SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Crtanje različitih modela pomoću jednog poziva

Karlo Miličević
Voditelj:

Zagreb, siječanj 2022.

SADRŽAJ

1. Uvod

Crtanje pomoću sklopovlja specijaliziranog za grafiku je dosta brzo. Postoje razni faktori koji ograničavaju količinu geometrije koju možemo nacrtati u nekom vremenskom razdoblju.

Par ograničenja koja se pojavljuju su količina informacija koje šaljemo od glavnog procesora do sklopovlja za crtanje, brzina obrade tih informacija na sklopovlju za crtanje i brzina kojom glavni procesor može proizvoditi informacije koje šalje.

Brzina obrade se može podijeliti na obradu informacija točaka trokuta, obradu informacija elemenata slike, obradu informacija instanci objekata koji su stvoreni od mnoštva trokuta i međuobrade informacija za instance.

Ovaj rad se dotiče toga kako glavni procesor šalje informacije na sklopovlje za crtanje.

1.1. Crtanje

Grafičko sklopovlje je napravljeno da najbolje radi s trokutima. Možemo informacije o trokutima zapisati i prenjeti na puno načina. Najjednostavniji način je zapisati informacije o svim točkama sljedno u memoriji i onda tu memoriju prenjeti.

U ovom slučaju je potrebno postaviti sklopovlje da interpretira po tri točke kao jedan trokut. Za realistične modele koji aproksimiraju stvarne objekte će se pojaviti puno točaka koje su dijeljene između trokuta. Ako pretpostavimo da model nije šupalj, imat ćemo barem trostruko zapisane informacije o pozicijama točaka. Ostale informacije - poput svjetline, pozicije teksture i vektoru normale će isto biti trostruke za granični slučaj s beskonačno precizno uzorkovane modele.

1.2. Indeksiranje

Rješavanje tog problema višestrukih točaka nam rješava indeksiranje. Indeksiranje je tehnika u kojoj pošaljemo niz točaka, ali ne crtamo direktno taj niz točaka, nego uz taj

niz šaljemo i dodatan niz koji opisuje trokute koji su napravljeni od točaka iz poslanog niza točaka.

Možemo, ali i ne moramo točke slati sa svim svojstvima isprepleteno, nego možemo imati odvojen niz za svako od svojstava točaka. A za izračunati informacije o točki koristimo informacije indeksa za svojstva.

Primjer **??** pokazuje odvojene nizove za svako od svojstava točaka.

```
pozicija  Pozicije [];
boja      Boje     [];
svjetlina Svjetline[];
struct indeksi
{
    indeks P;
    indeks B;
    indeks S;
};
indeksi Tocke[];
```

Isječak 1.1: Stuktura informacija točaka

Računanje informacija o pojedinoj točki bismo tada činili ovako kao u primjeru **??**.

```
pozicija  Pozicija  = Svjetline[Tocke[i].S];
boja      Boja      = Svjetline[Tocke[i].S];
svjetlina Svjetlina = Svjetline[Tocke[i].S];
```

Isječak 1.2: Dohvat informacija o točkama

1.3. Instanciranje

Instanciranje je još jedna od tehnika. Instanciranje pokušava riješiti problem ponavljanja skupine trokuta.

Na primjer, ako simuliramo promet u gradu i imamo model nekog automobila, morali bismo imati zasebno crtanje modela za svako od ponavljanja tog automobila. Nadalje, automobil ima četiri kotača, te bismo morali četiri kotača opet crtati zasebno za svaki od autautomobila.

Pomoću instanciranja broj crtanja možemo smanjiti tako da stvorimo niz instanci gdje nam jednu instancu opisuju neki podaci poput pozicije i orijentacije.

Za gornji primjer bi to bio niz instanci automobila i niz instanci kotača. Crtanje automobila u gradu bi se svodilo na crtanje instanci kotača te crtanje instanci ostatka automobila.

```
for automobil in automobili:
    crtaj(automobil)
    for 1 .. 4:
        crtaj(automobil.kotaci[i])
```

Isječak 1.3: Crtanje bez instanciranja

```
izracunaj sve automobile i kotace.
crtaj(svi_automobili)
crtaj(svi_kotaci)
```

Isječak 1.4: Crtanje s instanciranjem

Instanciranje nema direktnog utjecaja na indeksiranje i tehnika indeksiranja se može koristiti zajedno s tehnikom instanciranja.

