Somapah Worldscapes:
Procedurally Generated World by Design

Team 1

Ragul, Xing Yi, James, Mark, Yingjie

Procedural Generation

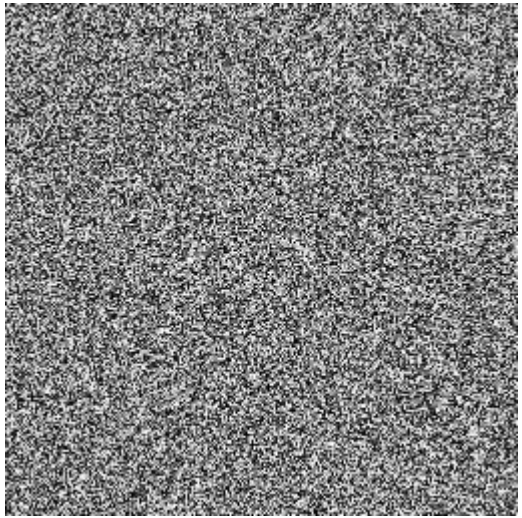
We want **infinite diversity** but **impossible** to model everything

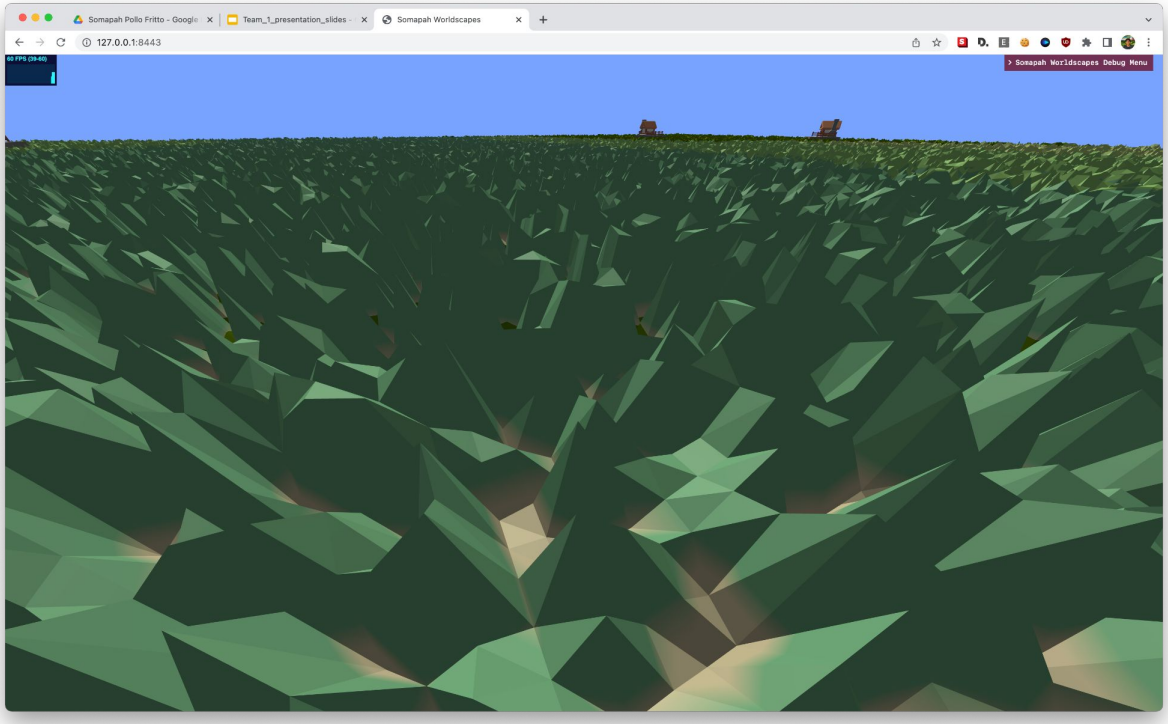
But... **why exactly?**

+

01 Problem

NAÏVE

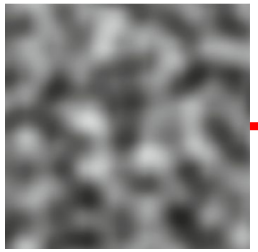


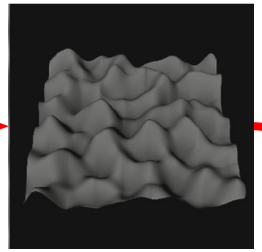


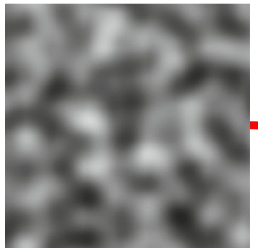
Uniform Noise

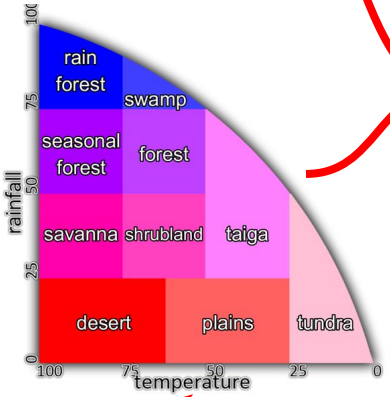
Looks like 💩

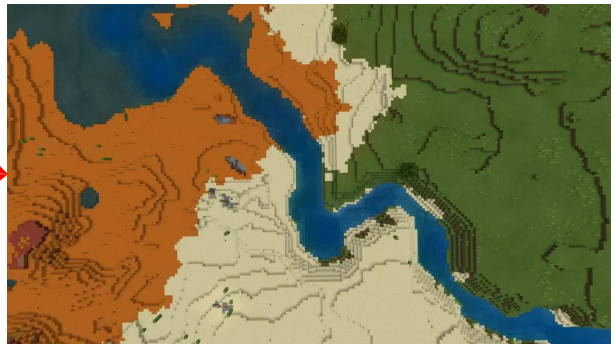
02 Approach

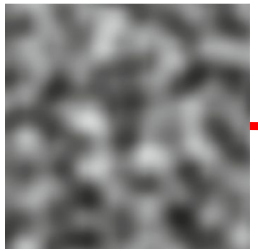












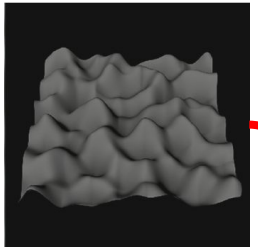
Layers of Gradient Noise

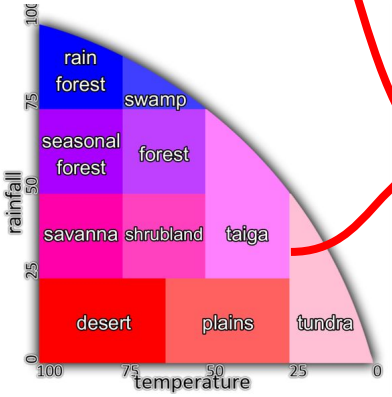
Height Map + Environment

Infinite World with distinct regions!

```
noise.seed(PARAMETERS.world_seed)
for (let y = 0; y < PARAMETERS.chunk_size; y++) {
  for (let x = 0; x < PARAMETERS.chunk_size; x++) {
    const j = 2 * (y * PARAMETERS.chunk_size + x)
