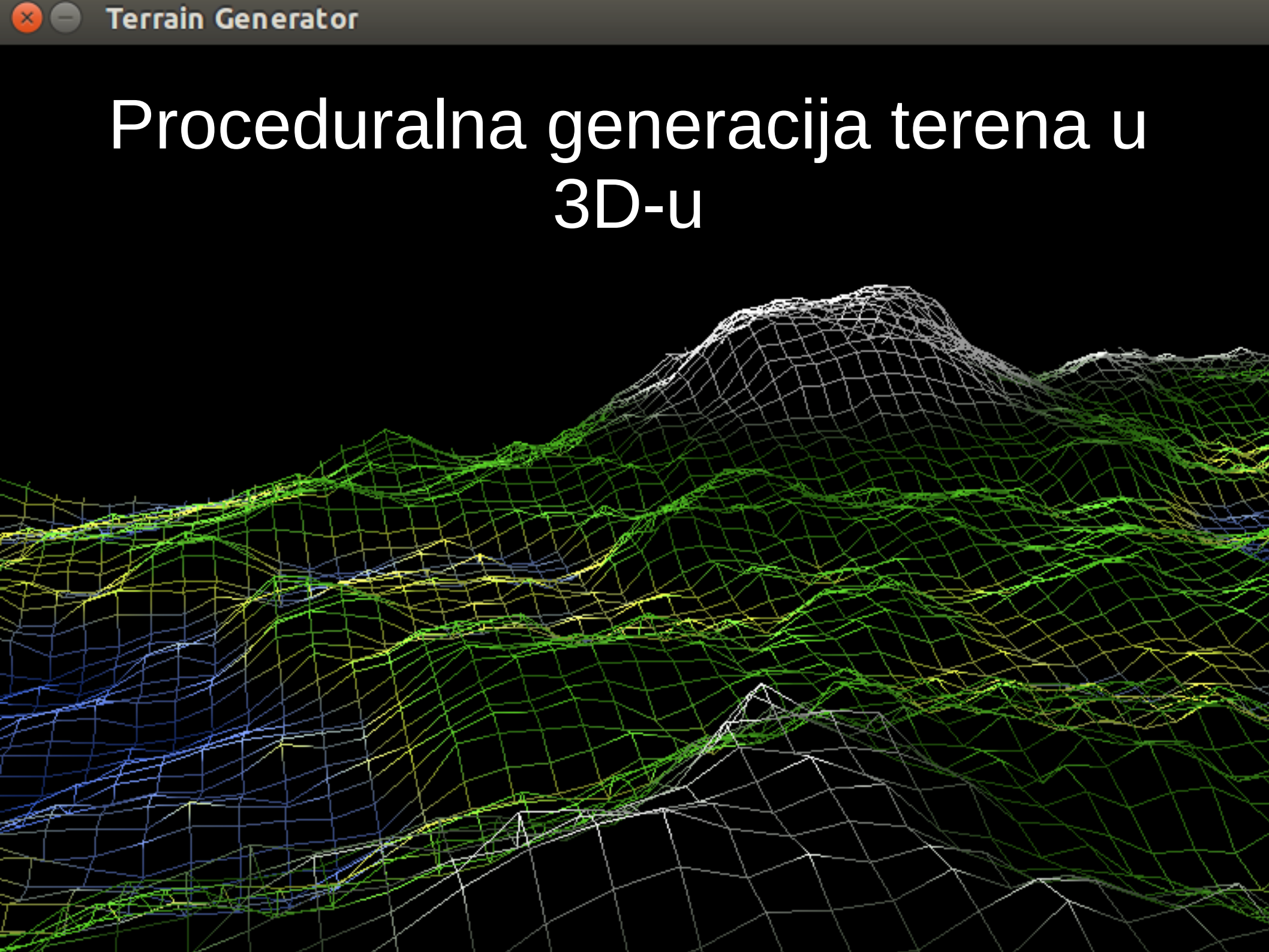
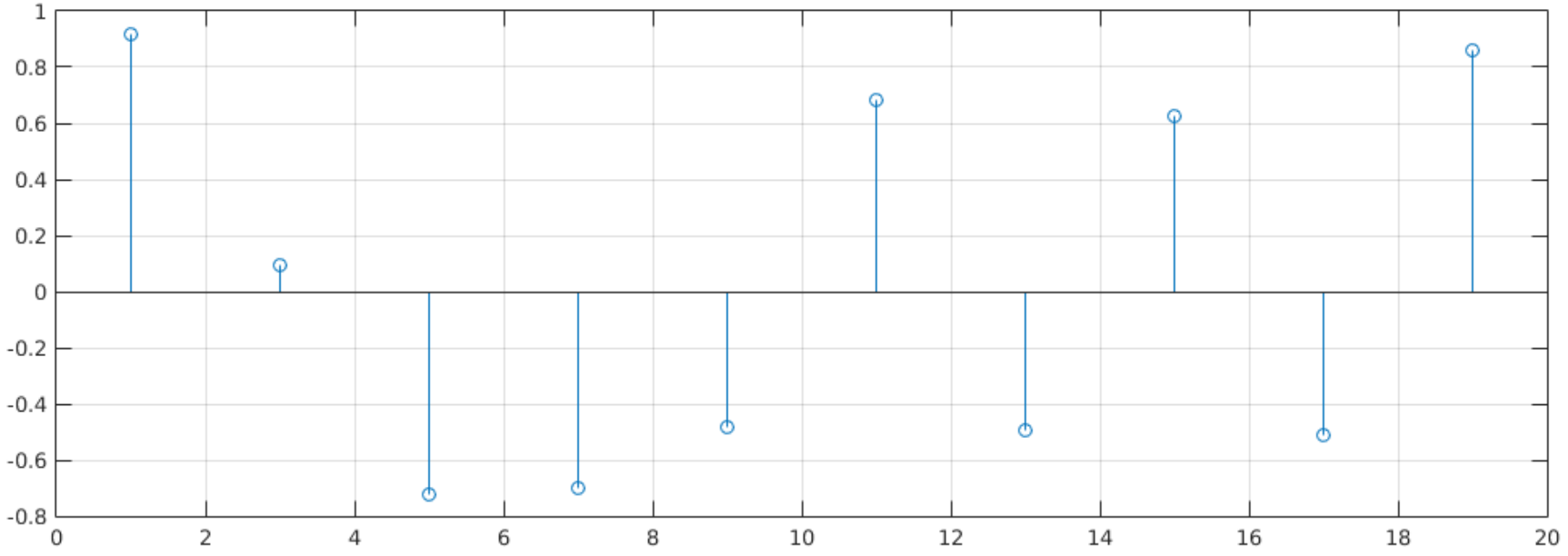

