Sveučilište u Rijeci
Fakultet informatike i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Tin Švagelj

Metode prikaza volumetrijskih struktura u računalnoj grafici

Završni rad

Mentor: doc. dr. sc., Miran Pobar

Rijeka, 21. rujna 2024.

(Iza naslovne stranice, na ovome mjestu, prilikom uvezivanja umetnite original zadatka završnog rada kojeg ste preuzeli od mentora)

Sažetak

Treba biti 100-300 riječi:

Cilj završnog rada je proći kroz različite metode rasterizacije volumetrijskih podataka u području računalne znanosti.

Rad započinje s uvođenjem različitih podatkovnih struktura i njihove primjene u različitim područjima poput medicine, geoprostornoj analizi, DODAJ SADRŽAJ i računalnim igrama. Zatim ulazi u temu rasterizacije takvih podataka i njihovog prikaza.

Br. riječi: 47

Ključne riječi: računalna grafika; vokseli; rasterizacija

SADRŽAJ

1. Uvod

Cilj računalne grafike je deterministički prikaz trodimenzionalnog (3D) sadržaja na zaslonu računala. Kako bi se to postiglo, primjenjuju različiti algoritmi na strukturama podataka koje su zavisne o području primjene i ciljevima softvera. Tradicionalni način prikaza 3D sadržaja je predstavljanje takvog sadržaja korištenjem jednostavnih matematičkih tijela (engl. *primitives*) poput trokuta, linearna transformacija njihovog prostora, projekcija na plohu koja je uzorkovana (engl. *sampled*) pravilnom rešetkom (engl. *grid*) piksela, te konačno prikazana na ekranu.

Nakon inicijalnog razvoja grafičkih kartica (engl. *graphics processing unit*, GPU) sa fiksiranim dijelovima procesa prikaza (engl. *fixed function pipeline*, FFP), kasnije je razvijena i sljedeća generacija GPUa sa programabilnim procesom prikaza (engl. *programmable pipeline*). Takve grafičke kartice dozvoljavaju uporabu različitih programa (engl. *shader*) za prikaz i izračun. Shaderi su konstantnim napretkom postajali sve fleksibilniji te se danas primjenjuju u mnoge svrhe koje nisu usko vezane za grafiku, ali zahtjevaju visoku razinu konkurentnih izračuna, poput:
- simulacije,
- analize podataka,
- neuralnih mreža,
- obrade multimedijskih podataka, te brojne druge.

Prikaz volumetrijskih struktura je jedna od takvih namjena za koje FFP često nije prikladan jer se oslanja na specijalizirane algoritme koji često ne rade s trokutima nego simuliraju zrake svijetlosti.
U svrhu unaprijeđenja performansi ovakvog pristupa su proizvođači grafičkih kartica od 2020. godine počeli uključivati specijaliziran hardver kako bi se postigla hardverska akceleracija prilikom rada s volumetrijskim podacima [1].

Cilj ovog rada je prikazati načela rada s volumetrijskim podacima, kao i njihovih prednosti u različitim područjima primjene.

Razvoj grafike koja utilizira FFP je popratio razvoj i popularnost sukladnih formata za pohranu modela koji opisuju isključivo površinsku geometriju predmeta (engl. *mesh*). Za svrhe jednostavnog prikaza je taj oblik pohrane idealan, no nedostatan je za obradu trodimenzionalnih podataka. U mnogim primjenama takvo ograničenje dovodi koncesija koje negativno utječu na performanse aplikacija ili njihove sposobnosti. Također je nedostatan i za primjene u simulacijama jer zahtjeva nadomještaj nedostatnih podataka o volumenu različitim aproksimacijama za koje traže sporu prethodnu obradu (engl. *preprocessing*).
Drugi cilj ovog rada je osvrnuti se na takve formate za pohranu trodimenzionalnih podataka i ukazati na uvedene neučinkovitosti zahtjevane njihovim nedostacima.

