

The background is a low-poly 3D rendering of a mountainous landscape. In the foreground, there's a dirt path leading towards a cluster of green pine trees. Behind the trees are more mountain peaks, some covered in snow and others in green vegetation. The sky is a clear blue.

Somapah Worldscapes: Procedurally Generated World by Design

Team 1

Ragul, Xing Yi, James, Mark, Yingjie

01 Problem

02 Approach

03 Implementation

04 Results

05 Discussion

06 Conclusion



Procedural Generation

We want **infinite diversity** but
impossible to model everything



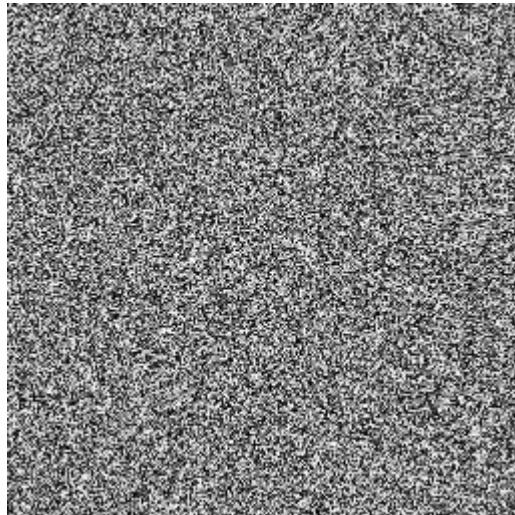
01 Problem

But... why exactly?

