# Algorithms for fractal Landscape generation

A Project Submitted
in Partial Fulfillment of the Requirements
for the Degree of
**Bachelor of Technology**
in
**Computer Science and ECE**

As part of the open elective course **"Introduction to Fractals"**

By
1. Mogili Giridhar (ECE)
2. Chakka Sai Teja (CSE)


SCHOOL OF ENGINEERING AND TECHNOLOGY
BML MUNJAL UNIVERSITY GURGAON
April, 2021

## Abstract:

This paper aims to provide information about algorithms for fractal landscape generation. Three methods for fractal landscape generation are studied and compared. We also discussed how real landscapes are different from fractal landscapes.

We discussed how Fourier transform method is different from others. We used diamond-square algorithm to simulate fractal landscape.

## Introduction:

A fractal landscape is a surface generated using a stochastic algorithm designed to produce fractal behavior that mimics the appearance of natural terrain**.**

A generated landscape can be made quite realistic by adding colour, proper lighting, water, plants, atmospheric effects and other such things.

The result of this procedure is not a deterministic fractal surface, but rather a random surface that exhibits fractal behaviour.

The output of the algorithms discussed is a set of altitudes assigned to a two-dimensional grid. Inputs into the algorithms are parameters that define certain desired characteristics of the generated landscape, mainly its roughness.

Any algorithms should use randomness, otherwise there will be no uniqueness or interesting results. But we can't simply assign random values because two points that are close to each other should have similar altitude. So, we have to use a algorithm the closeness of two points in determining their altitude.

Firstly, landscapes are self-similar. Looking at a landscape at different scales, though, you will notice that it exhibits the same basic characteristics at every scale.

The main characteristic that define the landscape as a fractal is Fractal dimension. Every landscape have a fractal dimension.

In 1967, Benoit Mandelbrot asked a popular question, "How long is the coast of Britain?". A measurement of the length of the coastline would depend entirely on the scale of the measurement device used. If you try to measure a coastline with 1 kilo-meter at a time you think that it produces an accurate result but not as accurate as when measured 1 meter at a time.

The answer is not simple because the coastline of Britain does not have a length because it is not one-dimensional. It is a fractal.

The landscape generating algorithms uses landscapes fractal properties and converts random numbers into landscapes.

The Triangle Division and Diamond-Square algorithms use iterative processes. They take advantage of the self-similarity property by applying the same rule to the landscape at a smaller scale with each iteration.

The Fourier Transform algorithm uses discrete Fourier transforms to represent a landscape in the frequency domain. By moulding the frequencies, this algorithm can convert a set of random numbers into a landscape.

## Details of the work:

## Triangle Division Algorithm:

This algorithm takes a triangle-shaped terrain, and it divides each triangle into four smaller triangle at each iteration by applying a random perturbation to each new vertex created.

1. Start with a triangle. Pick random altitudes for the endpoints.
2. For each edge on the triangle, take the midpoint and add a random perturbation. The perturbation is equally distributed between –kr and kr , where k is the length of the divided edge and r is a roughness parameter.
3. Divide the triangle into 4 smaller triangles by connecting the midpoints, then repeat the process on each of the smaller triangles.

The process is iterated until the desired amount of detail is reached. Notice that at each iteration the random perturbations are dependent on the size of the triangles at that iteration, so the amount of perturbation becomes smaller with every iteration. This property makes the generated object landscape-like. The fact that the same process is applied to the landscape but at a smaller scale each time makes it self-similar or fractal.

This algorithm has a roughness parameter (r) which gives some control over the resulting landscape. Notice that with higher r, the perturbation decreases more quickly after each iteration than with lower r. So with relatively high r, the perturbations become relatively small at high levels of detail. The result is a smooth landscape. Conversely, a low value of r will result in a rough landscape. Therefore, r is called the roughness parameter.

Below are two landscapes using the triangle division algorithm. The algorithm can be applied to a square grid by dividing the square along a diagonal, creating two triangles.