- https://www.sciencedirect.com/topics/computer-science/volumetric-dataset
- https://developer.nvidia.com/gpugems/gpugems/part-vi-beyond-triangles/chapter-39-volume-rendering-techniques
- https://web.cse.ohio-state.edu/~shen.94/788/Site/Reading_files/Leovy88.pdf

1.1. Definicija volumetrijskih podataka

Volumentrijski podaci se mogu predstaviti na mnogo različih načina, no u osnovi se radi o skupu vrijednosti koje su pridružene koordinatama u nekom prostoru.

Memorijska ograničenja računala nalažu da su prostori u memoriji učinkovito **konačni**. Također su svi produktni prostori koje možmo stvoriti i **prebrojivi** neovisno o načinu na koji ih pohranjujemo u memoriji. To se odnosi i na decimalne tipove podataka jer za svaki postoji neki korak ε između uzastopnih vrijednosti koje možemo pohraniti.
Konačno, svaka topologija koja se rasterizira se u nekom dijelu rasterizacijskog pipelinea **poprima prekide**: u krajnjem slučaju prilikom prikaza na zaslonu računala, no obično i ranije prilikom obrade.
U većini primjena je cilj izbjeći vidljivost tih kvaliteta prilikom prikaza, no ta činjenica opušta mnogo problema (engl. *problem relaxation*) prilikom rada jer dozvoljava međukoracima algoritama za obradu da aproksimiraju rezultate s ciljem bržeg izvođenja.

Neko tijelo u 3D prostoru predstavljamo skupom uređenih trojki, tj. vektora koji zadovoljavaju neki:

$$P : A^3 \to \{\top, \bot\}$$
$$V :\equiv \{(x, y, z) \in A^3 \mid P(x, y, z)\}$$

Izraz 1: volumen tijela

gdje je (x, y, z) uređena trojka koordinata, a P neki sud koji određuje uključuje li razmatrani volumen tu trojku/vektor. Tip suda P zavisan o ulaznim vrijednostima, tj. tip vrijednosti (A) nije bitan za izraz te je zamjenjiv s $\mathbb{R}$ ili nekim drugim tipom poput `f32` ili `u64` - ovisno o namjeni.

U slučaju volumetrijskih podataka volumenu želimo pridružiti neku vrijednost pa ako vrijedi da

$$\exists f : (A^3 \subseteq V) \to \mathcal{C}$$
$$\Downarrow$$
$$\exists (D : A^3 \times \mathcal{C}).D :\equiv \{(x, y, z, f(x, y, z)) \mid (x, y, z) \in V\} \equiv \{(x, y, z, c)\}$$

Izraz 2: skup volumetrijskih podataka

gdje je c neka vrijednost tipa $\mathcal{C}$ koju pridružujemo koordinatama, a f preslikavanje kojim ju određujemo za sve koordinate prostora V.

Dakle, bilo koja funkcija čija je kodomena skup *uređenih trojki elemenata tipa* A^3 je prikladna za predstavljanje volumena (*oblika* volumetrijskih podataka), te ako postoji neko preslikavanje tog volumena na neku željenu informaciju (npr. gustoču ili boju), imamo potpuno definirane volumetrijske podatke.

1.2. Računalna primjena

Proces pretvorbe zapisa s uvjetima (sudom), odnosno određivanja vrijednosti funkcije u određenim točkama, zove se **diskretizacija** funckije. Rezolucija diskretiziranih podataka ovisi o načinu na koji smo preslikali i pohranili funkciju u računalu s tipovima podataka ograničene veličine, te koji tip brojeva je korišten za pogranu (npr. `u16`/`u32`).

Diskretizacija je željena jer značajno umanjuje potreban rad na grafičkim karticama, kao i u nekim slučajevima daljnju obradu, no kompozicija zapisa s uvjetima je manje algoritamski zahtjevna zbog čega se taj pristup češće primjenjuje za generiranje volumetrijskih podataka kao i kod generativnih metoda prikaza poput koračanja zrakama (engl. *ray marching*).

Stvaranje volumetrijskih podataka iz stvarnog prostora uporabom perifernih uređaja poput laserskih skenera se zove **uzorkovanje**. Isti naziv se ponekad koristi i za diskretizaciju, no u ovom radu će se koristiti preteći u svrhu jasnoće, iako imaju preklapanja u značenju.