Terrain Generator
Proceduralna generacija terena u 3D-u

Pocetak naseg generisanja ce se zasnivati na nasumicnosti ili mozda bolje receno na pseudo nasumicnosti (pseudo jer cemo vrsiti vise korekcija na tome sto je nasumicno, a i nista nije stvarno nasumicno). Da bi objasnili pojam pseudo nasumicnosti prvo cemo se vratiti na neke osnove 3D prikaza i programiranja. Dakle mi imamo 3 ravni od kojih smo generisali nesto u ravnima X i Z ravanima, Y je visina i na njoj je 0. Principi generisanja se zasnivaju na trouglovima odnosno svako telo je generisano iz gomile malih trouglica.

Za generisanje tacaka visine koristimo javin random generator koji vraca brojeve u opsegu od 0 do 1. Mi ovaj opseg pomeramao na interval od -1 do 1 jer zelimo da postignemo I predele ispod nivoa mora.

```
private float getNoise(int x, int z) {
        random.setSeed(x*55228 + z*62457+ seed);
        return random.nextFloat() * 2f - 1f;
}
```

Seme sluzi da za iste kordinate generise uvek iste brojeve. Mnozi se sa nekim odabranim vecim brojevima jer slicna semena daju bliske vrednosti, te se ovim povecava razudjenost.