2. Indirektno crtanje

Tehnika indirektnog crtanja, za razliku od prethodno opisanih tehnika, ne smanjuje količinu podataka koje šaljemo na sklopovlje za crtanje.

Što nam idirektno crtanje omogućuje je smanjenje količine posla koji upravljački program koji komunicira sa sklopovljem za crtanje treba obaviti.

Upravljački program mora za svaki od poziva crtanja, slanja, dohvata informacija, promjena sjenčara i slično provjeriti jesu li informacije koje mu pružamo ispravne i prevesti ih u instrukcije koje može predati sklopovlju. Upravljački program mora moći podržati sve što specifikacija traži i mora biti dovoljno generičan da može podržati puno različitih načina ponašanja programa koji ga koriste te ga je iz tog razloga teško optimirati.

Smanjenjem broja zahtjeva koje šaljemo upravljačkom programu ima učinak da se smanjuje opterećenje glavnog procesora.

2.0.1. Kako indirektno crtanje radi

Za indirektno crtanje trebamo stvoriti niz poziva. Grafičko sklopovlje će taj niz poziva koristiti kao da smo izvršili te pozive svaki za sebe.

```
struct draw_indirect_command
{
    u32 VertexCount;
    u32 InstanceCount;
    u32 VertexFirst;
    u32 InstanceFirst;
};
struct draw_indexed_indirect_command
{
    u32 IndexCount;
```

```
    u32 InstanceCount;
    u32 IndexFirst;
    u32 VertexOffset;
    u32 InstanceFirst;
};
```

Isječak 2.1: Struktura elemenata niza naredbi za indirektno crtanje

Konkretno, grad s automobilima bismo mogli crtati tako što pošaljemo informacije o modelu automobila i modelu kotača, informacije o instancama automobila i instancama kotača. Za sada izgleda identično kao i za primjer instanciranja i indeksiranja, no trebamo poslati još i niz od dvije naredbe za indirektno crtanje - jedna za kotače i jedna za automobile. Crtanje tada izvodimo pomoću samo jednog poziva upravljačkom programu.

Lako je vidjeti da za proširiti ovo crtanje na još, recimo, semafore, autobuse, autousne stanice, zgrade, i slično, nije teško - poslali bismo modele za svaki od njih na grafičko sklopovlje, izračunali podatke o instancama i njih poslali te poslali niz naredbi za indirektno crtanje. Još uvijek bi to bio samo jedan poziv upravljačkom programu.

2.1. Prebacivanje dijela računanja na sklopovlju za crtanje

Računanje orijentacija i pozicija instanci u odnosu na gledište možemo raditi na sklopovlju za crtanje. To sklopovlje je pogodno za takvu vrstu zadatka - vektorske i matrične operacije. Osim toga, nakon računanja možemo provjeriti upada li rezultat u vidno polje kamere i ako ne upada, možemo ga odbaciti.

To provjeravanje i odbacivanje je moguće učiniti i na glavnom procesoru prilikom stvaranja niza instanci, ali veće brzine su moguće kada taj dio posla prebacimo na sklopovlje za crtanje.

Slanje i crtanje u ovom slučaju se blago mijenja. Kao i prije, šaljemo podatke o modelima, teksture i slično. Nakon toga, šaljemo informacije o instancama i pozivima koje su nerelativne na gledište. Pozivamo opći sjenčar za računanje koji pretvara informacije u one relativne na gledište te odbacije instance i pozive koji ne utječu na krajnju rezultat. Osim toga, sjenčar prati i broj poziva koje je stvorio i broj instanci za svaki od poziva.

Nakon izvršavanja tog sjenčara se poziva indirektno crtanje. Sklopovlje za crtanje

će informacije iz nizova naredbi za indirektno crtanje koje je prethodni sjenčar stvorio pozvati crtanja.

2.2. Ograničenja

Problem koji nas sprječava da cijelu scenu crtamo pomoću jednog poziva je što nam ovaj pristup ne dozvoljava korištenje različitih sjenčara. Svaka od naredbi za indirektno crtanje opisuje crtanje u istom sjenčaru.

To nije problem kada imamo mali broj sjenčara koji se koriste, ali to nije uvijek slučaj. Moderne igre imaju veliku količinu sjenčara, i za ovakvo crtanje bi bilo potrebno imati po barem jedan poziv za svaki od sjenčara.