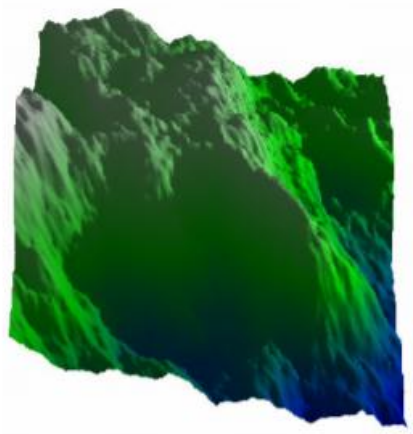


**Figure 1 - Triangle Division r = 1.0**

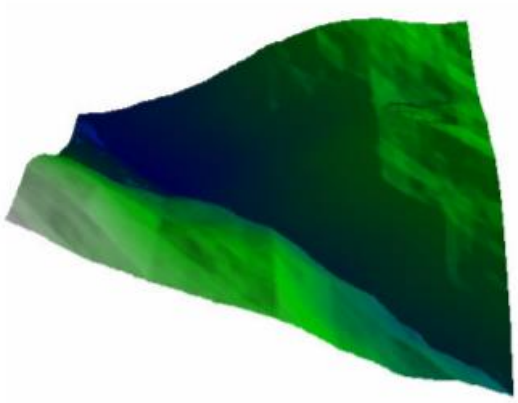


**Figure 2 - Triangle Division r = 1.5**

Notice that using a higher value for r produces a much smother landscape.

An apparent flaw in this algorithm is the forming of ridges along the edges of the triangles. This is more obvious in the smoother landscape, but notice in Figure 1 there is a ridge that forms along the main diagonal. This is a result of the geometry of the algorithm. Ideally, a fractal landscape would have no artificial forms such as this. The reason these ridges appear is

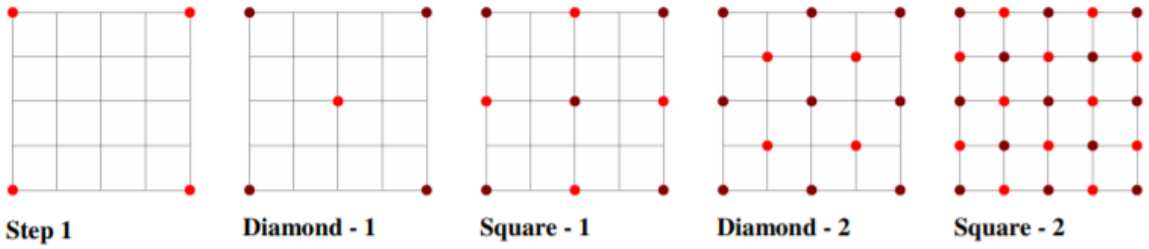
because the altitude of each point is determined from only two other points, so it is dependent on points in two opposite directions, but not in any other direction.

## Diamond-Square Algorithm:

This algorithm improves upon the flaw pointed out in the Triangle Division Algorithm, by having each point depend on points in four directions rather than two. The algorithm is as follows:

1. Assign random altitudes to the four corners of a grid.
2. Diamond step: Average the four corners and add a random perturbation evenly distributed between –ri and ri . Assign this to the midpoint of the four corners.
3. Square step: For each diamond produced, average the four corners and add a random perturbation with the same distribution.
4. Repeat steps 2 and 3 for the desired number of iterations

The value r is a roughness parameter, and i is the current iteration. The following diagrams illustrate the points computed in the diamond and square steps of the first two iterations:




Notice that this algorithm uses a different computation to determine perturbations as was used in the Triangle Division algorithm. Here the random numbers are scaled by a factor of ri , compared to kr in the other algorithm, where k is the side-length of a triangle or square. The two methods are, in fact, equivalent. Notice that the side-length of a triangle or square in either algorithm is halved at each iteration, so k = (½)i and kr = (½)ri, which is equal to using r'i where r' = (½)r . The main difference is that in this algorithm, increasing r increases roughness.

Here are two fractal landscapes generated by using the Diamond-Square algorithm:

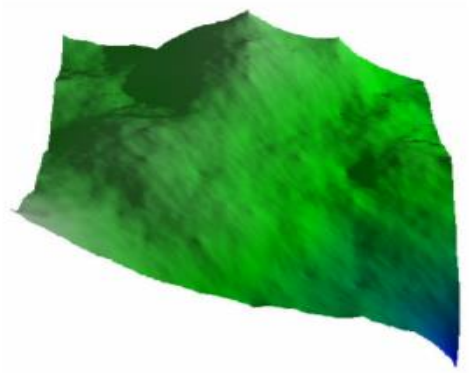


Figure 3 - Diamond-Square r = 0.4

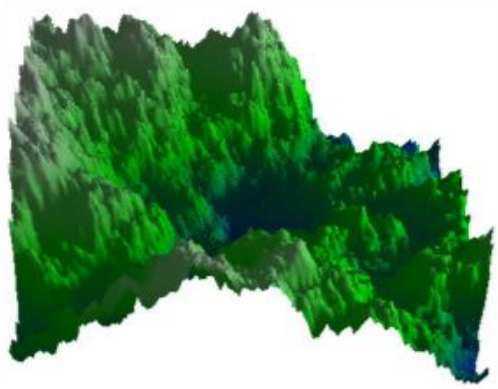


Figure 4 - Diamond-Square r = 0.6

As if you observe the above figures, there aren't any ridges as like appeared in Triangle Division algorithm, but you can see on the smother landscape there are small peaks or troths along the major gridlines. This is better than the previous algorithm, but we still do not have a truly random fractal landscape.

A possible improvement upon this algorithm would be to use a cubic spline interpolation for each point, or some other non-linear interpolation method rather than the linear interpolation that is used here. This could help smooth out the peaks and troths that are formed by taking into account very many other points rather than just the four closest points in interpolation. An algorithm using cubic splines is not explored in this paper.

## Fourier Transform Algorithm:

Basically, the idea of Fourier transform is a function can be expressed as sum of sine or cosine waves at different frequencies. The fourier transform is used to convert a function into frequency domain.

A Discrete Fourier Transform (DFT) is a Fourier Transform applied to a set of discrete values. The result obtained is also a set of complex numbers, which are amplitude frequencies of original set.

To make DFT efficient we use Fast fourier transform, which can be performed in O(n log(n)). A DFT can also be applied to multiple dimensions. Fractal landscapes using Fourier transform is landscape as sum of waves at different frequencies. Observe that the amplitude of these waves has a decreasing trend as the frequency increases.

Fourier transform is different because it is not iterative process. At a high level it is quite simple:

1. Create a 2-dimentional grid of random values.

2. Apply a 2-dimentional FFT.

3. Scale each of the values in the transform by $1/f^{r}$, where f is the frequency represented by that value, and r is the roughness parameter.

4. Apply the inverse FFT. The result is a fractal landscape.

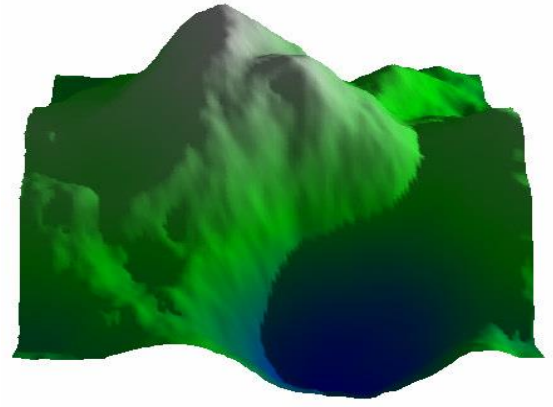


Fig-5: Fourier r = 2.5

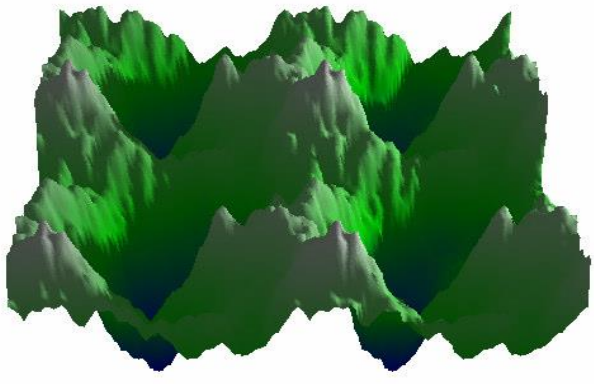


Fig-6: Fourier r = 2.0

## Conclusion:

We have examined three algorithms for generating fractal landscapes. The Triangle Division and Diamond-Square algorithms are very similar. Both algorithms produce unnatural forms in the generated landscape, the Diamond-Square being an improvement upon the Triangle Division algorithm because the computation of each point is dependent upon four, rather than two, nearby points. The Fourier Transform algorithm takes advantage of the observation that frequencies can be used to describe a natural landscape. It has the property that a generated landscape can be tiled.

It can also be modified to create mountain range-like landscapes. The landscapes generated by the algorithms described fail to mimic certain characteristics of real landscapes. An easy way to create a more realistic fractal landscape is to multiply two generated landscapes together, creating smoother valleys and rougher peaks. More advanced methods exist to convert a generated landscape into a realistic landscape such as simulating the erosion process. The algorithms discussed here provide an introduction to generating realistic landscapes.