Diskretne podatke je moguće predstaviti kao neuređeni niz točaka (tj. kao skup volumetrijskih podataka (2)), no taj oblik pohrane ima najgoru složenost za pronalazak vrijednosti $\Theta\left(n_x n_y n_z\right)$.
Kako bismo odabrali optimalan način strukturiranja podataka, potrebo je znati odgovor na sljedeća pitanja [2]:

1. S kojim tipovima podataka baratamo?
2. Za kakve su operacije korišteni?
3. Trebamo li ih kako organizirati ili prostor u kojeg ih ugrađujemo (engl. *embedding space*)?
4. Jesu li statični ili dinamični (tj. može li broj točaka u volumenu porasti tokom rada aplikacije)?
5. Možemo li pretpostaviti da je volumen podataka dovoljno malen da ga u potpunosti smjestimo u radnu memoriju ili trebamo poduzeti mjere potrebne za pristup uređajima za trajnu pohrani?

Zbog različitih odgovora na ta pitanja ne postoji rješenje koje funkcionira približno dobro za sve namjene. U poglavlju o strukturama su podrobnije razjašnjene razlike između različitih načina pohrane volumetrijskih podataka.

1.3. Primjene volumetrijskih podataka

- Medicina - https://www.sciencedirect.com/topics/computer-science/volumetric-data
  - Rendgenska tomografija
  - Elektronski mikroskopi
    - (engl. *Transmission Electron Microscopy*, TEM) i (engl. *Scanning Transmission Electron Microscopy*, STEM)
  - DICOM format
- Geoprostorna analiza
- Prozivodnja (doi: 10.1117/12.561181)
  - Brzina prototipiranja
- Simulacije
- Računalne igre

1.4. Komercijalni primjeri

- C4, by Terathon http://www.terathon.com/
  - discontinued
- https://voxelfarm.com/index.html
- https://gareth.pw/web/picavoxel/
- Euclideon
  - Koristi point cloud mislim

- claim: “Ne koristi GPU pipeline”

2. Alternativne metode prikaza

Konkretno u računalnoj znanosti su česte primjene:
- diskretnih podataka (unaprijed određenih vrijednosti) u obliku
  - nizova točaka ili "oblaka točaka" (engl. *point could*), ili
  - polja točaka (engl. *voxel grid*)
- jednadžbi pohranjenih u shaderima koje se koriste u konjunkciji s algoritmima koračanja po zrakama svijetlosti (engl. *ray marching*).

Diskretni podaci imaju jednostavniju implementaciju i manju algoritamsku složenost, no zauzimaju značajno više prostora u memoriji i na uređajima za trajnu pohranu. Za ray marching algoritme vrijedi obratno pa se ponajviše koriste za jednostavnije volumene i primjene gdje su neizbježni.

Definicija za volumen tijela (1) pruža korisno ograničenje jer pokazuje da možemo doći do volumetrijskih podataka i na druge načine.
Primjer toga je česta primjena složenijih funkcija koje proceduralno generiraju nizove točaka za prikaz. Ovaj oblik uporabe je idealan za računalne igrice koje se ne koriste stvarnim podacima jer se oslanja na dobre karakteristike diskretnog oblika, a izbjegava nedostatak velikih prostornih zahtjeva na uređajima za trajnu pohranu.

2.1. Ray tracing

Svijetlost se može odbijati i ne završiti u kameri [3]:

$$L_0(p, \omega_0) = L_e(p, \omega_0) + \int_{S^2} f(p, \omega_0, \omega_i) L_i(p, \omega_i) |\cos(\theta_i)| \, d\omega_i$$

Pojednostavljenje… promatramo zrake koje završe u kameri tako što simuliramo zrake *iz* kamere i badabim, badapam zbrojimo konačan broj odskakanja, bla bla, itd.

Denoizing može biti savršen za voksele: Per-voxel Lighting via Path Tracing, Voxel Engine Devlog #19 https://www.youtube.com/watch?v=VPetAcm1heI

2.2. Ray marching

Koristimo neki jednostavan primitiv za testiranje kolizije zrake sa tijelom, obično kugla ili kocka. Koračamo za veličinu najvećeg takvog tijela unaprijed.