    const temp = (noise.simplex2((x0 + x) / 512, (z0 + y) / 512) + 1) / 2
    let rain = (noise.simplex2((x0 + x) / 256, (z0 + y) / 256) + 1) / 2
    rain = Math.min(rain, 0.99 - temp)
    uv[j] = temp
    uv[j + 1] = rain

    const i = 3 * (y * PARAMETERS.chunk_size + x)
    let h =
      noise.simplex2((x0 + x) / 8, (z0 + y) / 8) * (rain + 0.3) +
      noise.simplex2((x0 + x) / 128, (z0 + y) / 128) * 4 +
      Math.min(32, noise.simplex2((x0 + x) / 768, (z0 + y) / 768) * 32 + 16)

    const m = Math.abs(noise.perlin2((x0 + x) / 512, (z0 + y) / 512))
    const mt = Math.abs(noise.simplex2((x0 + x) / 512, (z0 + y) / 512) * 0.2)
    if (m < mt && h > 24) {
      h *= (1 + (mt - m) * h / 5)
    }

    vertices[i + 1] = Math.max(0, h)

    const r =
      Math.abs(noise.simplex2((x0 + x) / 96, (z0 + y) / 96)) <
      Math.min(0.2, Math.abs((3 - h) / 8))

    if (h < 0) {
      uv[j] = 1
      uv[j + 1] = 1
    }
    if (r && vertices[i + 1] < 3) {
      vertices[i + 1] *= 0.2
      uv[j] = 1
      uv[j + 1] = 1
    }
```