Postoje NVidia proširenja koja omogućuju i određivanja sjenčara unutar elemenata niza naredbi za indirektno crtanje. U tom slučaju je problem riješen.

Ako nemamo ta proširenja, možemo i spojiti više sjenčara u jedan i na temelju podataka instance odabirati način ponašanja. Ovaj pristup radi, ali je nezgodan za koristiti i opisivati drastično različite sjenčare.

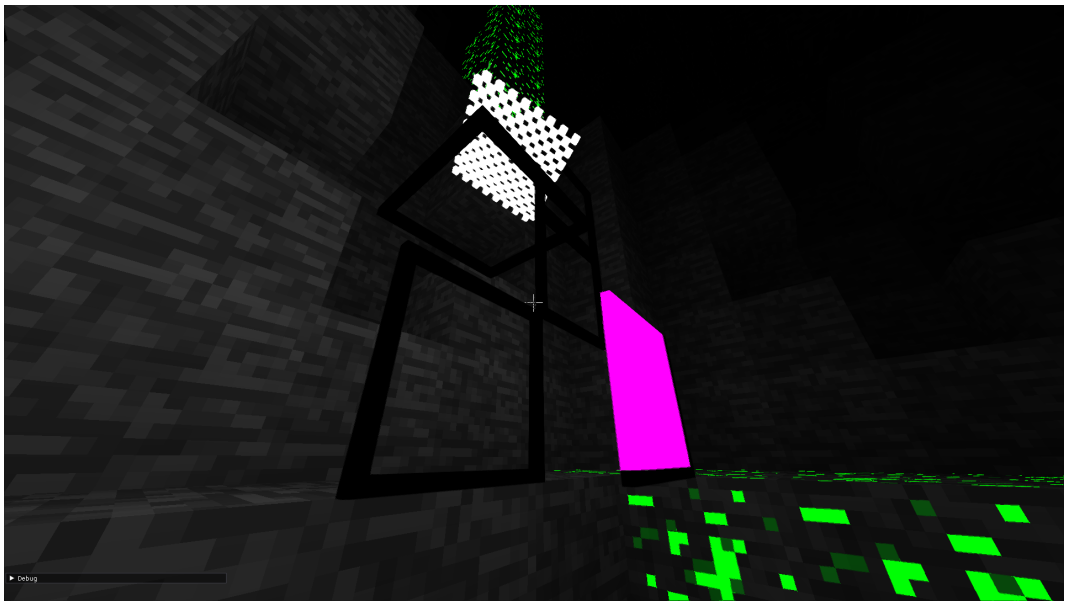
Srećom, količina poziva upravljačkom programu u puno slučajeva je zanemariva. Pogotovo je to slučaj kada koristimo tehniku instanciranja.

3. Praktični rad

Za praktični rad je napravljen u prethodnom poglavlju opisan postupak. Korištena je biblioteka za grafiku Vulkan 1.2 i programski jezik C++.

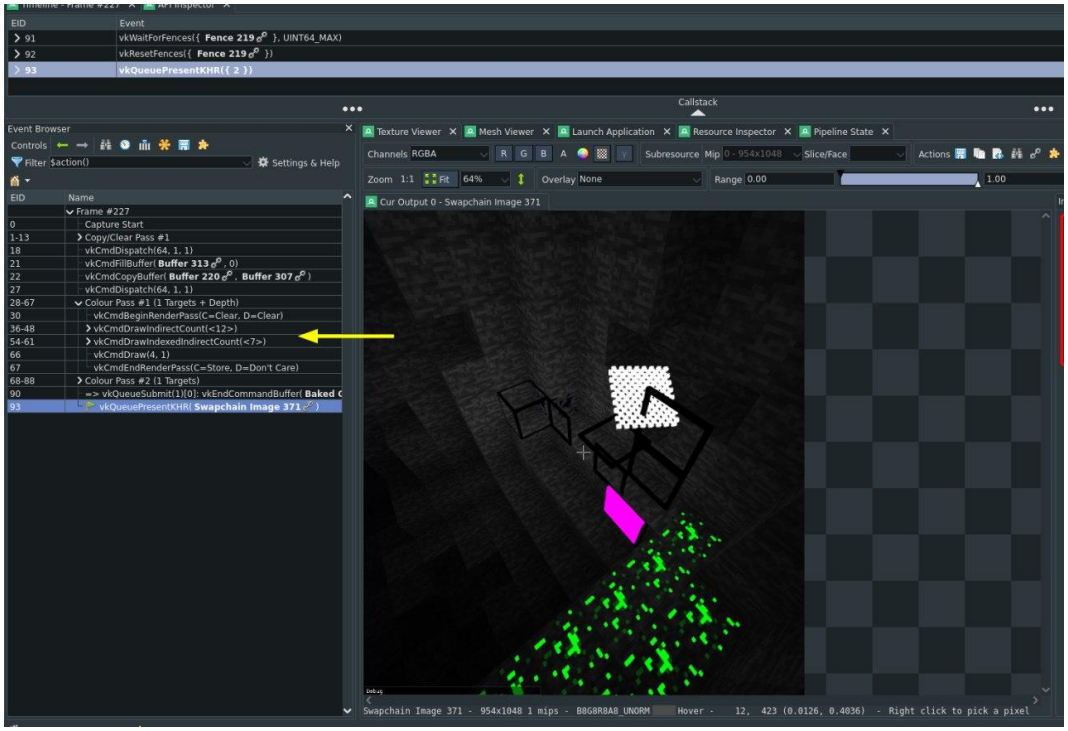
Program je moguće pokrenuti na operacijskom sustavu Arch Linux. U samom programu nema značajnije interakcije osim upravljanja kamere mišem i tipkovnicom pomoću tipki "A", "D", "S", "W", "Shift" i "Space".

