter se vanj usmeri.

Z namenom pomoči tekmovalcem CERN-ovega izziva TrackML smo dodali tudi možnost nalaganja lastnega dogodka eksperimenta, ki ga lahko primerjamo s katerim izmed izbranih. Na tem mestu smo implementirali funkcionalnost, ki ob izbiri trajektorije priredi seznam najbolj podobnih trajektorij glede na centroid in število segmentov.

3 Analiza in rezultati

Končni izgled razvitega sistema je prikazan na sliki 5, ki prikazuje primer upodobitve enega dogodka trka.



Slika 5: Primer upodobitve fizikalnega dogodka trka delcev z našim sistemom.

Sistem smo testirali na napravi z 2-jedrnim procesorjem *Intel Core i7 4510U* pri taktu 2.00 GHz, 8 GB glavnega pomnilnika in grafično kartico *AMD Radeon R7 M260*. Pri testiranju smo v sceno naložili isti dogodek trka delcev, ob tem pa smo kamero pri vseh testih postavili v enak začetni položaj. Merjenje smo izvedli na vzorcu 10.000 časov upodabljanja, katere smo povprečnici za končni rezultat.

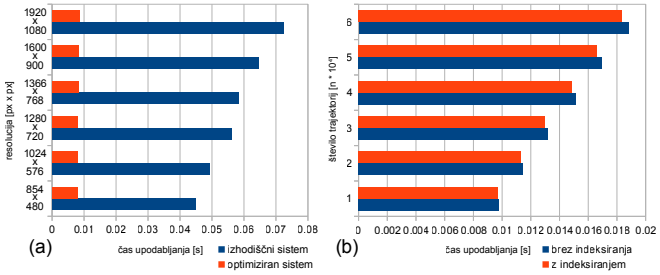
[4] https://github.com/dataarts/dat.gui

3.1 Pohitritev sistema

Testirali smo celotno pohitritev našega sistema v primerjavi z izhodiščnim sistemom (slika 6 (a)). Primerjali smo čas upodabljanja obeh sistemov v odvisnosti od dimenzije platna. Iz testa je razvidno, da skozi spremembo dimenzije naš sistem upodablja v krajšem času kot izhodiščni. To nam omogoča visoko interaktivnost in možnost, da lahko vizualizacijo dodatno nadgradimo (npr. implementiramo obojestransko prosojnost detektorjev za boljšo preglednost).

3.2 Indeksiranje trajektorij

Testirali smo doprinos indeksiranja trajektorij, rezultati so prikazani na sliki 6 (b). Merili smo doprinos uporabe indeksiranja geometrije trajektorij našega sistema v odvisnosti od njihovega števila. Razvidno je, da se doprinos veča s številom prisotnih trajektorij, kar je tudi pričakovano.



Slika 6: (a) pohitritev sistema, (b) čas upodabljanja trajektorij.

4 Zaključek in nadaljnje delo

Poleg omenjenega sistema želimo analizirati in uporabiti še hibridni način upodabljanja z uporabo platforme Med3D[5] [5], ki omogoča odloženo upodabljanje. S tem načinom upodabljanja pričakujemo, da se bo izkazal za še bolj učinkovito in hitrejše predvsem zaradi pospešenega izračuna osvetlitve, kot tudi možnosti združevanja različnih načinov izrisa.

5 Zahvala

Za sodelovanje se zahvaljujemo članom skupine za vizualizacijo pri organizaciji CERN in posameznih fizikalnih eksperimentih (ATLAS, CMS in Alice).

Literatura

[1] Theo Beffart, Maximilian Früh, Christoph Haas, Sachin Rajgopal, Jonas Schwabe, Christoph Wolff, and Marek Szuba. RootJS: Node.js Bindings for ROOT 6. *CoRR*, abs/1704.07887, 2017.

[2] Matthew Bellis, Riccardo Maria Bianchi, Sebastien Binet, Ciril Bohak, Benjamin Couturier, Hadrien Grasland, Oliver Gutsche, Sergey Linev, Alex Martyniuk, Thomas McCauley, Edward Moyse, Alja Mrak Tadel, Mark Neubauer, Jeremi Niedziela, Leo Piilonen, Jim Pivarski, Martin Ritter, Tai Sakuma, Matevz Tadel, Barthélémy von Haller, Ilija Vukotic, and Ben Waugh. Hep software foundation community white paper working group – visualization, 2018.

[3] Brian Danchilla. *Three.js Framework*, pages 173–203. Apress, Berkeley, CA, 2012.

[5] https://github.com/UL-FRI-LGM/Med3D

[4] Dean Jackson and Jeff Gilbert. Webgl specification. Technical report, The Khronos Group Inc., 2015.

[5] Primož Lavrič, Ciril Bohak, and Matija Marolt. Spletno vizualizacijsko ogrodje z možnostjo oddaljenega sodelovanja. In *Zbornik petindvajsete mednarodne Elektrotehniške in računalniške konference ERK 2016, 19. - 21. september 2016, Portorož, Slovenija*, pages 43–46, 2016.

[6] Mark Segal and Kurt Akeley. The OpenGL Graphics System: A Specification (Version 3.1). Technical report, The Khronos Group Inc., March 2009.