Admittedly, this is more art than science

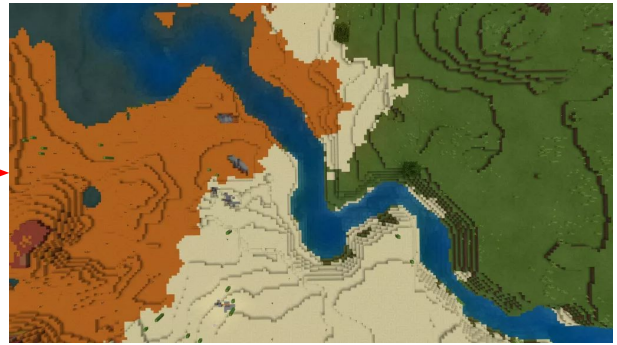




Height Map + Environment

```
phongMaterial.onBeforeCompile = function (materialInfo) {
  materialInfo.vertexUvs = true
  materialInfo.uvsVertexOnly = false
  materialInfo.vertexShader = materialInfo.vertexShader.replace(
    'varying vec3 vViewPosition;',
    `
    varying vec3 vViewPosition;
    varying vec3 vPosition;
    `
  ).replace(
    '}',
    `vPosition = position;
    }`
  )
  materialInfo.fragmentShader = materialInfo.fragmentShader.replace(
    'uniform float opacity;',
    `
    uniform float opacity;
    varying vec3 vPosition;
    `
  ).replace(
    '#include <map_fragment>',
    `
    if (vUv.x > 0.8 && vUv.y > 0.8) {
      vec4 sampledDiffuseColor = vec4(0.1, 0.2, 0.7 - vPosition.y, 1.0);
      diffuseColor *= sampledDiffuseColor;
    } else {
      vec4 sampledDiffuseColor = texture2D( map, vec2(vUv.x, vPosition.y / 64.0));
      diffuseColor *= sampledDiffuseColor;
    }
    `
  )
}
```

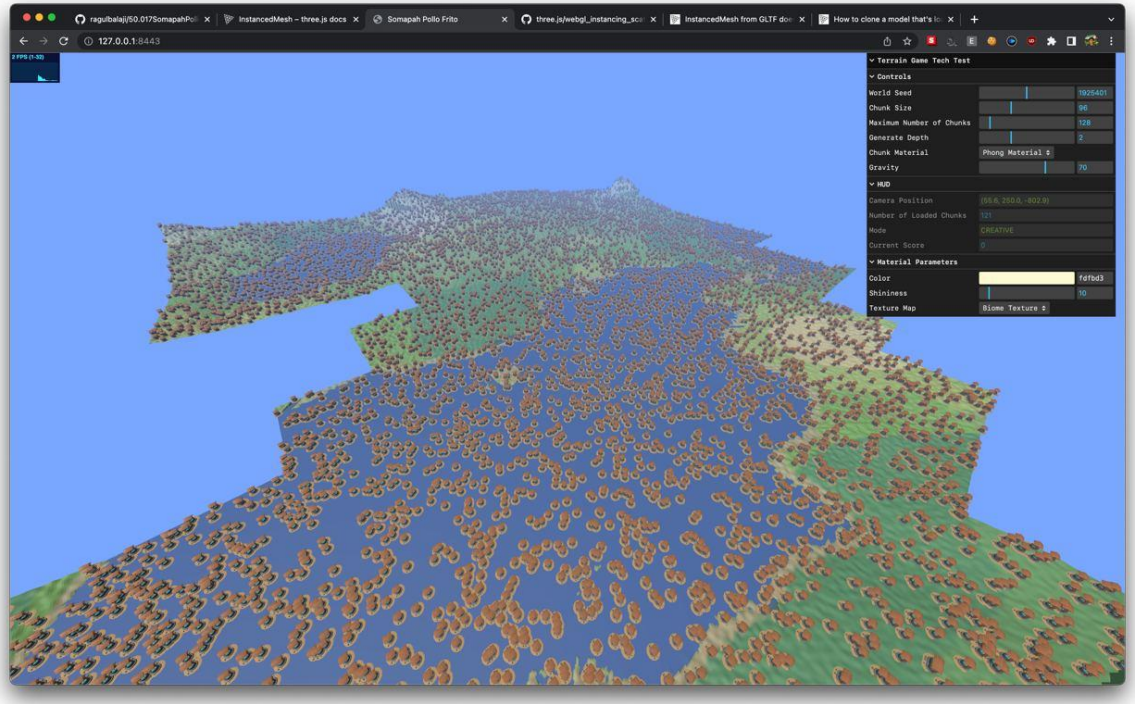
Custom Shader With Phong + Lighting Tricks

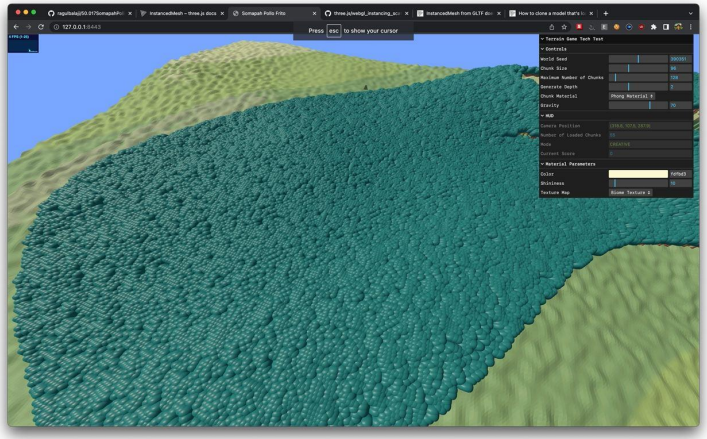


Infinite World with distinct regions!