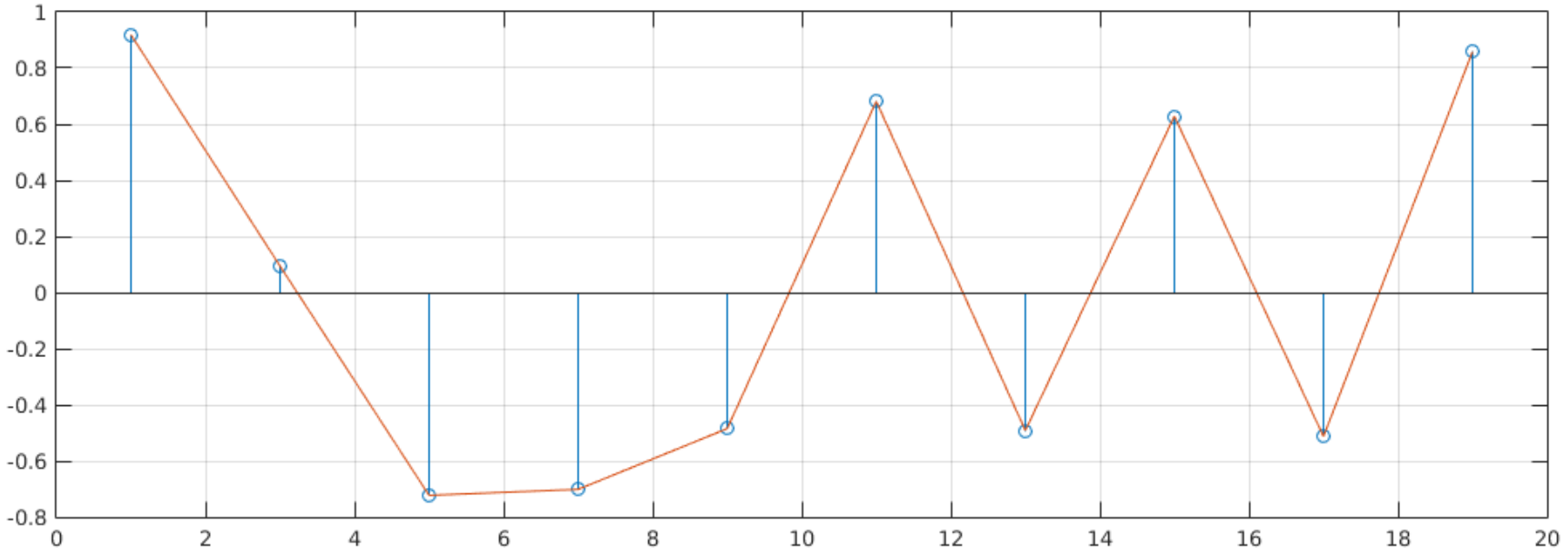
Prvo ogranicenje koje uvodimo jeste ogranicenje maksimalne visine (recimo 100). Nase slucajno generisane brojeve (funkciju) mnozimo sa maksimalnom visinom (amplitudom). Time menjamo interval na
-h(-100) do h(100).
Druga stvar je to sto kod nasumicnog generisanja javlja se problem da dve susedne kordinate (recimo (x,y) i (x+1,y)) mogu da imaju ogroman skok (npr. -97, i 100) sto nije prirodno, te da bi se to smanjilo koristimo u kodu osluskivanje okoline (neki filter, npr blurovanje) odnosno tacke okoline imaju uticaj na vrednost visine kordinate. Npr. Uzimamo da samo okolne tacke uticu na dobijanje visi nase tacke tako da imamo 8 tacaka koje okruzuju nasu.

Sledi korak interpolacije. Ovaj korak se moze vrsiti na milion nacina. Ispod je linearna.

```
private float linterp(float x, float z, float izmedju) {
        return x + (z-x)*izmedju;
    }
```

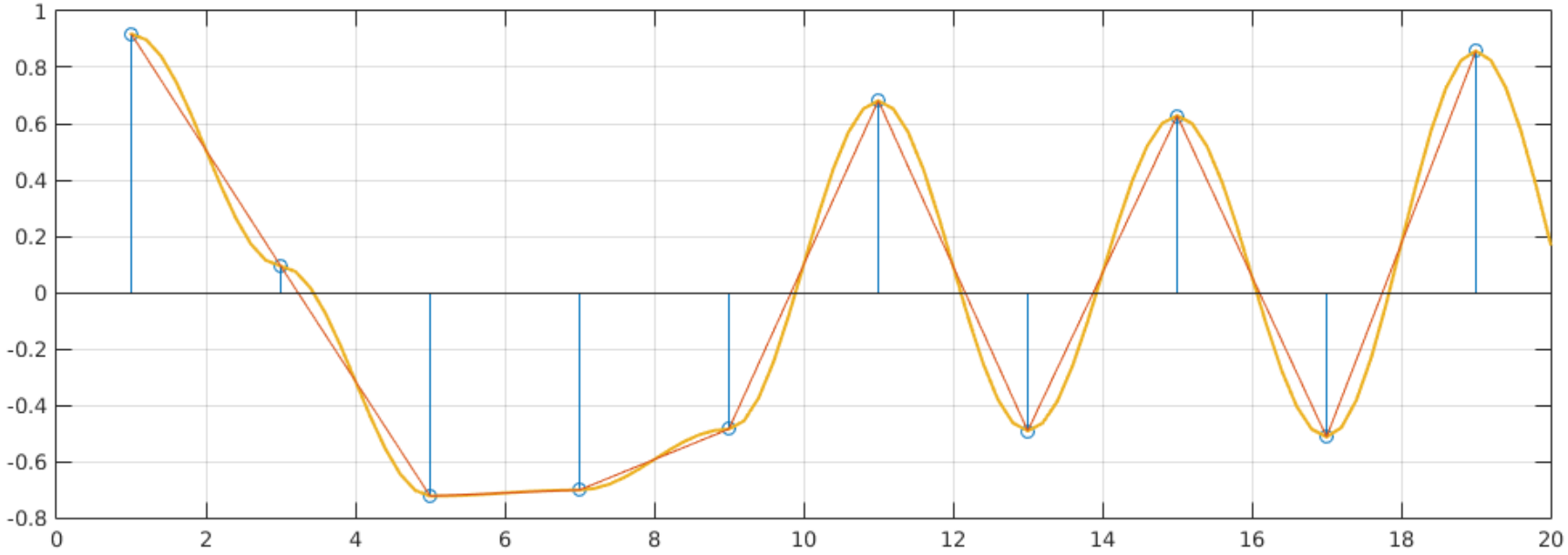
izmedju je broj izmedju 0 i 1 koji pokazuje koliko je blizu trazena tacka tacki x