- Dozvoljava fora efekte poput metaballs.
- Brže od raytraceanja (u nekim slučajevima, npr. vokseli), no ograničenije jer zahtjeva da su sva prikazana tijela izražena formulama i shader onda mora rješavati sustave jednadžbi
  - Postoji negdje neki algoritam s kojim se može aproksimirati arbitrarni mesh ali daje dosta složene formule u mnogim slučajevima (koje su onda teške za predstaviti u kodu i spore za izračun).
    - Vidio sam ga prije 4/5 godina, treba to iskopati sada…
  - Je li doista brže za voksele (e.g. Nvidia SVO) ili samo za loše slučajeve? Slabo se koristi.

3. Strukture za pohranu volumetrijskih podataka

Tablica 1: usporedba karakteristika struktura diskretnih volumetrijskih podataka

Ime strukture	Čitanje	Pisanje	Prednost
Jednodimenzionalna uređenja	$\Omega(1)$	1	
3D polje	1	1	
Range Tree	1	1	
Priority Search Tree	1	1	
Oktalno stablo	1	1	
Raštrkana stabla voksela	1	1	
DAG	1	1	
K-d Stablo	1	1	
Bucket Methods	1	1	
PK-Stablo	1	1	

3.1. Jednodimenzionalna uređenja

3.2. 3D polje

```
const CHUNK_SIZE: usize = 32;
type EntryID = u16;

struct Chunk<T> {
  data: [[[EntryID; CHUNK_SIZE]; CHUNK_SIZE]; CHUNK_SIZE],
  values: Vec<T>
}
```

Kôd 1: struktura 3D polja

- Jednostavna i najčešća implementacija za real time render
- Postoji relativno puno primjera, alogritama, …

3.3. Range Tree

3.4. Priority Search Tree

3.5. Oktalno stablo

Oktalna stabla (engl. *octree*) su jedna od vrsta stabla koja imaju iznimno čestu uporabu u 3D grafici za ubrzavanje prikaza dijeljenjem prostora. Strukturirana podjela prostora dozvoljava značajna ubrzanja u ray tracing algoritmima jer zrake svijetlosti mogu preskočiti velike korake.

Koriste se i u simulaciji fizike jer olakšavaju brzu segmentaciju prostora čime se postiže brzo isključivanje dalekih tijela iz izračuna kolizija.

```
enum Octree<T> {
  Leaf(T),
  Node {
    children: [Box<Octree>; 8],
  }
}
```

Kôd 2: struktura oktalnog stabla s pokazivačima

```
enum OctreeNode<T, const DEPTH: usize> {
  Leaf(T),
  Node {
    children: [Octree<T, {DEPTH - 1}>; 8],
  }
}
```

Kôd 3: struktura oktalnog stabla koje je sekvencijalno u memoriji

3.5.1. Raštrkana stabla voksela

Raštrkana stabla voksela (engl. *sparse voxel octree*, SVO) su vrsta stablastih struktura koja pohranjuje susjedne čvorove u nelinearnim segmentima memorije te zbog toga dozvoljava "prazne" čvorove.

```
enum Octree<T> {
  Leaf(Option<T>),
  Node {
    children: [Box<Octree>; 8],
  }
}
```

Prednost ovakvih struktura je što prazni dijelovi strukture ne zauzimaju prostor u memoriji, te ih nije potrebno kopirati u memoriju grafičkih kartica prilikom prikaza.

No iako rješavaju problem velike potrošnje memorije, čine izmjene podataka iznimno sporima te se zbog toga primjenjuju skoro isključivo za podatke koji se ne mijenjaju tokom provođenja programa. Izvor loših performansi izmjena su potreba za premještanjem (kopiranjem) postojećih podataka kako bi se postigla njihova bolja lokalnost u međuspremniku (engl. *cache locality*) na grafičkoj kartici.