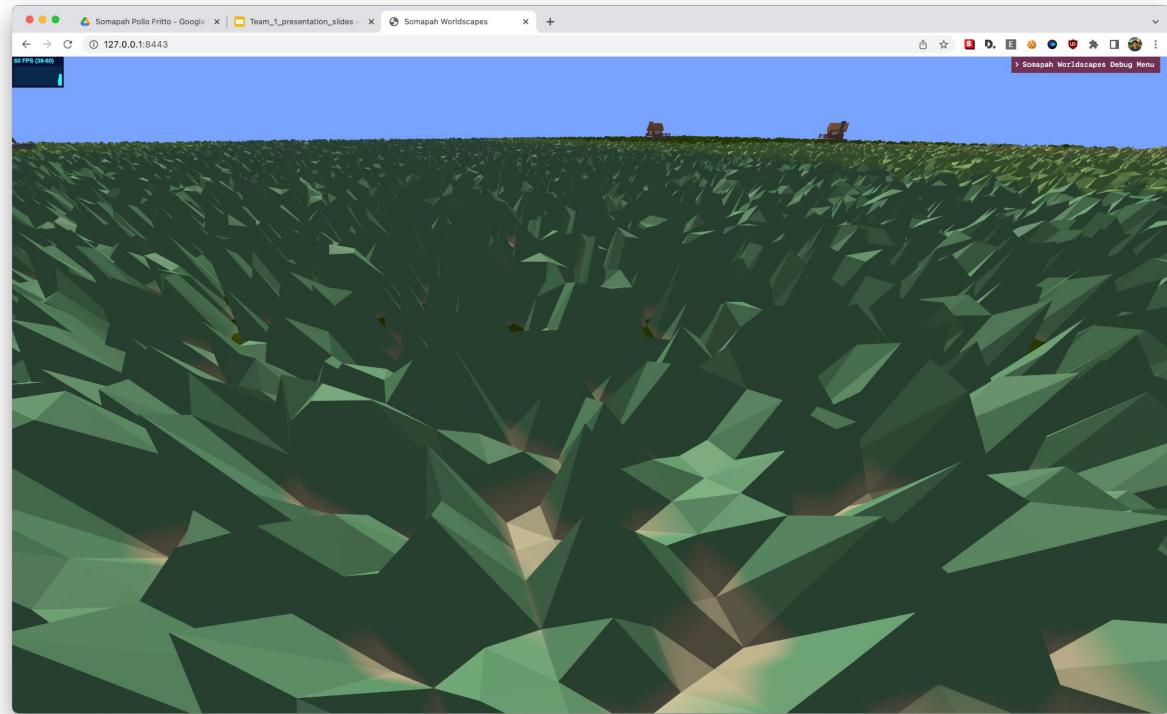


01 Problem

NAÏVE

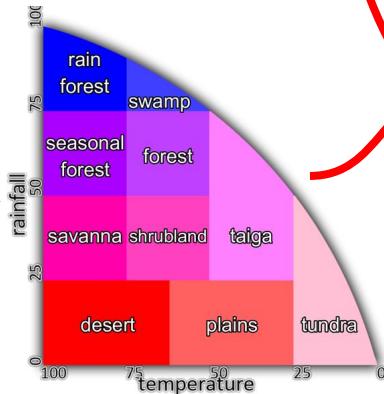
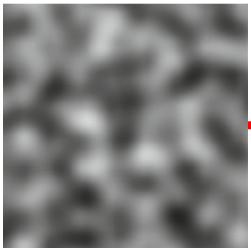
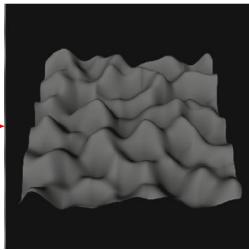
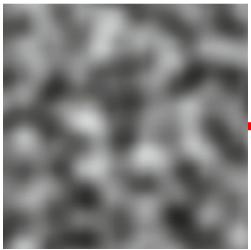


Uniform Noise



Looks like 💩

02 Approach



**Infinite World with
distinct regions!**

Layers of
Gradient Noise

Height Map +
Environment

02 Approach

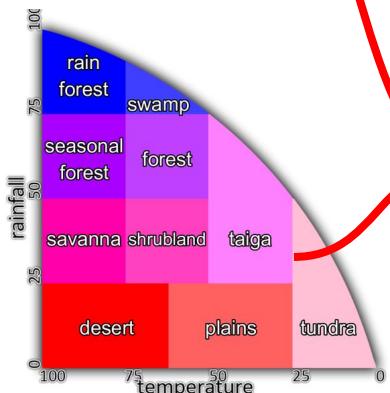
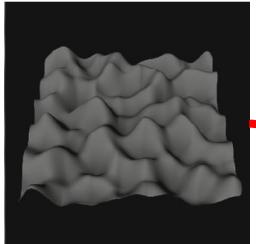
```

361 noise.seed(PARAMETERS.world_seed)
362 for (let y = 0; y < PARAMETERS.chunk_size; y++) {
363   for (let x = 0; x < PARAMETERS.chunk_size; x++) {
364     const j = 2 * (y * PARAMETERS.chunk_size + x)
365     const temp = (noise.simplex2((x0 + x) / 512, (z0 + y) / 512) + 1) / 2
366     let rain = (noise.simplex2((x0 + x) / 256, (z0 + y) / 256) + 1) / 2
367     rain = Math.min(rain, 0.99 - temp)
368     uv[j] = temp
369     uv[j + 1] = rain
370
371     const i = 3 * (y * PARAMETERS.chunk_size + x)
372     let h =
373       noise.simplex2((x0 + x) / 8, (z0 + y) / 8) * (rain + 0.3) +
374       noise.simplex2((x0 + x) / 128, (z0 + y) / 128) * 4 +
375       Math.min(32, noise.simplex2((x0 + x) / 768, (z0 + y) / 768) * 32 + 16)
376
377     const m = Math.abs(noise.perlin2((x0 + x) / 512, (z0 + y) / 512))
378     const mt = Math.abs(noise.simplex2((x0 + x) / 512, (z0 + y) / 512) * 0.2)
379     if (m < mt && h > 24) {
380       h *= (1 + (mt - m) * h / 5)
381     }
382
383     vertices[i + 1] = Math.max(0, h)
384
385     const r =
386       Math.abs(noise.simplex2((x0 + x) / 96, (z0 + y) / 96)) <
387       Math.min(0.2, Math.abs((3 - h) / 8))
388
389     if (h < 0) {
390       uv[j] = 1
391       uv[j + 1] = 1
392     }
393     if (r && vertices[i + 1] < 3) {
394       vertices[i + 1] *= 0.2
395       uv[j] = 1
396       uv[j + 1] = 1
397     }