02 Approach

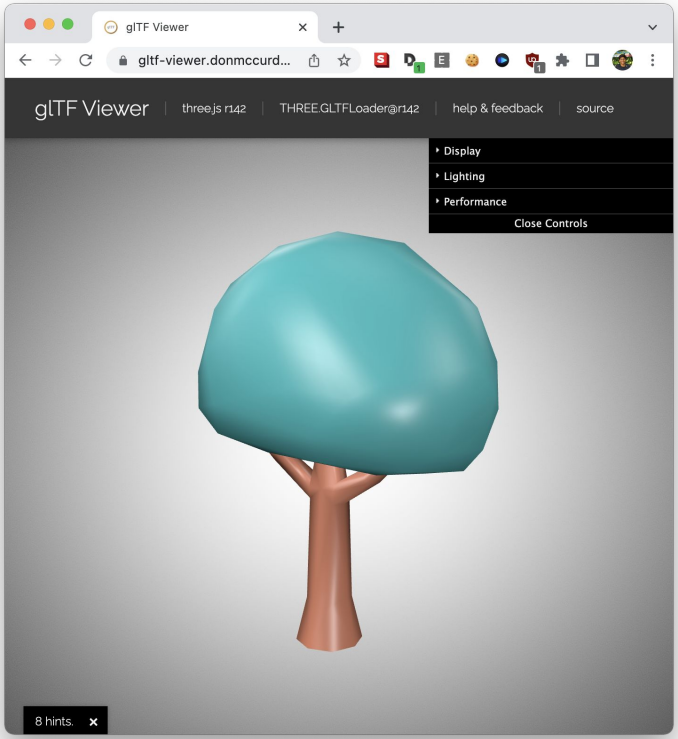
NAÏVE Mesh Loading

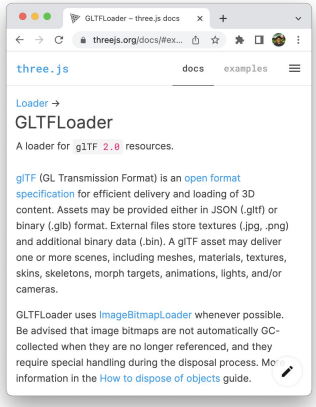


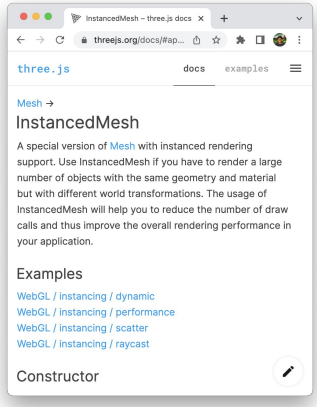


Slow as 💩
Fat (RAM) as 💩

02 Approach

