

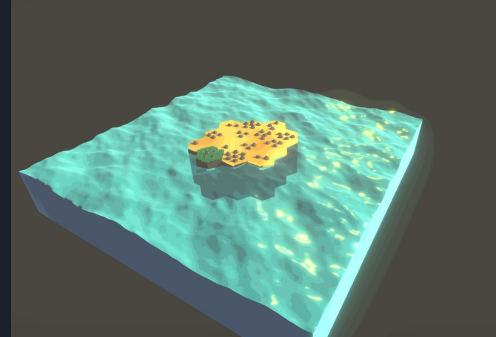
A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green. They are positioned diagonally, with the blue one partially covering the green one.

Procedural Map Generation

By Pablo Galve

What is procedural map generation?

- A map generated by a computer, not a human



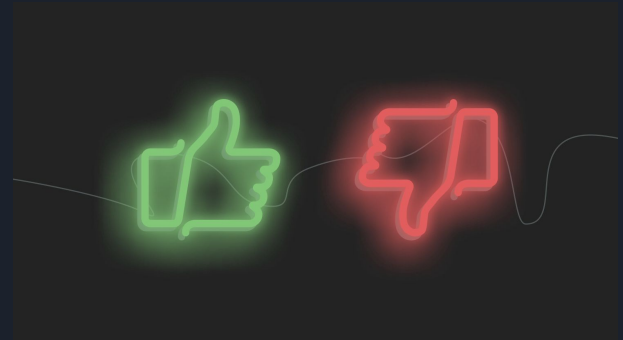
Why to use procedural map generation?

- **Pros**

- Each map is mathematically unique
- Saves development time
- Monetization can dramatically increase
 - Users play for years
- Perfect for creating a lot of maps

- **Cons**

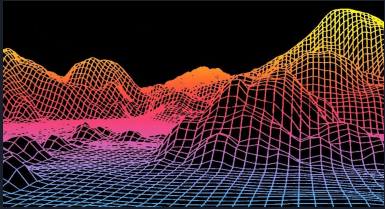
- Hard to make it visually unique
 - Computers are not creative
- You can't design it 100% as you want
- Bad for narrative games
- Bad for building only one map



Different type of maps

- Outdoor maps

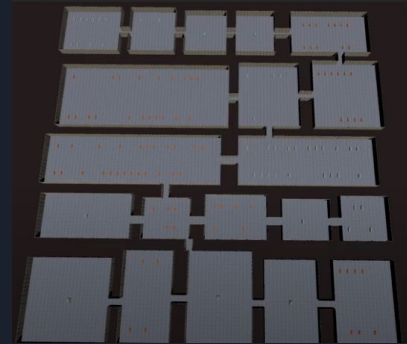
Using Perlin Noise



In this tutorial, we'll see how to generate outdoor maps

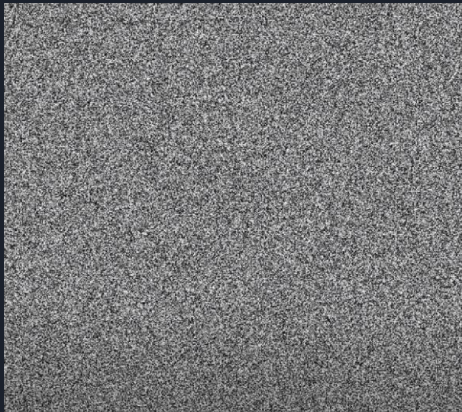
- Indoor / Dungeon maps

Explained in my website



What is Perlin Noise?