```

*Admittedly,
this is more art
than science*

02 Approach



```
8 phongMaterial.onBeforeCompile = function (materialInfo) {
9   materialInfo.vertexUvs = true
10  materialInfo.uvsVertexOnly = false
11  materialInfo.vertexShader = materialInfo.vertexShader.replace(
12    'varying vec3 vViewPosition;',
13
14    'varying vec3 vViewPosition;
15    varying vec3 vPosition;
16
17  ).replace(
18    '',
19    'Position = position;
20  '
21  )
22  materialInfo.fragmentShader = materialInfo.fragmentShader.replace(
23    'uniform float opacity;',
24
25    'uniform float opacity;
26    varying vec3 vPosition;
27
28  ).replace(
29    '#include <map_fragment>',
30
31    if (vUv.x > 0.8 && vUv.y > 0.8) {
32      vec4 sampledDiffuseColor = vec4(0.1, 0.2, 0.7 - vPosition.y, 1.0);
33      diffuseColor *= sampledDiffuseColor;
34    } else {
35      vec4 sampledDiffuseColor = texture2D( map, vec2(vUv.x, vPosition.y / 64.0));
36      diffuseColor *= sampledDiffuseColor;
37    }
38  )
39}
```



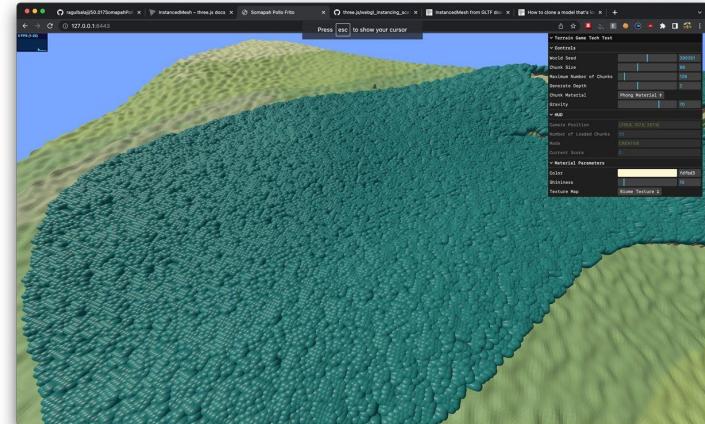
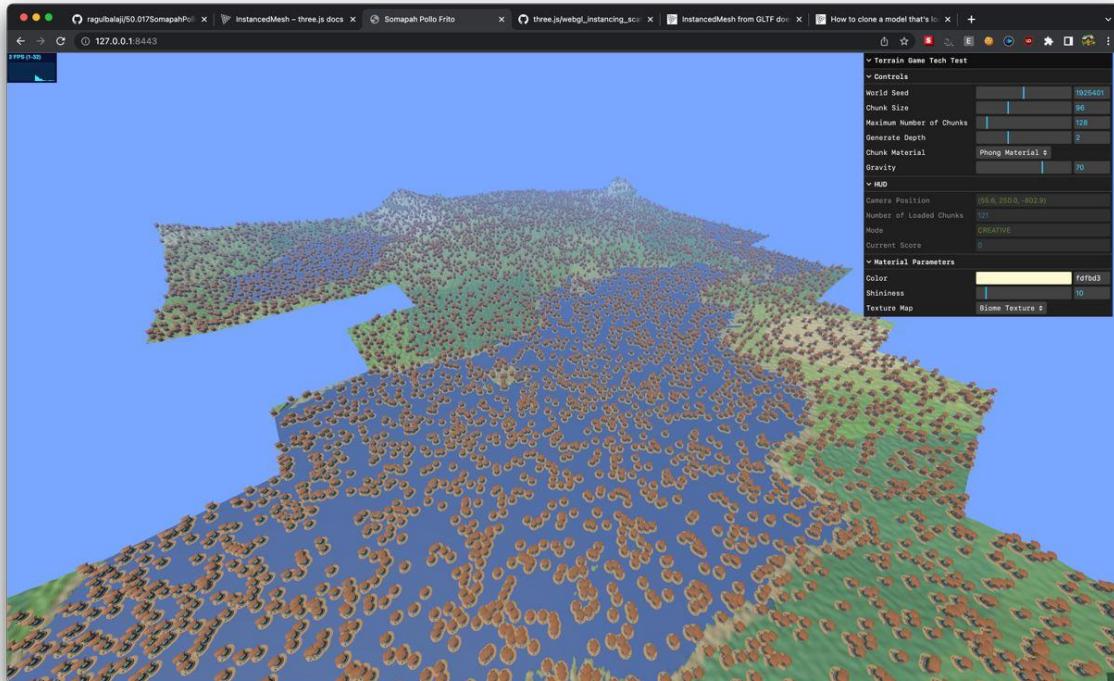
**Infinite World with
distinct regions!**

