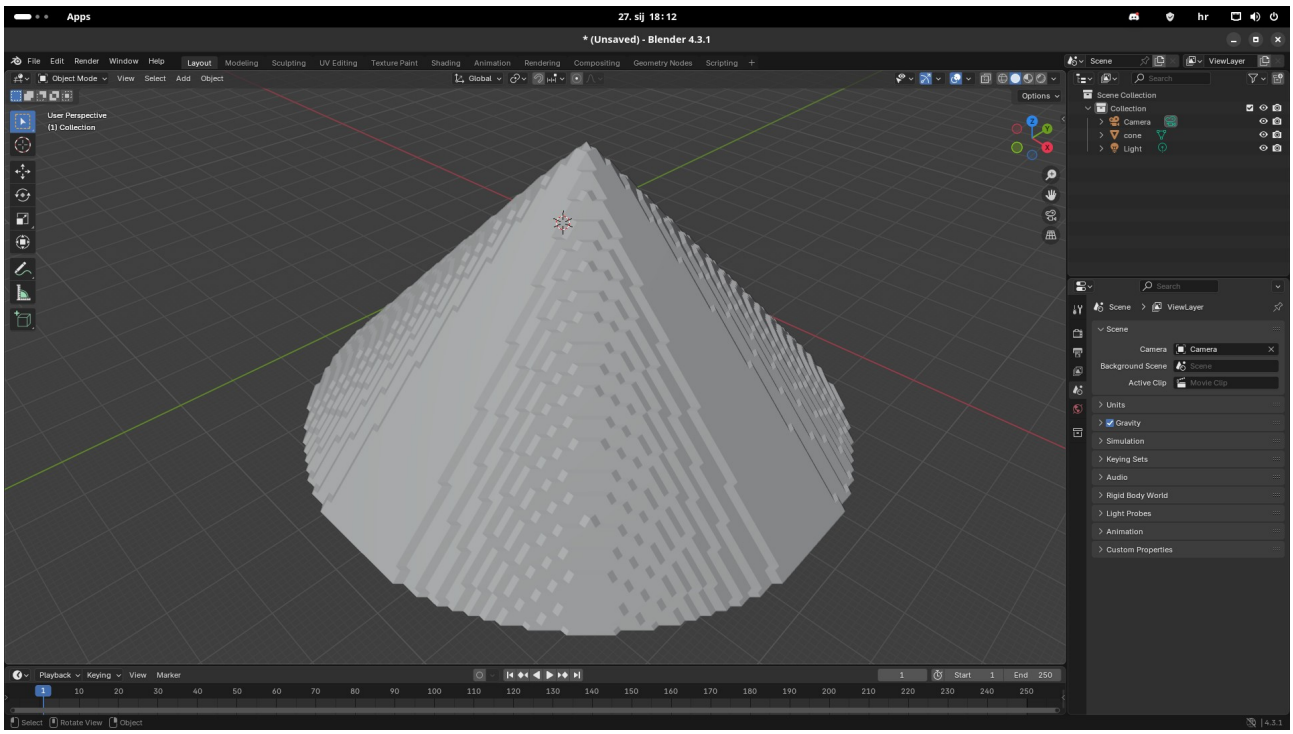
Algoritam Marching Cubes

3. laboratorijska vježba implementira algoritam marching cubes za arbitrarnu funkciju.
Rezultat pokretanja je rezultatni objekt u Wavefront .obj formatu koji se može spremiti na preferirani put na disku.
Ako se kod pokreće preko uređivača koda poput Intellij ili Eclipse, main funkcija je u datoteci "Terrain". Da se dobije tim putem rezultat, u kodu se treba odkomentirati željeni isječak koda.
Alternativno, moguće je napisati vlastitu klasu koja implementira sučelje "Function", koja sadrži metodu koja za dane koordinate x, y i z vraća *True* ako se te koordinate nalaze ispod funkcije, te vraća *False* ako se koordinate nalaze iznad funkcije. Algoritam Marching Cubes koristi tu metodu da konstruira teren – područje gdje koordinata s oznakom *False* graniči s koordinatom s oznakom *True* označava područje gdje se treba nacrtati poligon.
Metode trenutno implementirane pomoću lambda izraza su:
Spheroid – klasa koja je definirana preko 3 parametra: *height*, *width* i *depth*. Parametri određuju istoimene dimenzije sferoida.
Cube – klasa je također definirana preko 3 ista parametra kao i **Spheroid**. Prikladnije ime klase stoga bi bilo Cuboid.
Cone – klasa definira stožac. Ista 3 parametra kao i dosad se mogu koristiti da se mijenjaju svojstva stošca.