Normal Noise



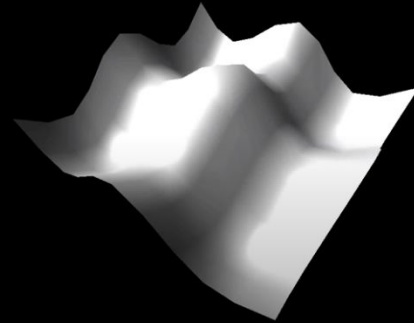
Perlin Noise



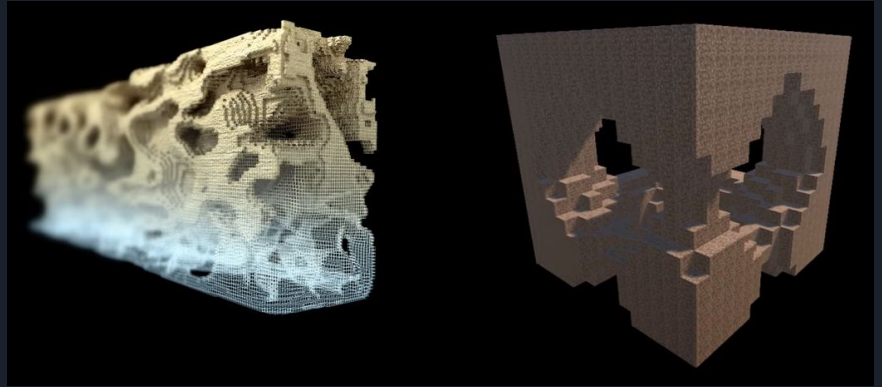
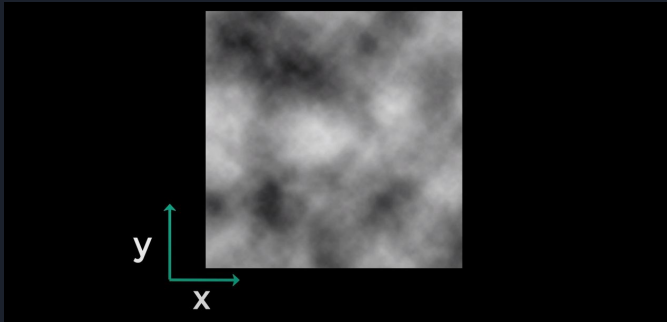
How to create infinite worlds

- We only need to store
 - Coordinates
 - A math formula
 - Changelog made by player
- Same seed = Same world

$$\text{height}(y) = \text{Cos}(x) + \text{Cos}(z)$$

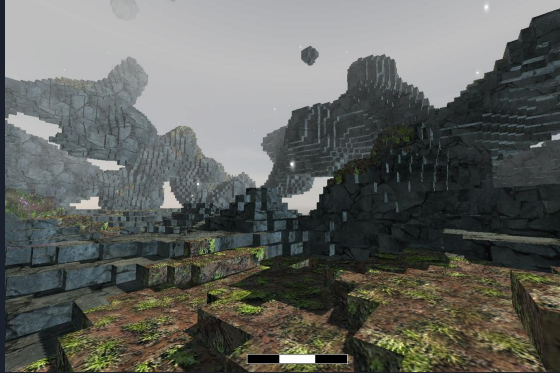


Perlin Noise - 1D, 2D & 3D



Games using procedural generation

Minecraft



Spore



No Man's Sky



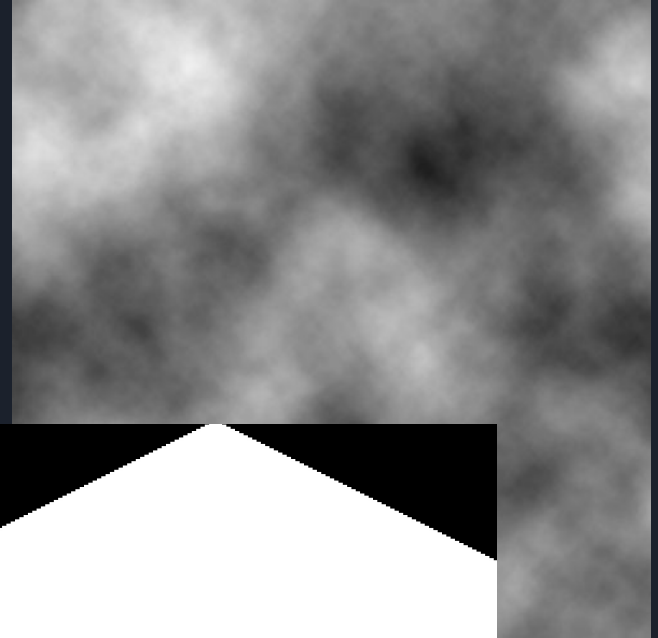


TODO 1: Generate Perlin Noise

- Create a FastNoise object
- Set noise type to “Perlin”
- Set frequency to 0.03
- Uncomment `myObject.SetSeed(seed)`