Height Map +
Environment

Custom Shader
With Phong +
Lighting Tricks

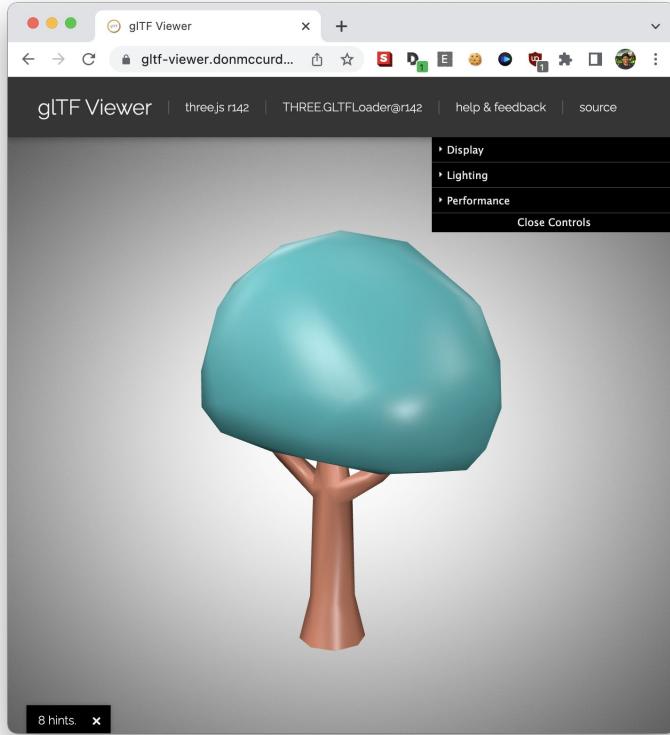
02 Approach

NAÏVE Mesh Loading



Slow as 💩
Fat (RAM) as 💩

02 Approach

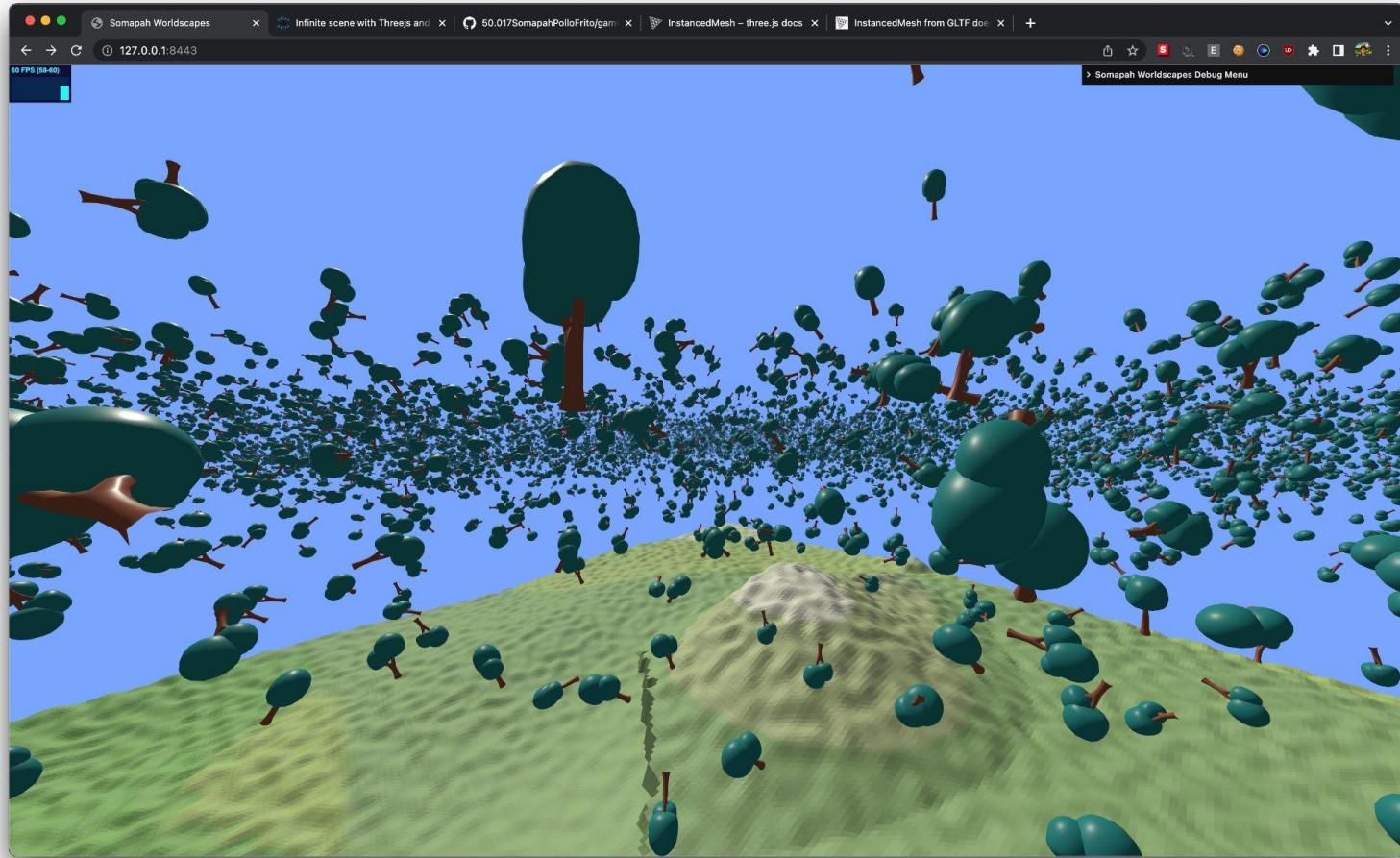


A screenshot of a web browser displaying the `three.js` documentation for the `GLTFLoader`. The page explains what GLTF is, how it works, and provides usage examples. A red arrow points from the text "A loader for glTF 2.0 resources." towards the InstancedMesh section on the right.