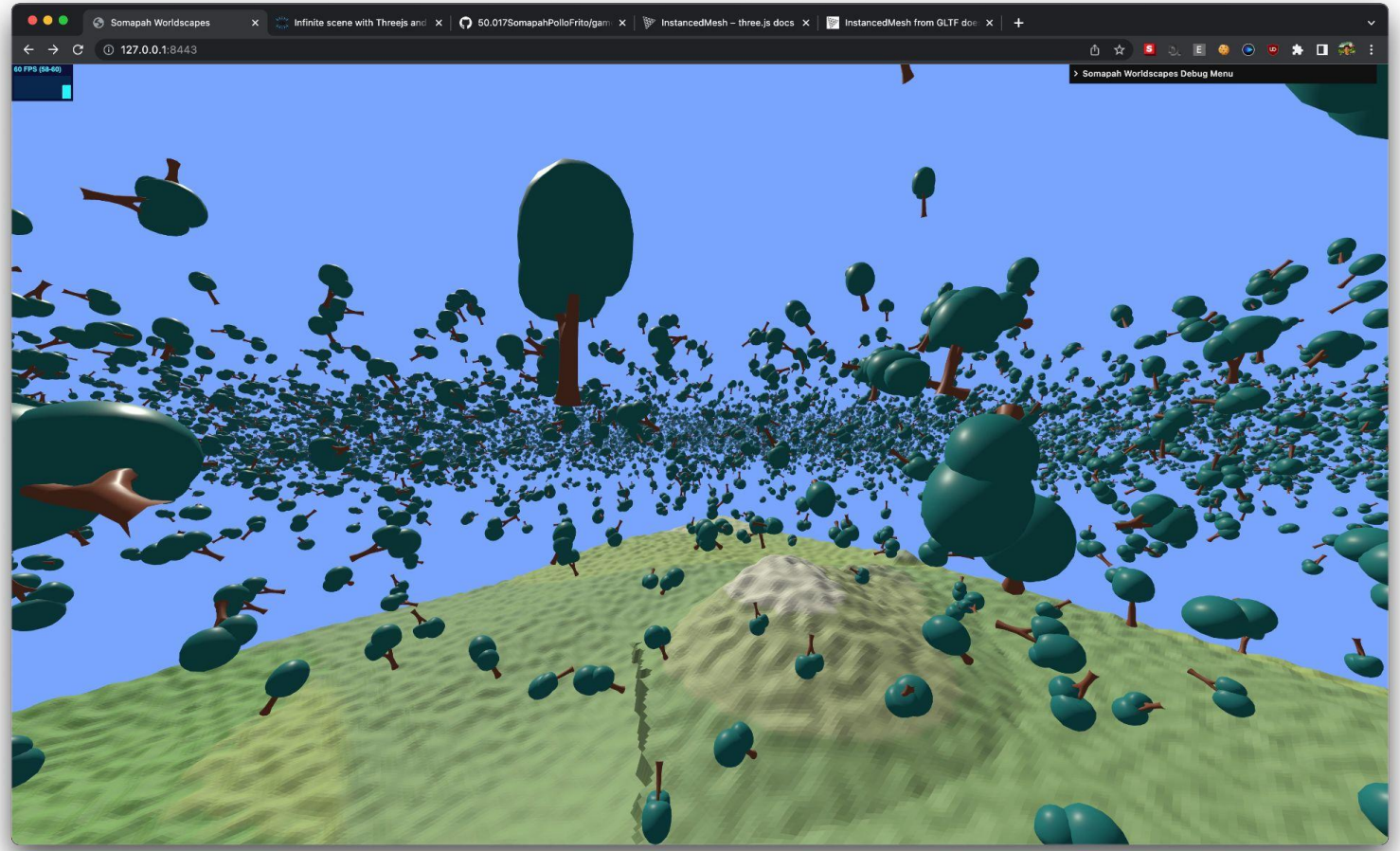




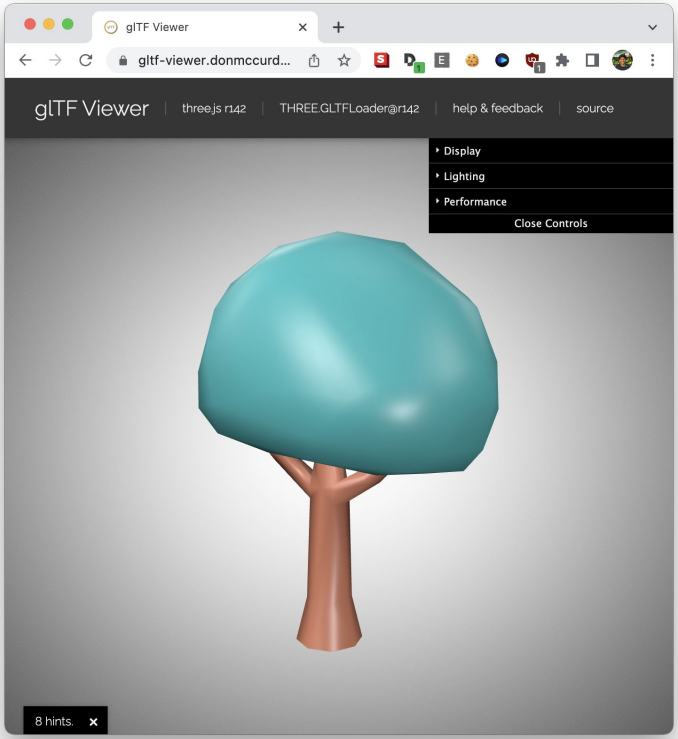


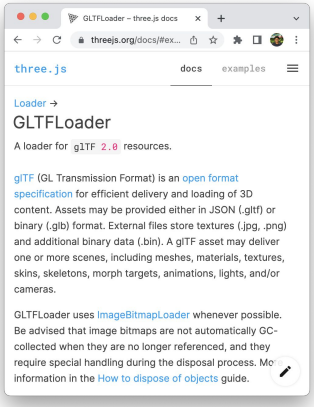
\+ Custom Code to Create & Manage these **Instanced** Meshes