U programu prikazano crtanje opisanim postupkom. Uz izvršni kod je priložena i datoteka s isječcima koji se tiču opisanog postupka.



Slika 3.1: Rezultat

Na slici 3.1 prikazan rad programa.



Slika 3.2: Rezultat

Na slici 3.2 prikazana analiza rada programa pomoću programa renderdoc.

4. Zaključak

Prikazivati prostor je moguće izvesti na razne načine. Svaki pristup ima svoje pozitivne strane. Jedan od načina prikaza prostora je pomoću volumnih elemenata (engl. voxel). Ta metoda nije raširena koliko i prikaz modela pomoću trokuta. Ideja rada je bila razmotriti prostor opisan volumnim elementima, opisati prikaz istoga i vidjeti implementaciju na sklopovlju.

Pobliže je opisana implementacija kockastog prikaza prostora opisanog volumnim elemntima. Opisane su i implementirane optimizacije kako bi izvedba na sklopovlju bila interaktivna. Opisani su doprinosi optimizacija. Optimizacije uključuju smanjenje memorijskog prostora potrebnog za opisati model regija, smanjenje broja lica koja prikazujemo i ubrzanje izgradnje modela. Opisan je jednostavan model svjetlosti i ubrzanje računanja istog. Objavljeni su i javno dostupni opisani algoritmi u programskom jeziku C++.

Dobiveni rezultati rada bi se mogli još unaprijediti. Implementacija bi se mogla još ubrzati tako da spajamo susjedna lica koja dijele stranicu u ravnini u jedno veće lice i time smanjiti broj lica koje je potrebno prikazati. Osim toga, mogli bismo u stvarnom vremenu odrediti za svaku regiju koji pristup najviše donosi i njega za tu regiju koristiti. Mogao bi se više istražiti i implementirati svijet ostvaren oktalnim stablom, napraviti implementacija simulacije fluida u prostoru opisanom volumnim elementima, istražiti metoda praćenja zrake i metoda pokretne kocke. Mogao bi se istražiti i zakrivljeni prostor opisan volumnim elementima.

Crtanje različitih modela pomoću jednog poziva

Sažetak

Postoje mnogi razlozi zašto nam program neučinkovito koristi sklopovlje za crtanje. Neki od njih su opravdana ograničenja brzine prijenosa informacija između sklopovlja i računanja na samom sklopovlju, no moguće je da nas ograničava i upravljački program. Ovaj rad opisuje način na koji možemo smanjiti količinu posla kojeg upravljački program treba obaviti.

Ključne riječi: optimiranje, indirektno crtanje, prikaz svijeta

Drawing multiple different models with a single draw call

Abstract

There are many reasons why programs could be underutilizing graphics hardware. Some of them are justified - for example data transfer speeds, vertex, fragment data processing speeds, but it could be that we are bottlenecked by the graphics driver. This seminar shows how we can reduce the amount of work that the driver has to do to render a scene.

Keywords: optimisations, indirect drawing, world rendering