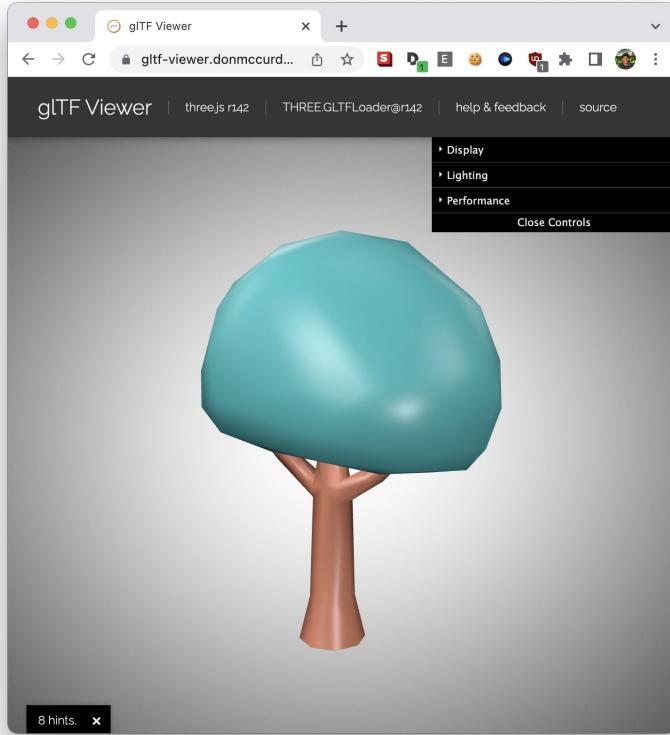
A screenshot of a web browser displaying the `three.js` documentation for the `InstancedMesh` class. It describes its purpose for efficient rendering of many objects with the same geometry. A red arrow points from the "A special version of Mesh with instanced rendering support." text towards the "Constructor" section on the right.

+ Custom Code to Create & Manage these **Instanced** Meshes

02 Approach



02 Approach



A screenshot of a web browser displaying the 'GLTFLoader' documentation from threejs.org. The page explains what GLTF is, how it works, and its benefits. It also covers how to use GLTFLoader and provides links to examples and the constructor.

A screenshot of a web browser displaying the 'InstancedMesh' documentation from threejs.org. It describes what InstancedMesh is, its purpose, and how it improves rendering performance. It includes examples and a constructor section.

+ Custom Code to Create & Manage these **Instanced** Meshes

1 draw call per 15000 instances
Small RAM and VRAM Footprint

02 Approach

georgealways/lil-gui

Makes a floating panel for controllers on the web.
Works as a drop-in replacement for dat.gui in most projects.



3
Contributors

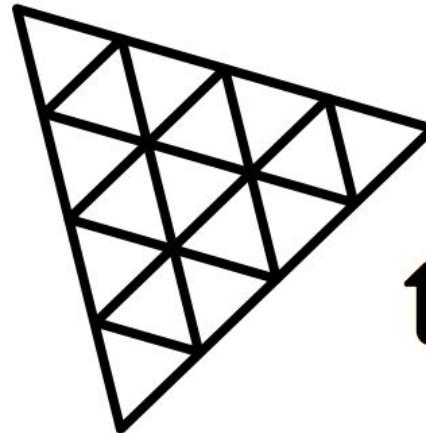
2k
Used by

408
Stars

14
Forks



A screenshot of a game asset pack page titled "MEDIEVAL BUILDER PACK". The page shows a colorful 2D map of a medieval town with various buildings, roads, and terrain pieces. Below the map, there's a "Download Now" button and a "Name your own price" field. A brief description states it's a bundle of assets for Real Time Strategy and Empire building needs. A "Contents" section lists items like buildings, walls, roads, tile variations, water tiles, and scenery. Two small preview images show a grid of tiles and a hillside with trees.



three.js

60 FPS (59-61)

8 MS (7-13)

386 MB (13-641)

243 x (60-452)

Perlin & Simplex Noise Library



03 Implementation

Click to Focus!