**Reference:**


1. Brown, Adam. Fractal Landscapes. http://www.fractal-landscapes.co.uk/maths.html

2. "Fast Fourier Transform." Wikipedia. http://en.wikipedia.org/wiki/Fast_Fourier_transform

3. "Fractal landscape" Wikipedia. https://en.wikipedia.org/wiki/Fractal_landscape#:~:text=A%20fractal%20landscape%20is%20a,surface%20that%20exhibits%20fractal%20behavior.
4. Generative landscapes https://generativelandscapes.wordpress.com/2014/09/15/fractal-terrain-generator-example-9-3/


## Appendices:

```
import numpy, random

from mayavi import mlab

levels = 8

size = 2 ** (levels - 1)

height  = numpy.zeros((size + 1, size + 1))


for lev in range(levels):

  step = size // 2 ** lev

  for y in range(0, size + 1, step):

    jumpover = 1 - (y // step) % 2 if lev > 0 else 0
```

```
    for x in range(step * jumpover, size + 1, step * (1 + jumpover)):
      pointer = 1 - (x // step) % 2 + 2 * jumpover if lev > 0 else 3
      yref, xref = step * (1 - pointer // 2), step * (1 - pointer % 2)
      corner1 = height[y - yref, x - xref]
      corner2 = height[y + yref, x + xref]
      average = (corner1 + corner2) / 2.0
      variation = step * (random.random() - 0.5)
      height[y,x] = average + variation if lev > 0 else 0
xg, yg = numpy.mgrid[-1:1:1j*size,-1:1:1j*size]
surf = mlab.surf(xg, yg, height, colormap='gist_earth', warp_scale='auto')
mlab.show()
```