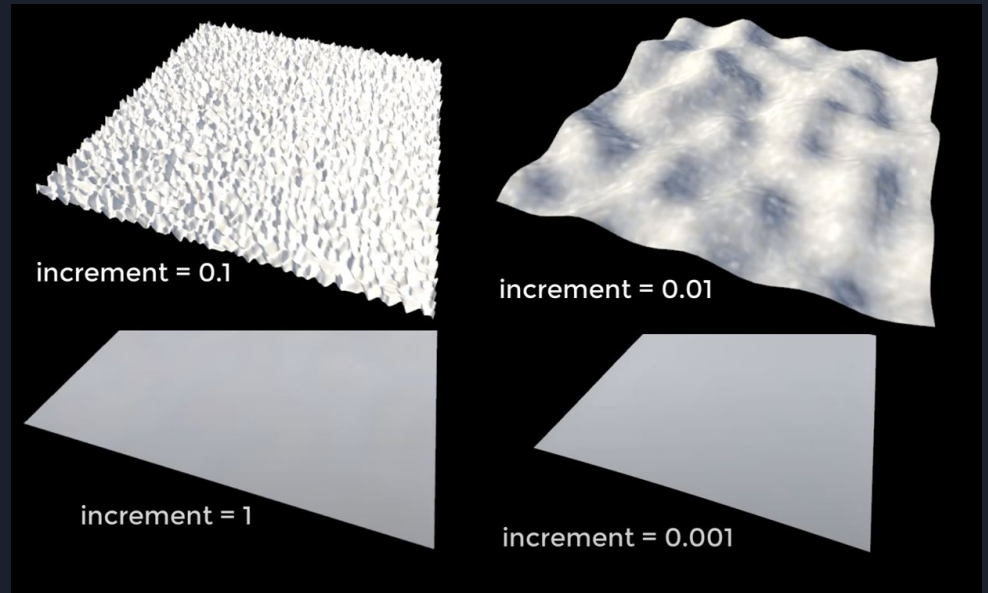
Output:

You won't see anything on screen yet



Why set frequency as 0.03?

- To avoid problems like that
- Frequency is the same as Increment



TODO 2.1: Store Perlin Noise

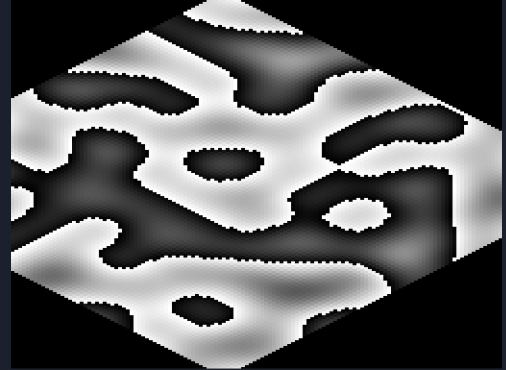
- We have a loop from 0-100 both in x and y
- Get noise at your desired coordinates
- Store values in “height_map” variable

Output:

You should have an output in console with values

And you will see that on screen

```
Noise on x: 99 y: 35 is: 0.219702
Noise on x: 99 y: 36 is: 0.186352
Noise on x: 99 y: 37 is: 0.145584
Noise on x: 99 y: 38 is: 0.099693
Noise on x: 99 y: 39 is: 0.050925
Noise on x: 99 y: 40 is: 0.001100
Noise on x: 99 y: 41 is: -0.049828
Noise on x: 99 y: 42 is: -0.105215
Noise on x: 99 y: 43 is: -0.166059
Noise on x: 99 y: 44 is: -0.230605
Noise on x: 99 y: 45 is: -0.295342
Noise on x: 99 y: 46 is: -0.355876
```



TODO 2.2: Adjust Perlin Noise to 0-1

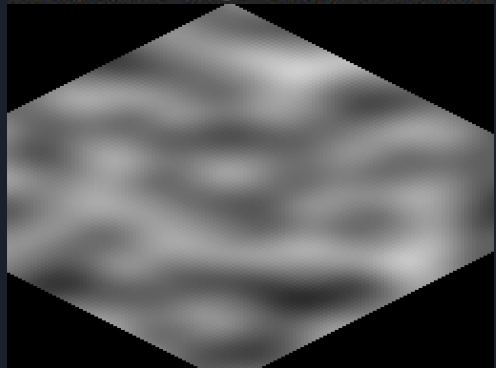
- We have a problem:
 - Some results are negative
- We need to fix that!
 - Use that formula: $(\text{Noise} + 1) * 0.5$