Uz ove osnovne metode, implementirane su dve varijante Perlinovog šuma – **Perlin2D** i **Perlin3D**. Obje metode koriste Perlinov šum za generaciju pseudonasumičnog terena, no **Perlin2D** se koristi za područja nalika brdima i dolinama, dok **Perlin3D** služi za generaciju područja nalik spiljama. Uz već definirane parametre height, width i depth, Ove dve klase također imaju parametre cell_x_amount, cell_y_amount i cell_z_amount (**Perlin2D** nema parametre depth i cell_z_amount jer nema strogo definiranu treću dimenziju). Prva 3 parametra služe za definiciju veličine prostora područja, dok zadnja 3 parametra služe kako bi se moglo utjecati na proces generacije terena – inače bi svaki teren iste veličine izgledao isto.
Parametri *height*, *width* i *depth* mogu se mijenjati pri samom početku metode main i uvijek služe za određivanje veličine prostora s kojim algoritam raspolaže.

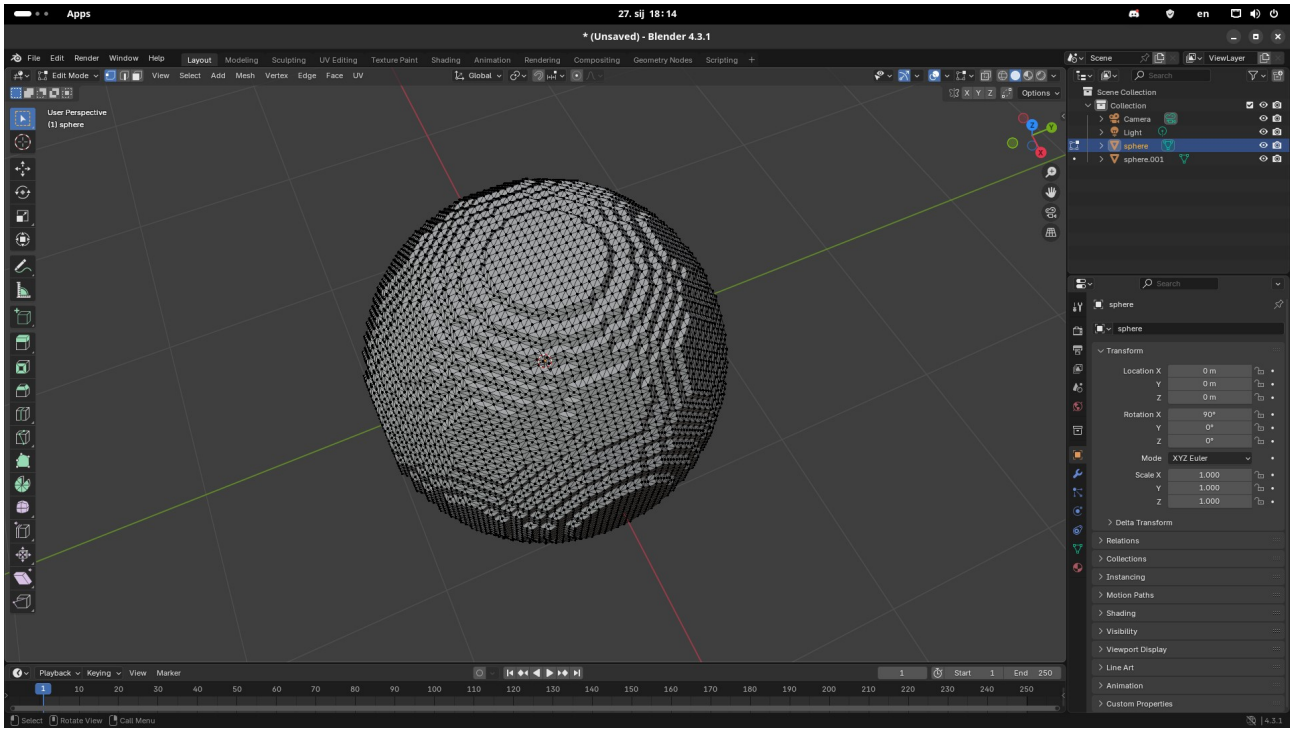
Druga varijanta pokretanja postupka generacije je pokretanjem .jar datoteke marchingCubes.jar. Kad se pokreće preko terminala, datoteka na ulaz prima 3 parametra: height, width i depth, te jednu od idućih ključnih riječi: spheroid, cube, cone, perlin2d ili perlin3d. Ključne riječi odgovaraju iznad opisanim klasama, a parametri definiraju veličine, tj. kompleksnosti generiranih objekata. Potrebno je primijetiti da pokretanje preko jar datoteke ne nudi mogućnost mijenjanja parametara Perlinove buke, tako da će rezultat za istu veličinu objekta uvijek biti isti.