**Result:**

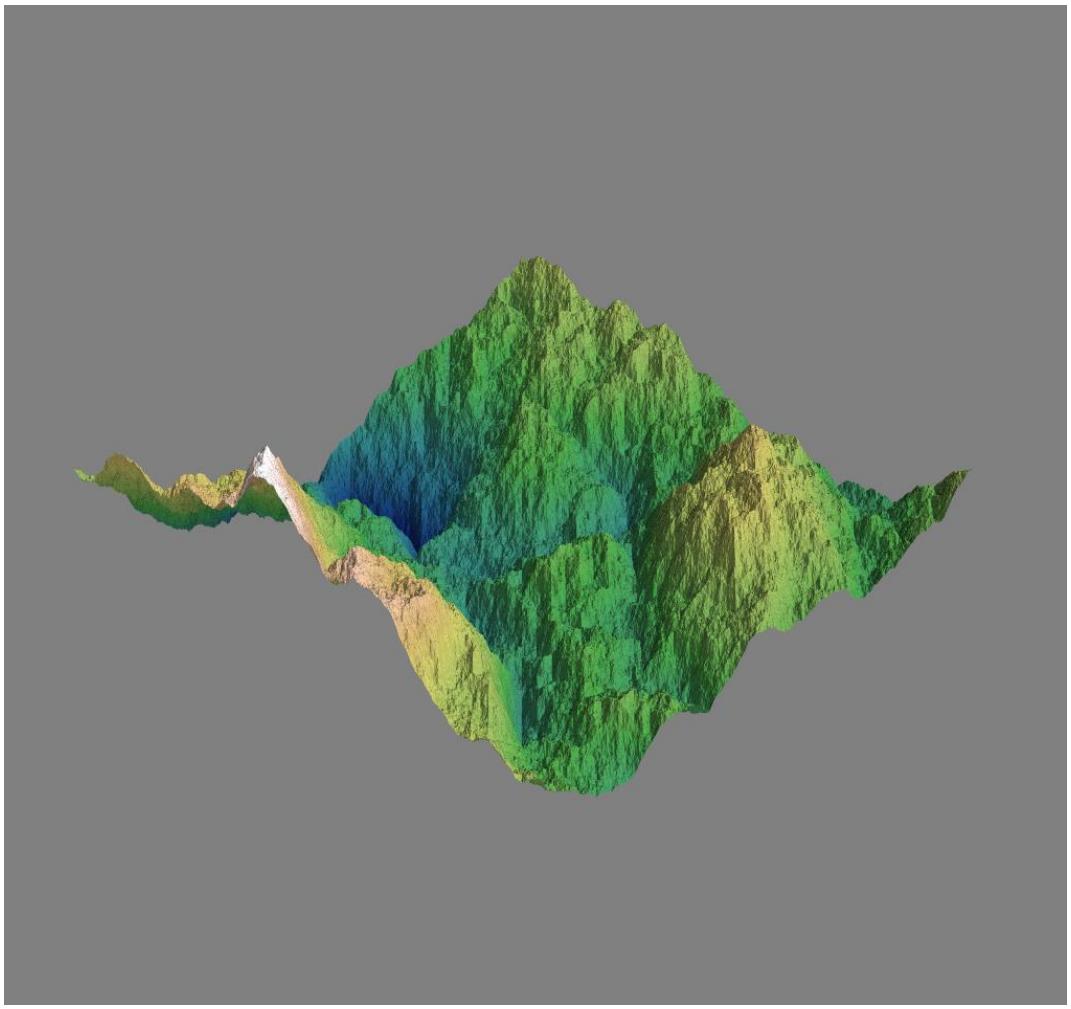


**Fig :** mountain landscape with level-12

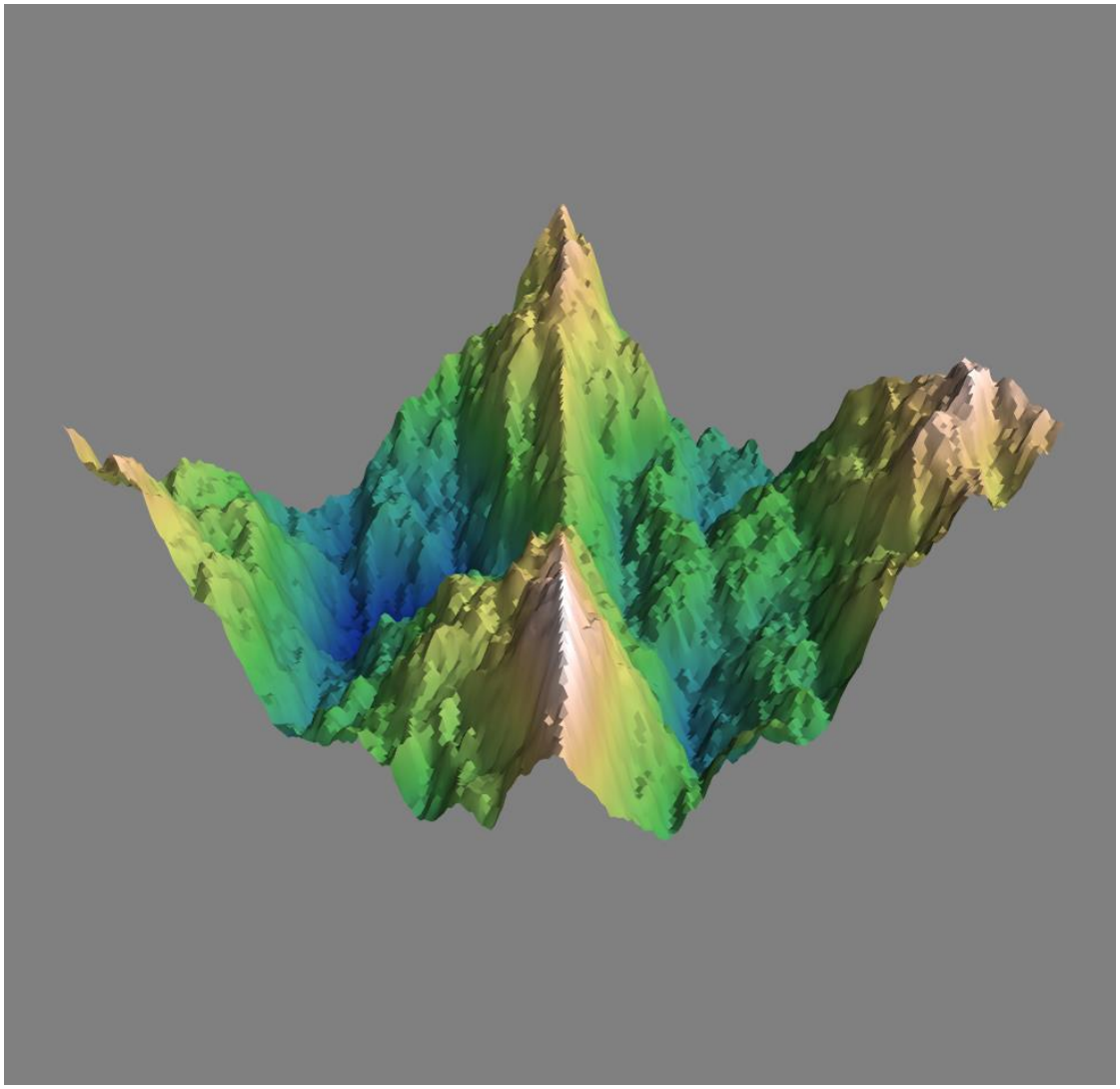