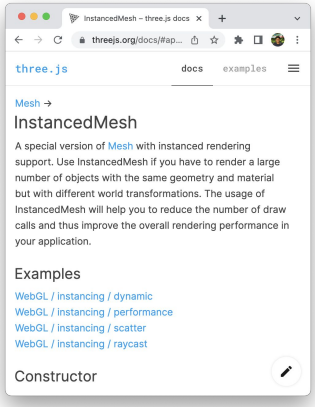
02 Approach


Somapah Worldscapes
Infinite scene with Threejs and
50.017SomapahPolloFrito/gam
InstancedMesh – three.js docs
InstancedMesh from GLTF doe
127.0.0.1:8443
60 FPS (58-60)
Somapah Worldscapes Debug Menu

02 Approach







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

1 draw call per 15000 instances

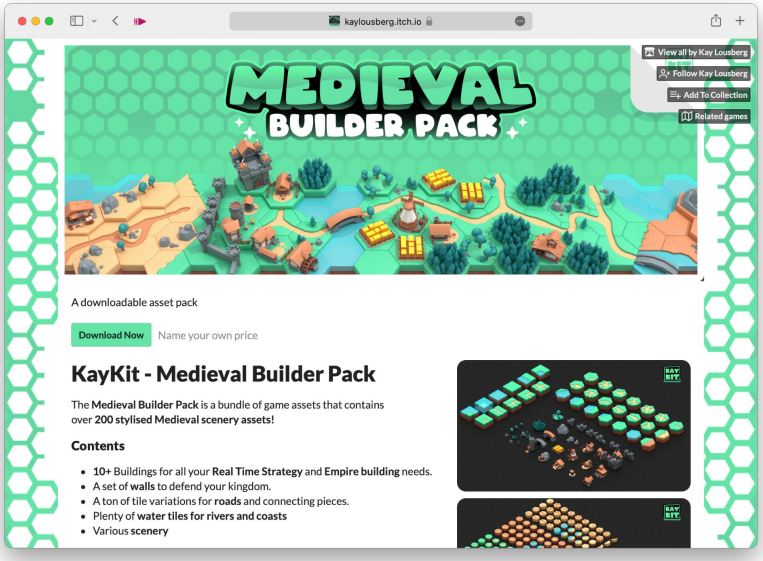
Small RAM and VRAM Footprint

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

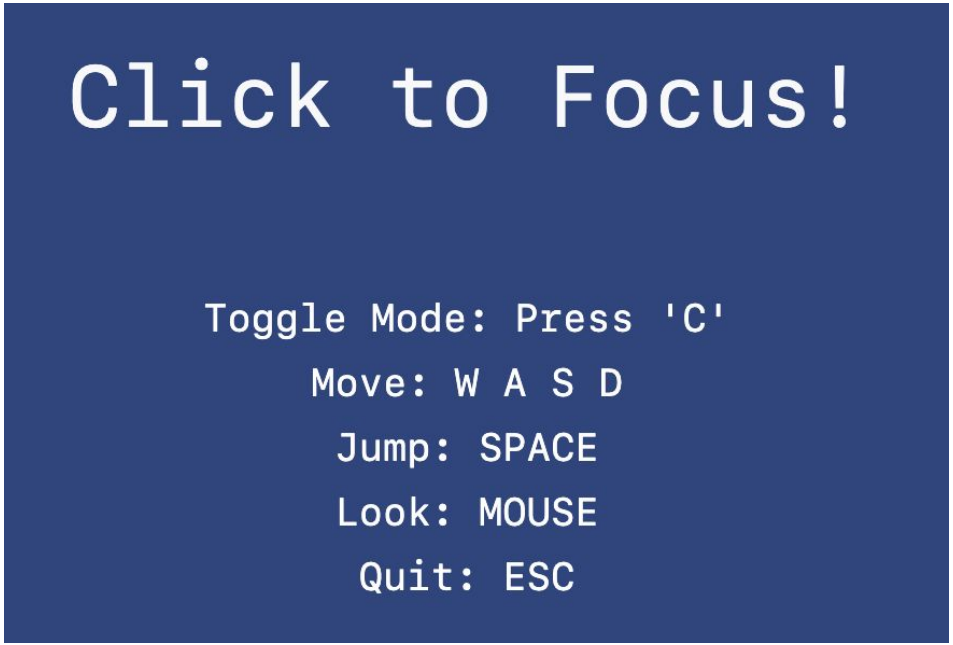


three.js

Perlin & Simplex Noise Library

HTML

03 Implementation



Additional Controls in CREATIVE mode:

‘Q’ to go up

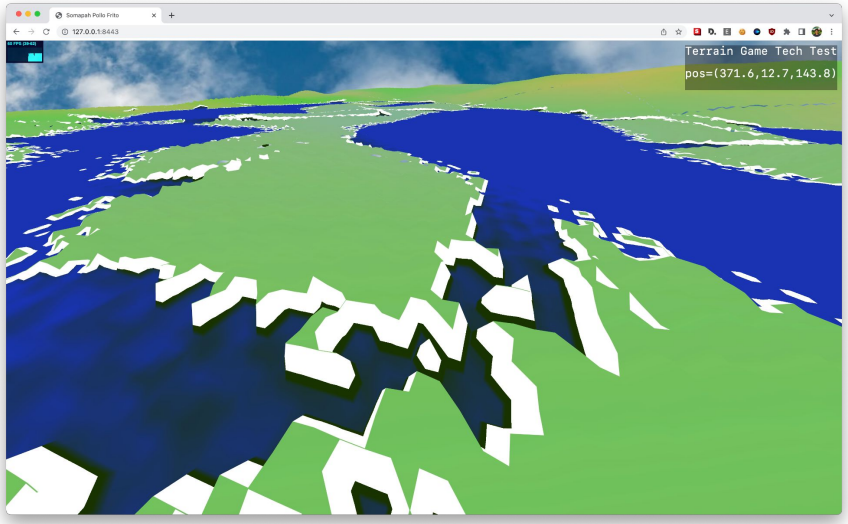
‘E’ to go down

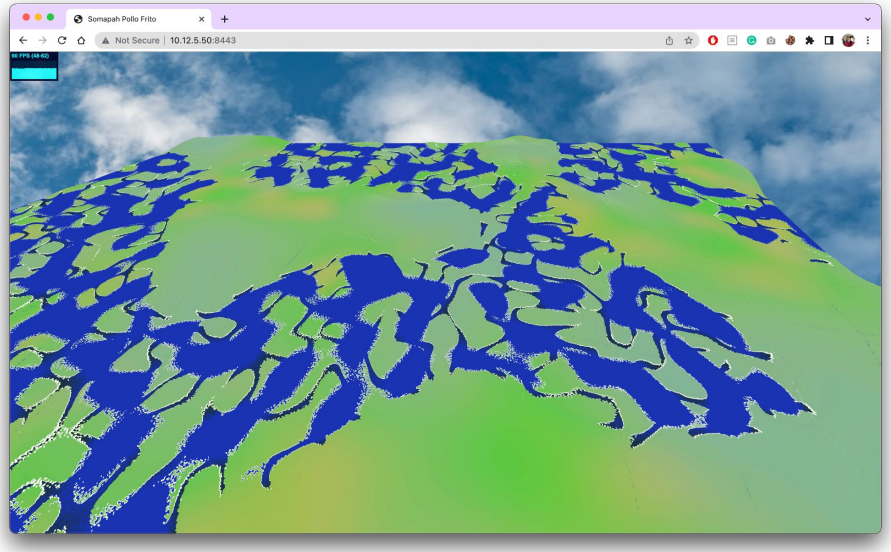
Tweak 30+* Parameters with live reload!

*With a few extra material-specific parameters!