Toggle Mode: Press 'C'

Move: W A S D

Jump: SPACE

Look: MOUSE

Quit: ESC

**Additional Controls in
CREATIVE mode:**

'Q' to go up

'E' to go down

03 Implementation

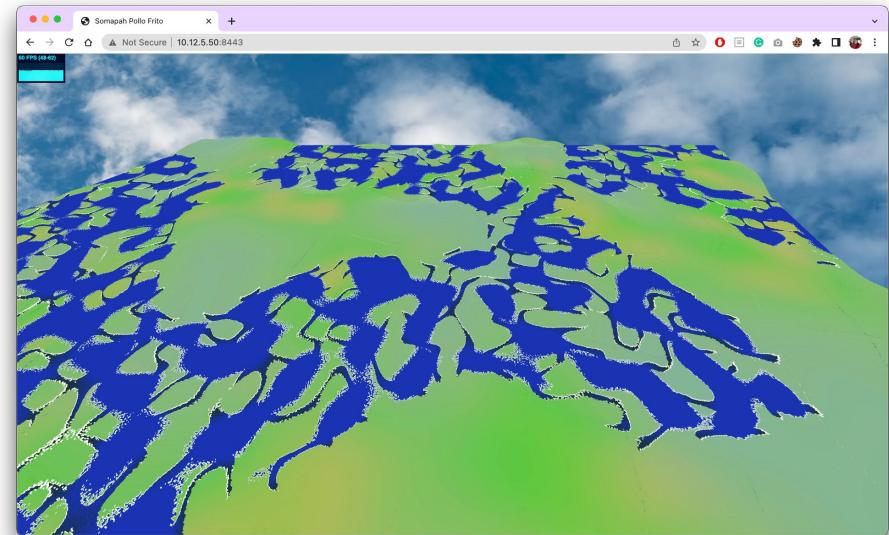
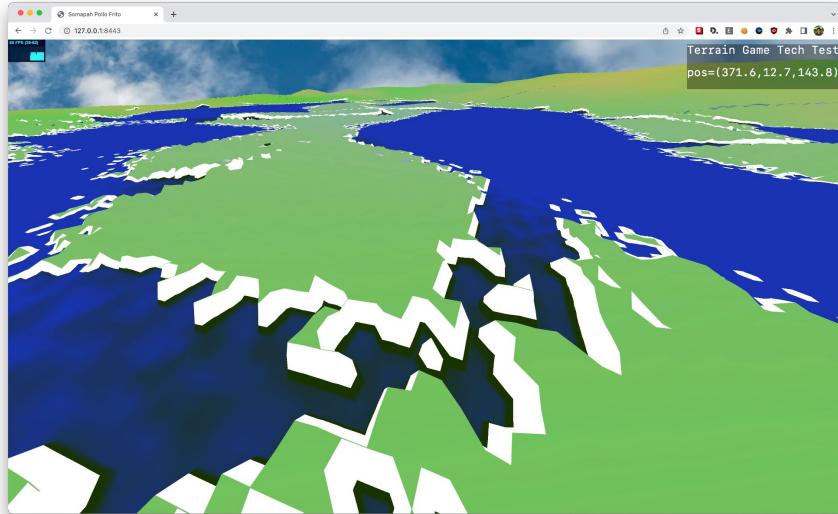
▼ Somapah Worldscapes Debug Menu		
▼ Controls		
World Seed		2503984
Chunk Size		96
Max. Number of Chunks		150
Generation Depth		3
Chunk Material	Phong Material	↳
Gravity		70
Day-Night Cycle Speed		0
Base Sky Color		79a6ff
Fog Density		0.001
▼ Noise		
Chunk X Offset		6969
Chunk Z Offset		6969
Temperature Divisor		512
Rain Divisor		256
Mountain Divisor		512
Mountain Threshold		24
Trotty Multiplier		123
Tree Choice X Multiplier		4242
Tree Choice Z Multiplier		6969
Tree A Threshold		0
Tree B Threshold		0.4
▼ Lights		
Ambient Color		404040
Ambient Intensity		0.5
Directional Color		fdfbd3
Directional Intensity		0.8
Directional Angle		90
Moon Intensity		0.15
▼ HUD		
Camera Position	(0.0, 0.0, 0.0)	
Number of Loaded Chunks	0	
Mode	CREATIVE	
Current Score	0	
▼ Material Parameters		
Color		fdfbd3
Shininess		10
Texture Map	Biome Texture	↳

Tweak 30+*
Parameters
with live reload!