**Fig :** mountain landscape with level-8

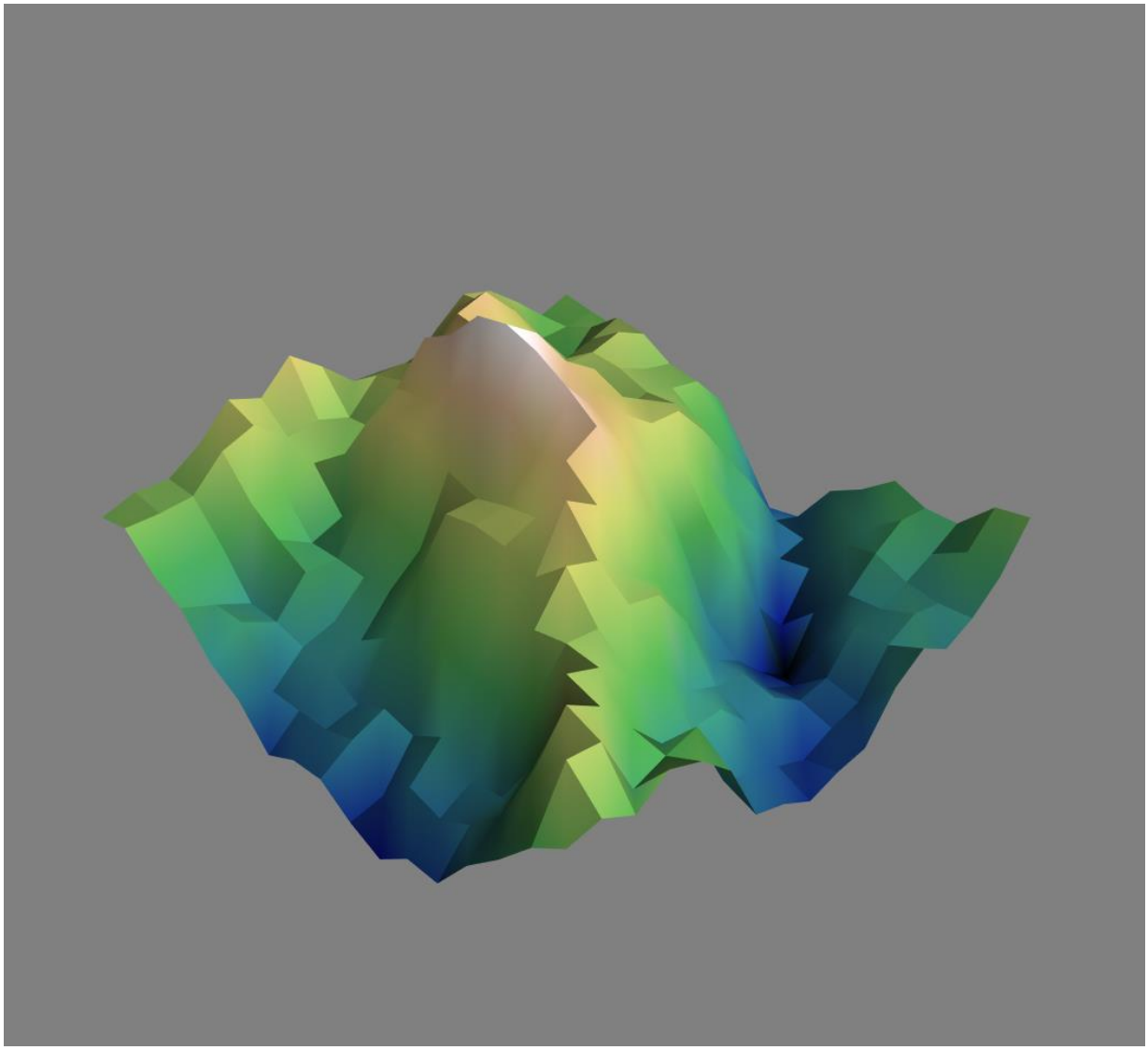


**Fig :** mountain landscape with level-5