* (0 to je tacka x, 1 to je tacka z, 0,5 izmedju x i z)

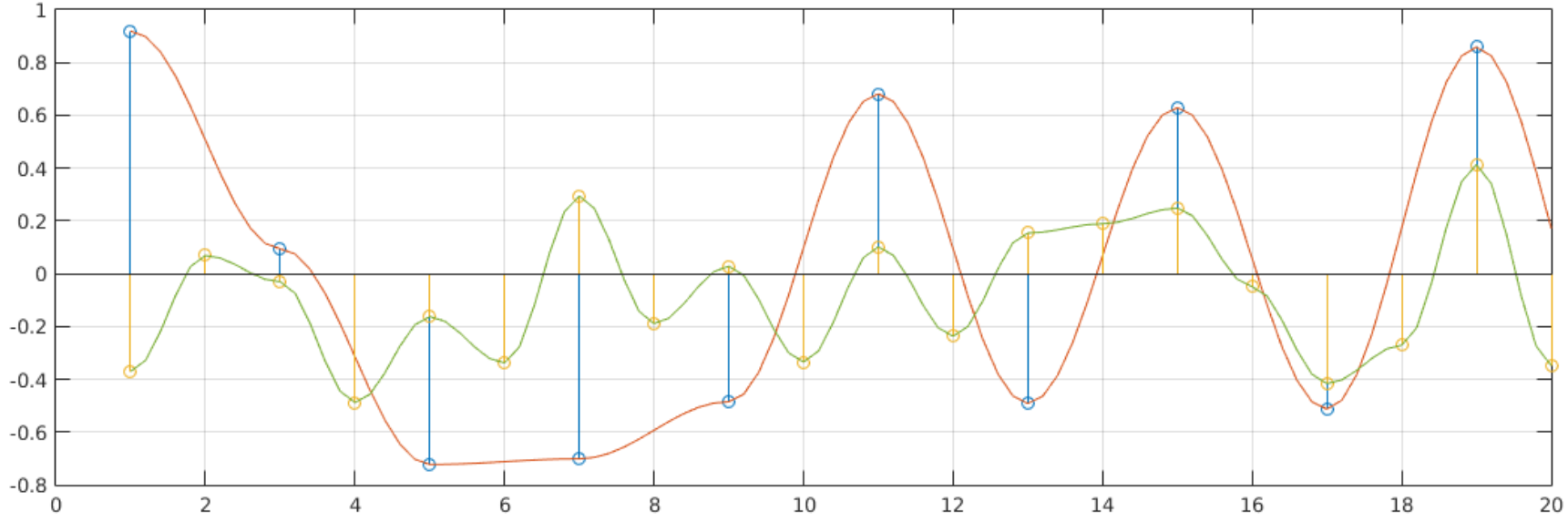


Kao sto se da primetiti ovo nije bas prirodno te cemo pokusati npr. Sa kosinusnom iterpolacijom gde dobijamo i krivine. Sve interpolacije su kroz 2 tacke, moze se implementirati i kroz vise tacaka.