Output:

You will always have values between 0 and 1

You'll see that on screen

```
Noise on x: 99 y: 83 is: 0.584841
Noise on x: 99 y: 84 is: 0.592686
Noise on x: 99 y: 85 is: 0.592697
Noise on x: 99 y: 86 is: 0.585048
Noise on x: 99 y: 87 is: 0.570527
Noise on x: 99 y: 88 is: 0.550417
Noise on x: 99 y: 89 is: 0.526371
Noise on x: 99 y: 90 is: 0.500290
Noise on x: 99 y: 91 is: 0.474197
Noise on x: 99 y: 92 is: 0.450120
Noise on x: 99 y: 93 is: 0.429960
```



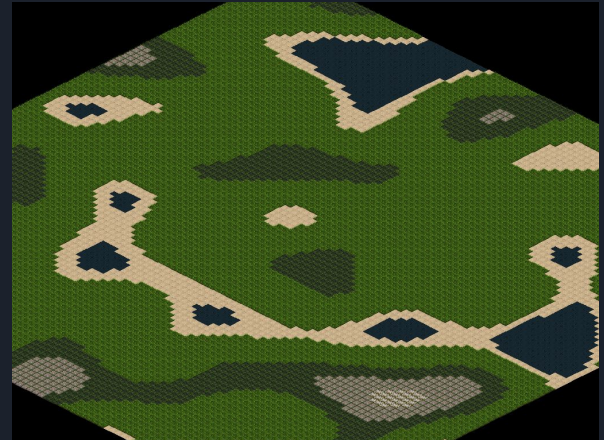
TODO 3: Draw Map from height_map

- 0 are low areas (valleys) and 1 are high areas (mountains)
- We have different textures:
 - Water, sand, grass, forest, mountain and snowy mountain
- Blit them depending on the value
- I used

```
<water min_value="0.00"/>  
<sand min_value="0.35"/>  
<grass min_value="0.4"/>  
<forest min_value="0.60"/>  
<mountain min_value="0.70"/>  
<snowy_mountain min_value="0.80"/>
```

Output:

You will see exactly this same map





TODO 4: Generate a random seed

- Seeds are usually current time in milliseconds
- An example of a seed is: 1587313622000

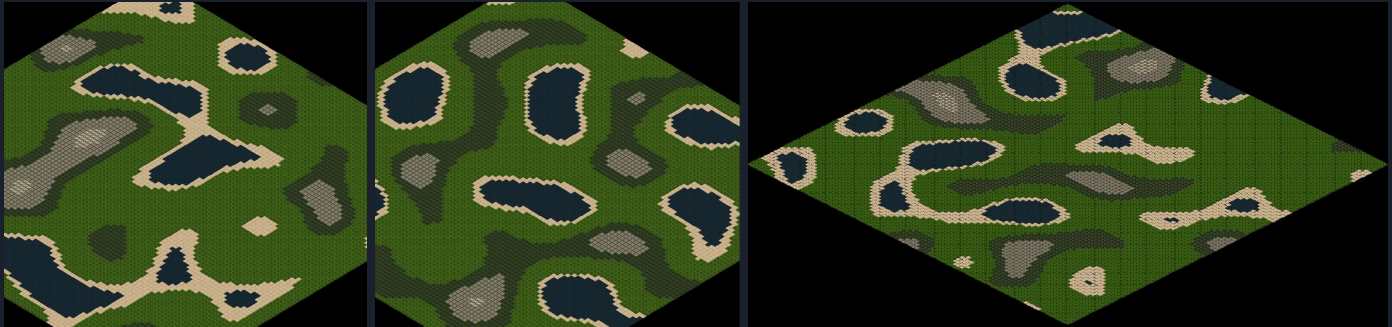
Output:

Nothing will change

TODO 5: Use the seed



- Pass the seed as an argument to generatePerlinNoise()



Output:

The map will be different each time or if you press space



Optional Homework - Infinite Map

- Occlusion Culling - Render only what is on screen
- Generate perlin noise for all the coordinates seen in screen
 - Not only from $[0][0]$ to $[100][100]$ like in the TODOs
- You should be able to render an infinite map with that