*With a few extra material-specific parameters!

03 Implementation

Previously...



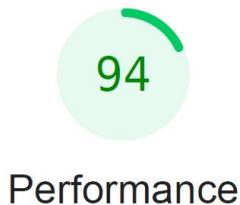
04 Results

Now!



04 Results

Performance (Google Lighthouse)



Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator.](#)



04 Results

Performance (Google Lighthouse)

METRICS

Expand view

- First Contentful Paint

0.2 s

- Time to Interactive

1.8 s

- Speed Index

1.5 s

- Total Blocking Time

160 ms

- Largest Contentful Paint

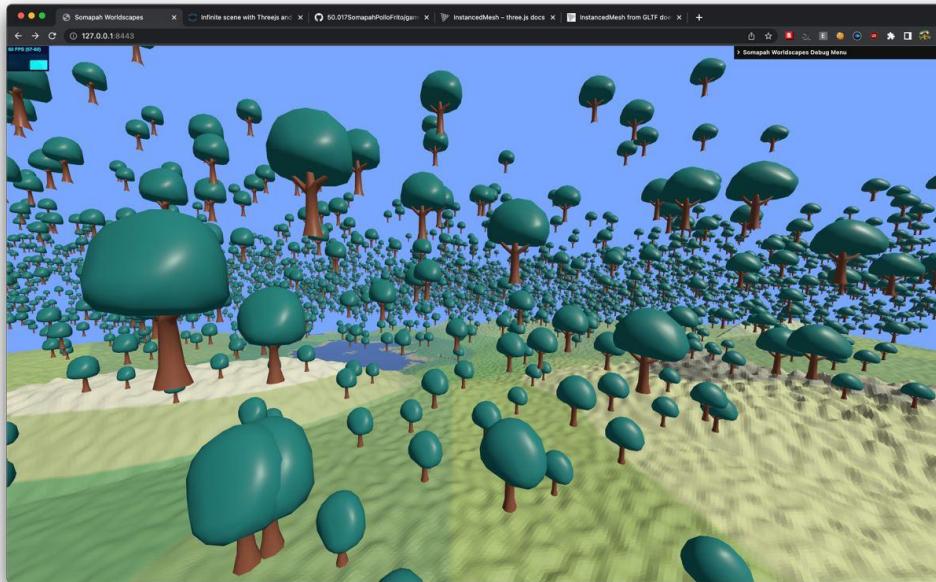
0.2 s

- Cumulative Layout Shift

0.003

04 Results

100s of Millions of Vertices!
Millions of Meshes!
Fewer than 200 draw calls.
Up to 120 FPS (VSYNC limited)



04 Results



See the Final Results at:

<http://ragulbalaji.com/worldscapes/>



**Interesting landscapes to explore.
High replay value for games.
Sparks creativity in players.**

Tuning random functions is hard.

**Even after tuning results may not be perfect.
World is not destructible, hard to store deltas.**

Learnings



Generating Meshes
Writing Shaders for Texturing
Phong Shading Model
Instanced Meshes
3D modelling and remeshing
Memory Management
Performance Optimisations
Random Noise Voodoo Magic

Procedural Generation

1. Using **noise** layers to generate terrain
2. Using **shaders** to generate environments
3. Using **instance meshes** to add trees & homes

Most importantly



A 3D rendering of a landscape featuring rolling green hills covered in small trees. A winding blue river flows through the valley. In the distance, a small town with several buildings is visible at the base of the hills under a clear blue sky.

Thank You for Visiting!

Any Questions?

References

Detailed Explanation on Random Biome Generation Using Perlin Noise:
<https://gamedev.stackexchange.com/a/186197>

Making Maps With Noise Functions:
<https://www.redblobgames.com/maps/terrain-from-noise/>

Three.js Library: <https://threejs.org/>

lil-gui Library: <https://lil-gui.georgealways.com/>

KayKit Medieval Builder Pack:
<https://kaylousberg.itch.io/kaykit-medieval-builder-pack>