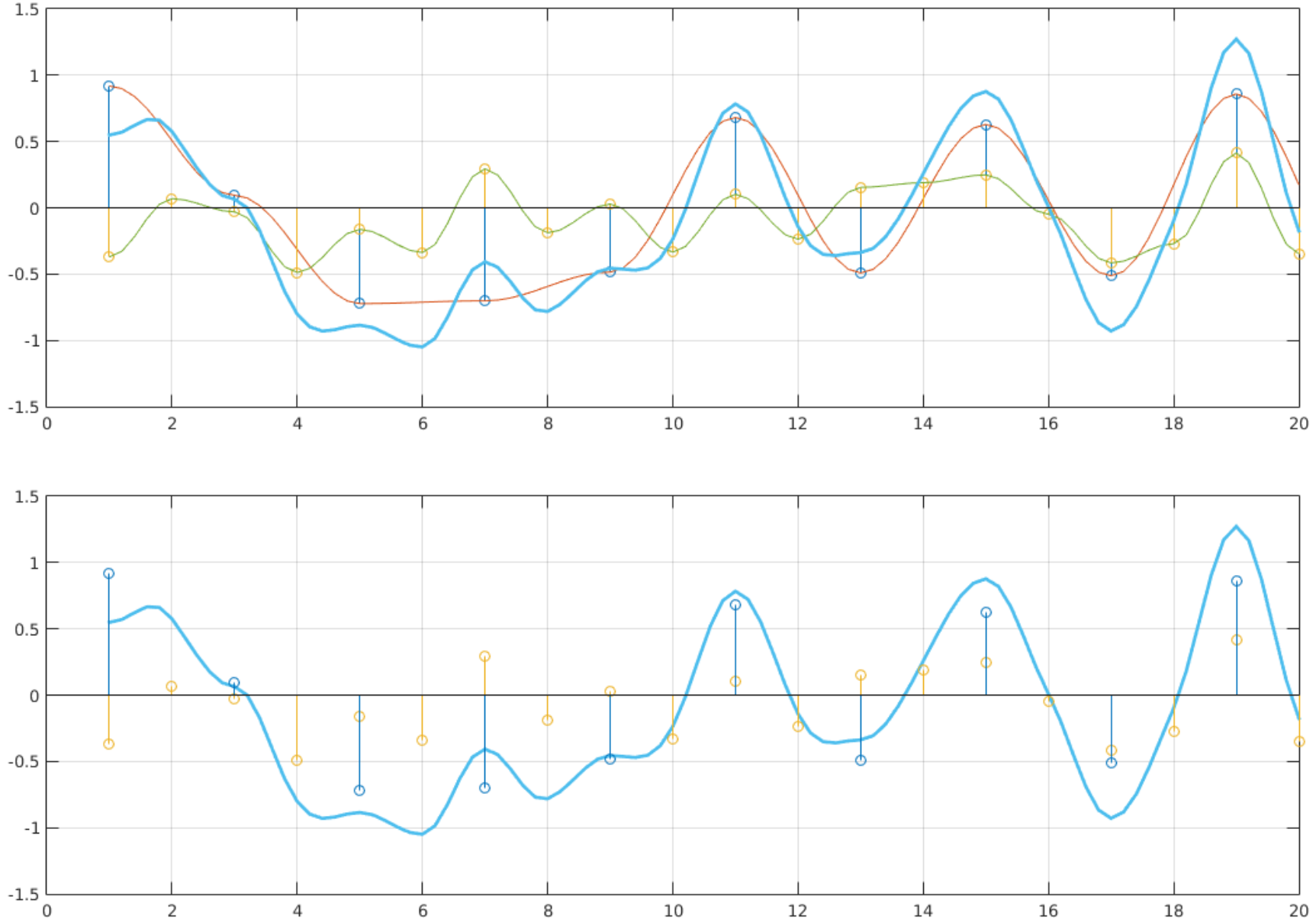
```
private float cosinterp(float x, float z, float izmedju) {
      double theta = izmedju * Math.PI;
       float f = (float)(1f - Math.cos(theta)) * 0.5f;
       return x * (1f - f) + z * f;
}
```



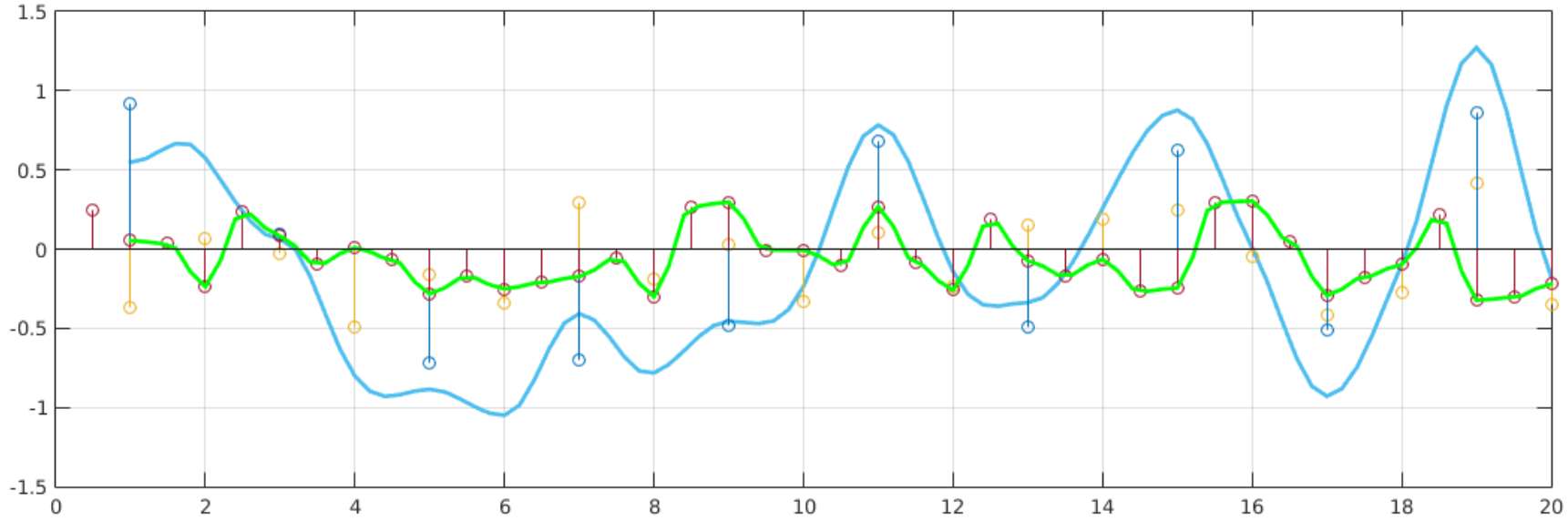
No idalje ovo nije bas prirodno te se pribegava sabiranju funkcija, da bi se dobile sitne promene funkcije koje su prirodne. To se vrsi tako sto se frekvencija funkcije poveca ali se njena amplituda smanji.