- Komplicirana implementacija
  - Postoji već gotov shader kod za ovo u par shading jezika negdje
- https://research.nvidia.com/sites/default/files/pubs/2010-02_Efficient-Sparse-Voxel/laine2010tr1_paper.pdf

3.5.2. DAG

- Varijanta SVOa, navesti razlike.
- Grozne karakteristike izmjena (po defaultu)
  - https://github.com/Phyronnaz/HashDAG

3.6. K-d Stablo

3.7. Bucket Methods

3.8. PK-Stablo

3.9. Point-cloud data?

Spremljeno u Octreeu zapravo?

Laserski skeneri ju generiraju, česta primjena u geoprostornoj analizi.
- Dronovi ju koriste za navigaciju.

4. Prijevremen prikaz

- Primjene: medicina, statične scene i slike

4.1. Ray casting

- Skoro je i real time sada, mislim da je noisy, ima light propagation delay, …

4.2. Splatting

Transparentni slojevi za brz prolazak kroz slojeve MRI slike.

4.3. Shear warp

5. Prikaz u realnom vremenu

- https://0fps.net/2012/06/30/meshing-in-a-minecraft-game/
- https://www.reddit.com/r/VoxelGameDev/comments/j89l6j/texturing_with_greedy_meshing/
  - Vokseli sa teksturama nisu vokseli, no ovo je temelj za povezivanje drugih podataka (boje u praktičnom dijelu, occlusion, normale, …)

5.1. GPU streaming

Opisati kako LOD funkcionira za svijet sastavljen od chunkova.

5.2. Metode optimizacije

- https://acko.net/blog/teardown-frame-teardown/

6. Animacije

- Ona metoda gdje se generira AABB za segmente koji međusobno ne colideaju i koristi skeleton
- Metoda sa deformacijom voksela
  - Nije “pravi” voxel renderer
- Metoda gdje se u compute shaderu samplea animirani triangle mesh svaki frame
  - Izgleda meh i relativno je zahtjevno
- Metoda gdje je definirana funkcija koja mapira deltatime na konfiguraciju voksela
- Opisati kako je DAG neprikladan za animiranje - ili spor ili jako potrošan glede memorije

7. Usporedba s poligonima

Zbog prethodnih ograničenja u računalnoj memoriji, uporaba voksela je značajno slabije istražena metoda pohrane modela i prikaza. Postojeći harver je optimiran za prikaz tradicionalnih

7.1. Problemi

- Kako zaobliti svijet
  - Je li vrijedno spomena, vrlo specifično za računalne igre…
  - Postoji negdje efekt sa shaderima; ne rješava probleme topologije
  - https://www.youtube.com/watch?v=bJr4QlDxEME

7.1.1. Izgled

- Sampliranje uvijek narušava kvalitetu modela, ili zauzima previše memorije
  - Artisti su naviknuti raditi s trokutima
  - Nije bitno za proceduralan sadržaj
    - Marching cubes i djeca
- Vrlo teško modelirati nepravilne oblike

7.2. Interaktivnost

- “Svijet nije maketa od papira nego ima volumen”.
  - Postoje metode lažiranja nekih vrsta interakcija.
    - Dodavanje svake zahtjeva zaseban trud dok je voxel engine uniforman (veliki upfront cost, lakše dodavanje sadržaja)

7.3. Košta

- Kako bi vokseli bili praktični u real time aplikacijama potrebno je jako puno rada u optimizaciji njihove strukture u GPU memoriji, rendering kodu, LOD sustavu, …
- Nvidia paper je dobar početak ali ne zadovoljava mnogo zahtjeva modernih računalnih igara
  - Mislim da nema transparentnost?

8. Zaključak

Literatura

[1] R. Smith, „Quadro No More? NVIDIA Announces Ampere-based RTX A6000 & A40 Video Cards For Pro Visualization“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.anandtech.com/show/16137/nvidia-announces-ampere-rtx-a6000-a40-cards-for-pro-viz

[2] H. Samet, *Foundations of multidimensional and metric data structures*. u The Morgan Kaufmann Series in Computer Graphics. Oxford, Engleska: Morgan Kaufmann, 2006.

[3] M. Pharr i W. Jakob, *Physically Based Rendering, četvrto izdanje*. MIT Press.

Popis priloga