Objekt je, nakon spremanja, moguće uredno prikazati bilokojim alatom koji nudi mogućnost otvaranja Wavefront .obj datoteka, poput Blendera.

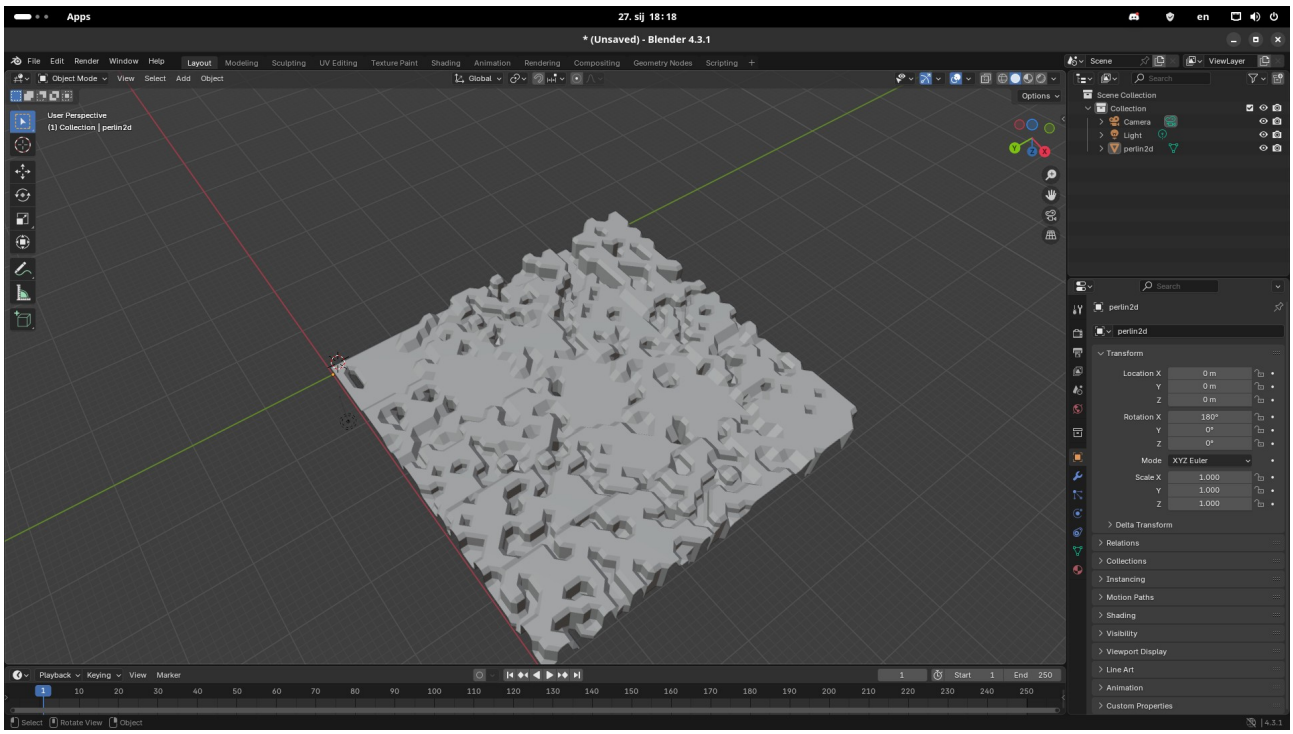
NAPOMENA: algoritam nije paraleliziran! Preveliki iznosi parametara height, width i depth rezultirati će jako dugim čekanjem izvršavanja programa, i također je moguće ostati bez memorije na hrpi. Savjetuje se držati brojeve ispod 300 za osnovne funkcije, te ispod 100 za funkcije perlinovog šuma. Također, parametri ne smiju biti manji od 3 jer bi to uzrokovalo pogrešku pri radu programa!