Dalje se saberu te dve funkcije i dobija se:

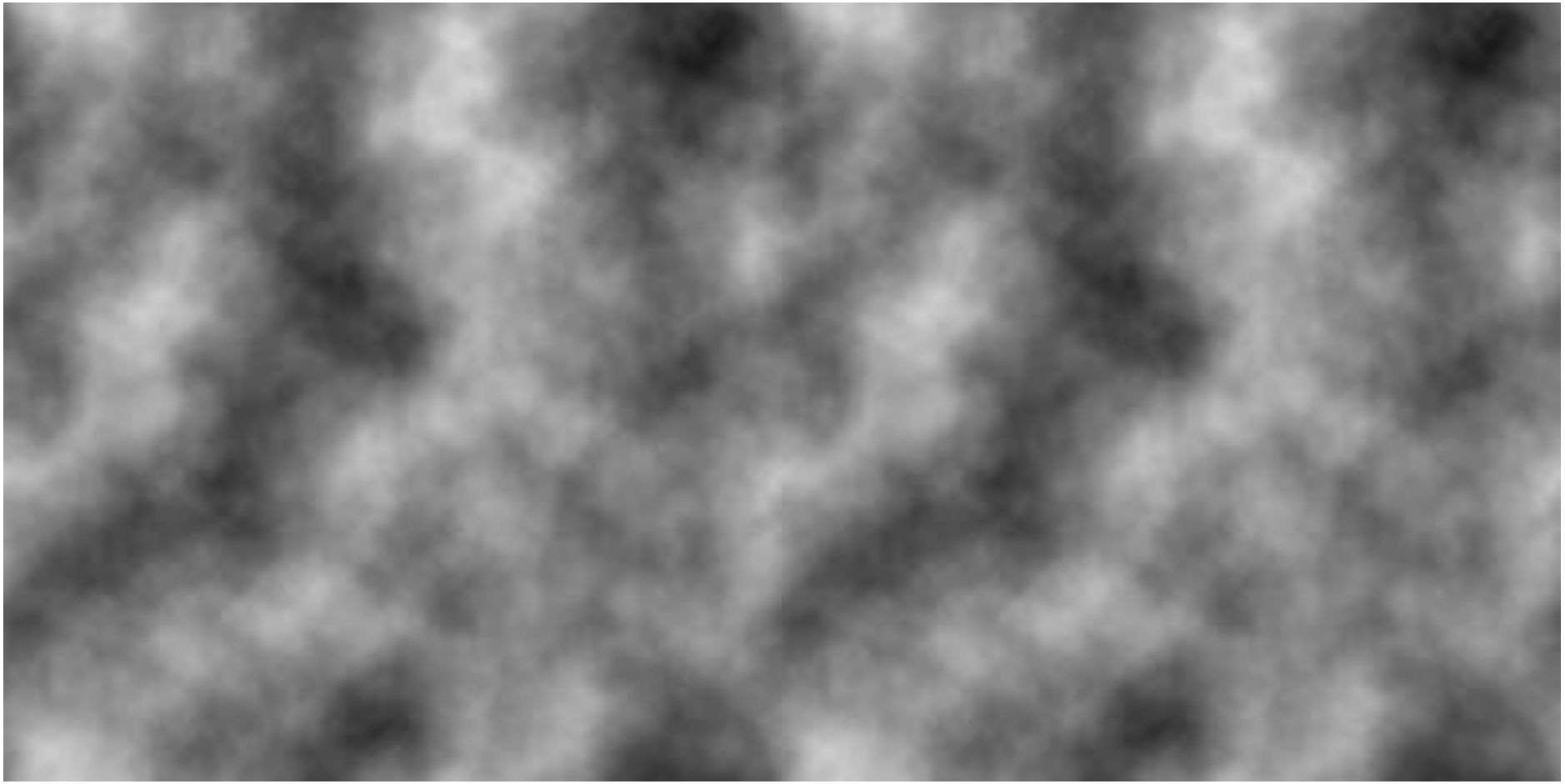


Vec se dobija malo realnija slika terena sada ovaj postupak se ponavlja dok nismo zadovoljni.



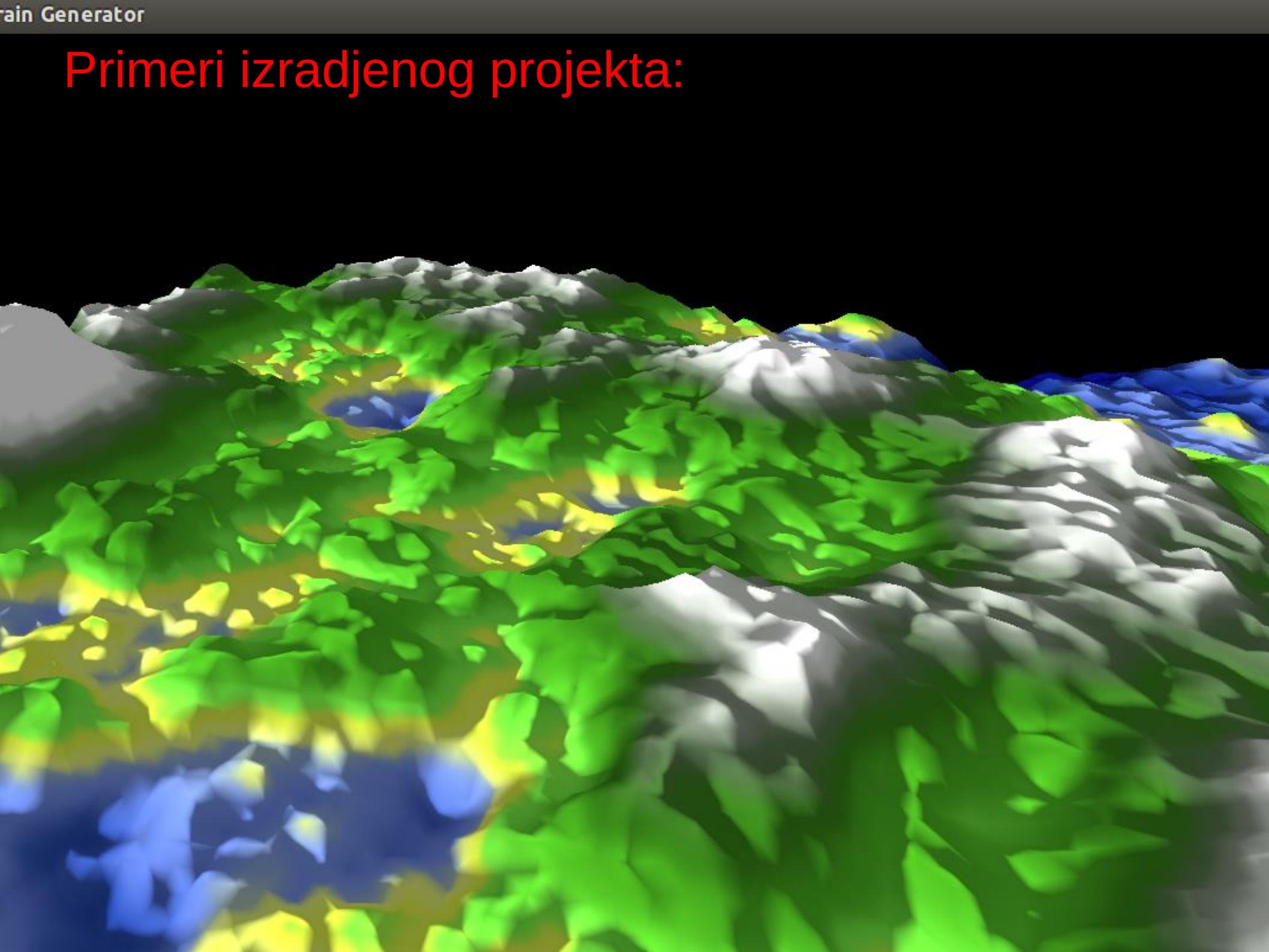
Bitno za izvuci odavde jeste to da povecanjem amplitude dobijamo detaljnost funkcije, ali ne zelimo da nam ta detaljnost kvari nas prvobitni oblik, te smanjujemo njen uticaj na globalni oblik tako sto joj smanjujemo amplitudu. Ove korake mozemo ponavljati vise puta.

