Marching Cubes

Bruno Perković

Siječanj 2024

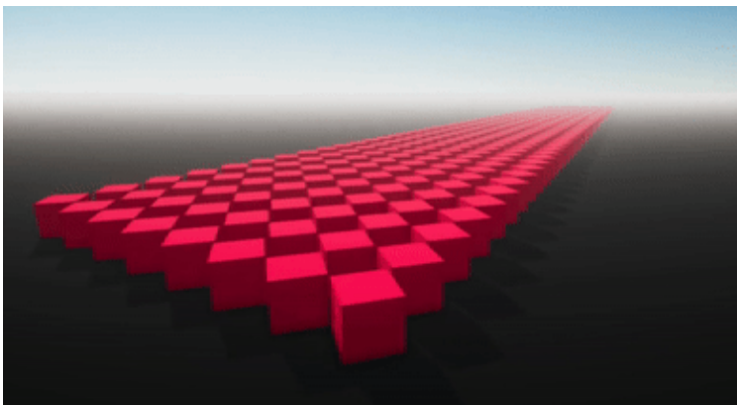

Figure 1: Prikaz kocaka kako marširaju

1 Uvod

U okviru ovog projekta, fokus je bio na razvoju algoritma "marching cubes" u Unity okruženju, zajedno s implementacijom vizualizacije rezultata. Algoritam marching cubes koristi se za generiranje 3D modela pomoću nekakvih podataka, u ovom slučaju perlinovog šuma.

Također, u istom Unity projektu nalazi se i sustav čestica, izrađen na prethodnoj laboratorijskoj vježbi.

2 Upute za pokretanje

Za pokretanje ovog projekta, prvo osigurajte da imate Unity3D razvojno okruženje instalirano na vašem računalu. Nakon toga, preuzmite projekt datoteku i raspakirajte je na željeno mjesto. Zatim pokrenite Unity3D, odaberite opciju "Open Project" te navigirajte do glavnog direktorija projekta.

Scena pod nazivom "MarchingCubes" sadrži ogledni primjerak algoritma marching cubes. Otvorite ovu scenu kako biste istražili generiranje 3D modela

pomoću ovog algoritma. Možete promatrati rezultate i prilagoditi parametre kako biste dobili različite oblike.

3 Postavke

Ovdje ukratko prolazimo kroz parametre koje korisnik može modificirati preko sučelja.

3.1 Osnovno

- **Mesh Material** Definira materijal za vizualizaciju generiranog mesh-a.
- **Mesh Size** Postavlja uniformnu veličinu tijela, s zadanim vrijednostima postavljenim na 1.
- **Value Border** Određuje granicu gustoće koja razlikuje unutarnji i vanjski prostor tijela (vrijednosti od 0 do 1).
- **Mesh Resolution per Dim** Postavlja rezoluciju generiranog mesh-a uniformno po dimenzijama x,y i z.

3.2 Perlinov šum

- **Octaves** Definira broj oktava.
- **Lacunarity** Definira vrijednost lakunarnosti.
- **Frequency** Određuje frekvenciju.
- **Offset** Određuje odstupanje.
- **Frequency Speed** Kontrolira brzinu promjene frekvencije funkcije.
- **Offset Speed** Utječe na brzinu promjene odstupanja generirajuće funkcije.
- **Lacunarity Speed** Kontrolira brzinu promjene lakunarnosti.

4 Implementacija

4.1 StableCubeMarcher.cs

Ova C# skripta djeluje kao most između CPU-a i GPU-a. Odgovorna je za komunikaciju s GPU-om, slanje parametara i vizualizaciju dobivenih oblika. Skripta olakšava koordinaciju operacija između CPU-a i GPU-a.

4.2 StableCubeMarcher.compute

Veći dio operacija odvija se u ovom sjenčaru. Ovdje se uzorkuje Perlinov šum te na temelju njega gradi oblik koji se potom transformira te umetne u spremnik.

4.3 StableMarchShader.shader

Ove skripte su zaslužne za iscrtavanje tijela iz spremnika. Koriste se vertex, geometry i pixel sjenčari.

5 Rezultati

Algoritam se uspješno izvodi s više od 100FPS pri rezoluciji od 128x128x128. Zahvaljujući tome što se generacija tijela odvija u globalnom sustavu, implementacija se lako može particionirati, što pridonosi efikasnom upravljanju resursima.

Daljnji razvoj projekta uključuje implementaciju post-process tehnika poput generiranja normala, što bi omogućilo primjenu Phongovog sjenčanja.

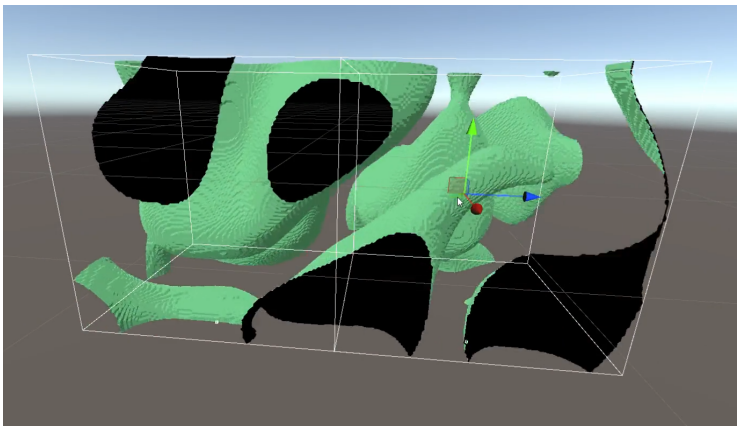

Figure 2: Prikaz tijela generiranog Perlinovim šumom