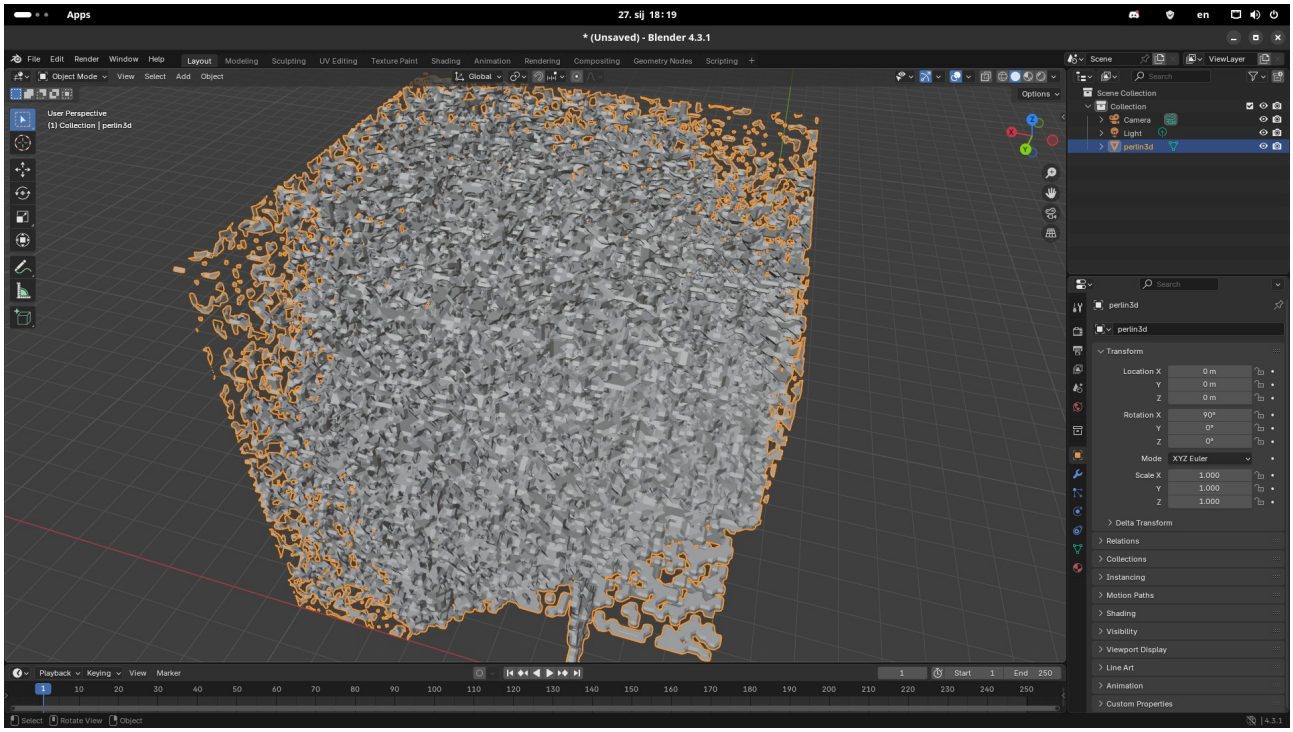
Slika 1. Prikaz stošca s parametrima *height*=*width*=*depth*=101



Slika 2. Prikaz poligona sfere parametara *height*=*width*=*depth*=76



Slika 3. Prikaz terena generiranog metodom **Perlin2D**



Slika 4. Prikaz terena generiranog metodom **Perlin3D**

Potencijalne nadogradnje

U datoteci experiments nalazi se kratka animacija koja prikazuje sferu kako se deformira dok pada. Efekt je postignut ručno kroz što bi se efektivno moglo opisati kao stop-motion, gdje se kroz kadrove miče jedna generirana sfera i umeće druga. Sve sfere su generirane ručno tako što su se namještali parametri dimenzija.
No moguće je kroz jednostavnu implementaciju nove funkcije stvoriti niz objekata koji prikazuju promjenu oblika u ovisnosti o varijabli t kroz promjenu funkcije. Problem je što bi postupak spremanja bio memorijski zahtjevan jer je potrebno objekt za svaki kadar zasebno spremiti.
No time je u teoriji moguće postići proceduralnu generaciju terena u ovisnosti o vremenu, čime bi bilo daleko jednostavnije konstruirati navedenu animaciju.