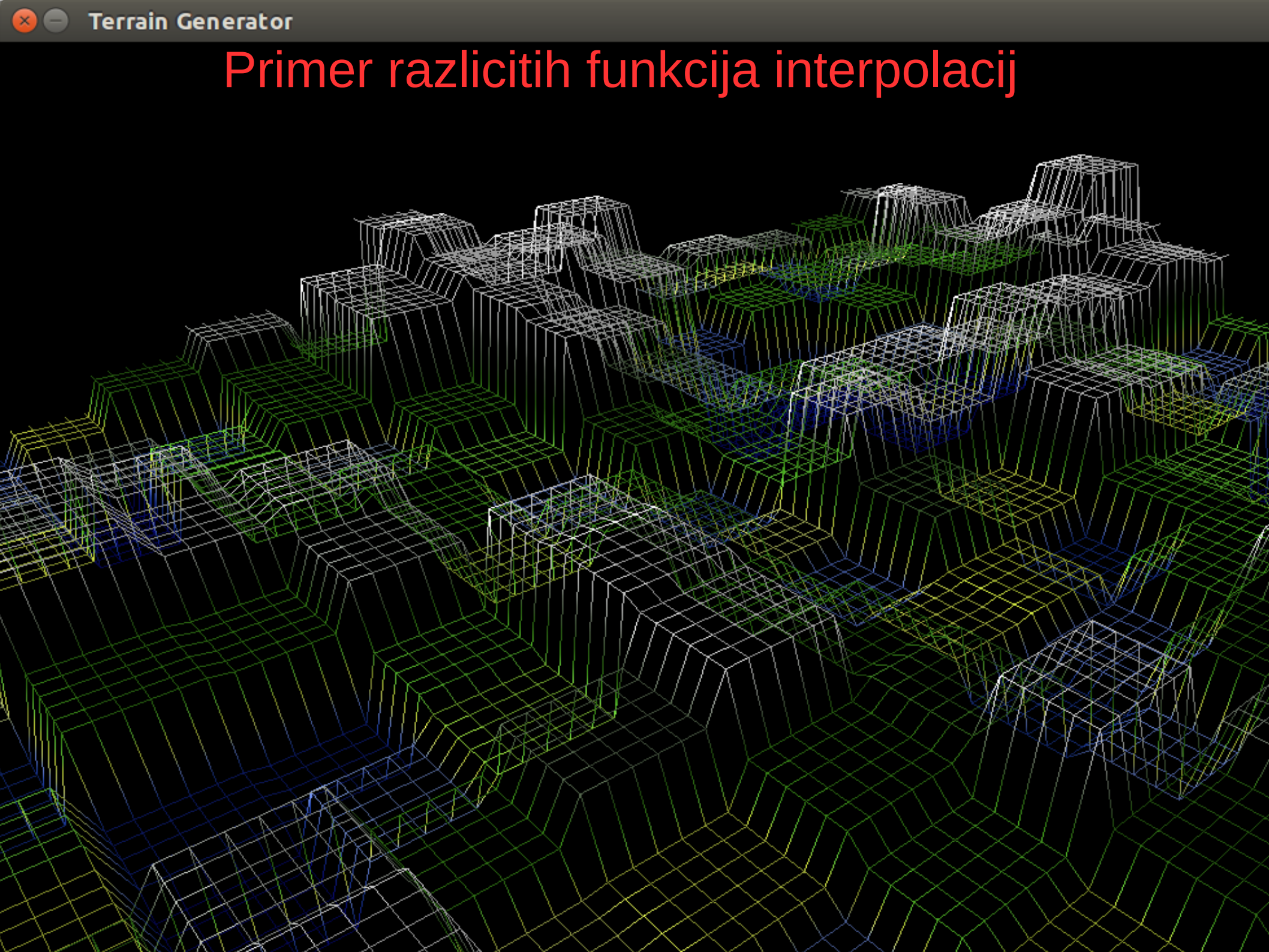
Gore zadato je bio prikazan postupak u 2D-u. No sve isto vazi I za 3D prostor.
Konacan izgled nase funkcije kad bi se prikazivao bojom treba da izgleda kao neka slika suma sa blagim prelazina. Ovakvi sumovi se zovu Perlin Noise I koriste se kao Height Mape za generaciju terena.



rain Generator

Primeri izradjenog projekta:




Terrain Generator
Primer razlicitih funkcija interpolacij