03 Implementation

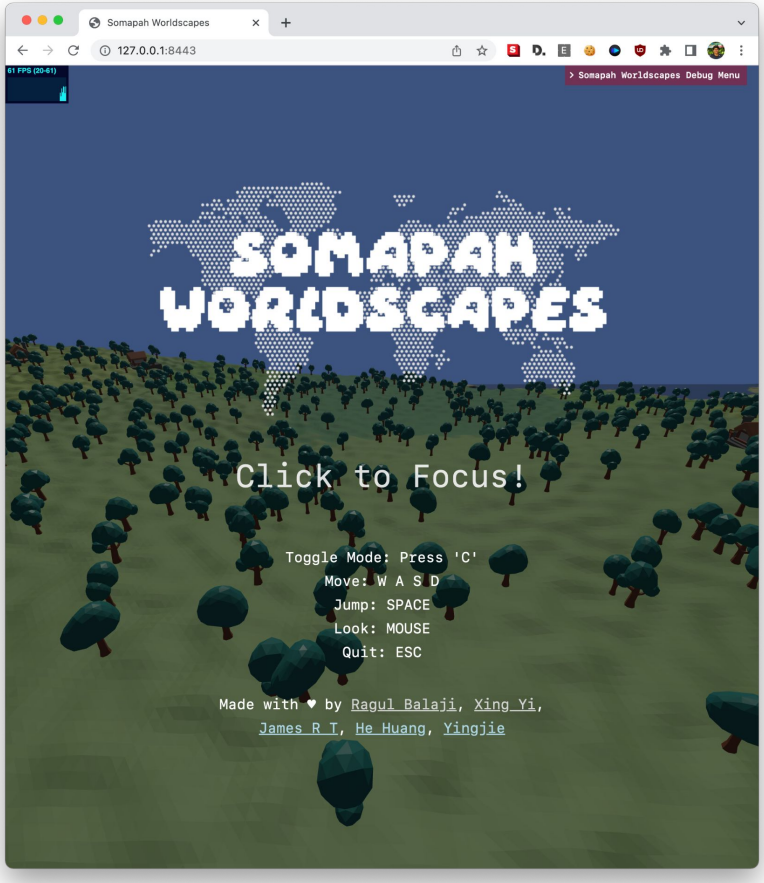
Previously…





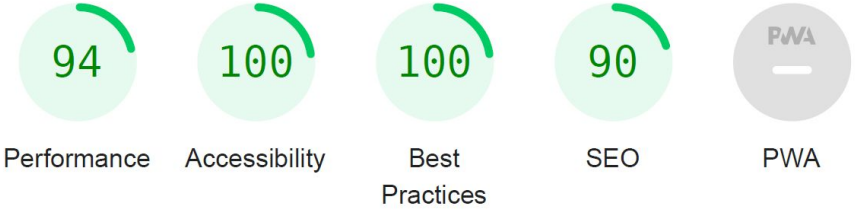
04 Results

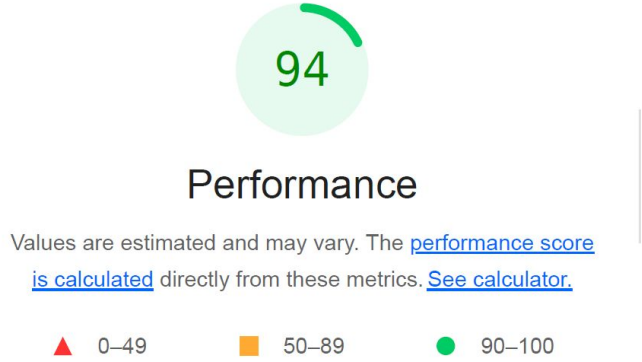
Now!



04 Results

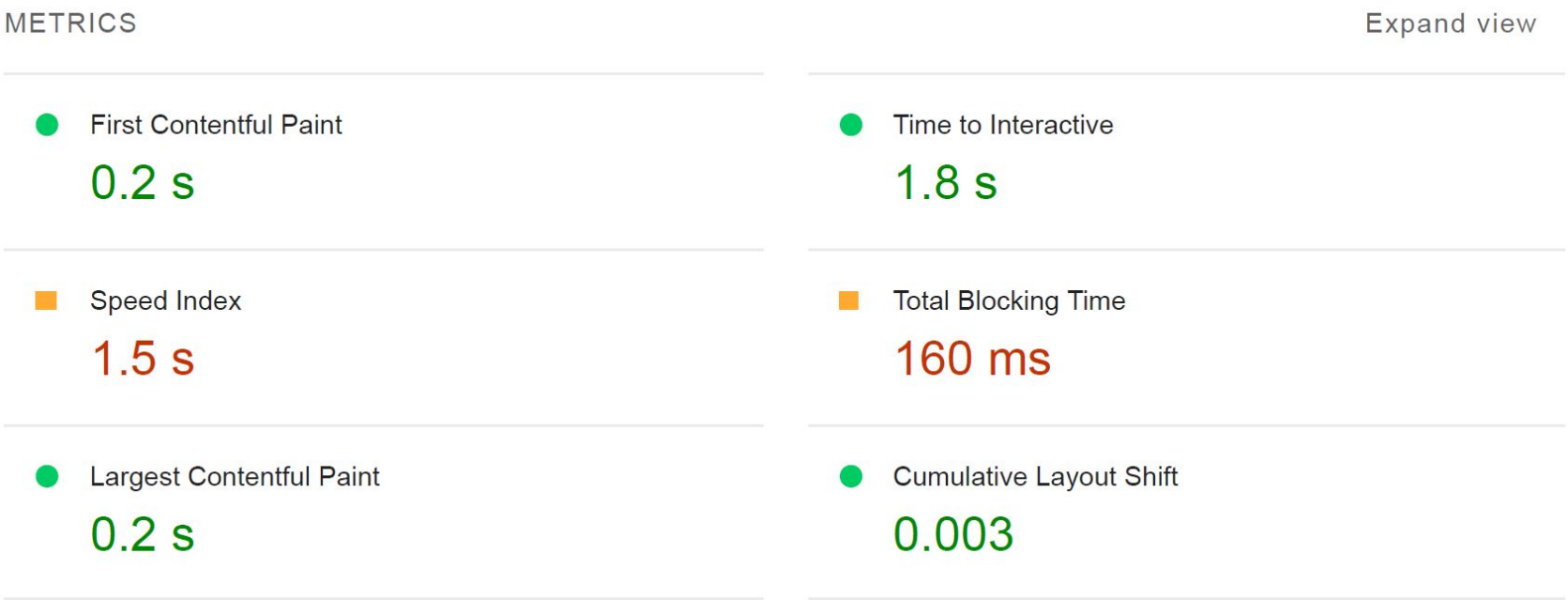
Performance (Google Lighthouse)





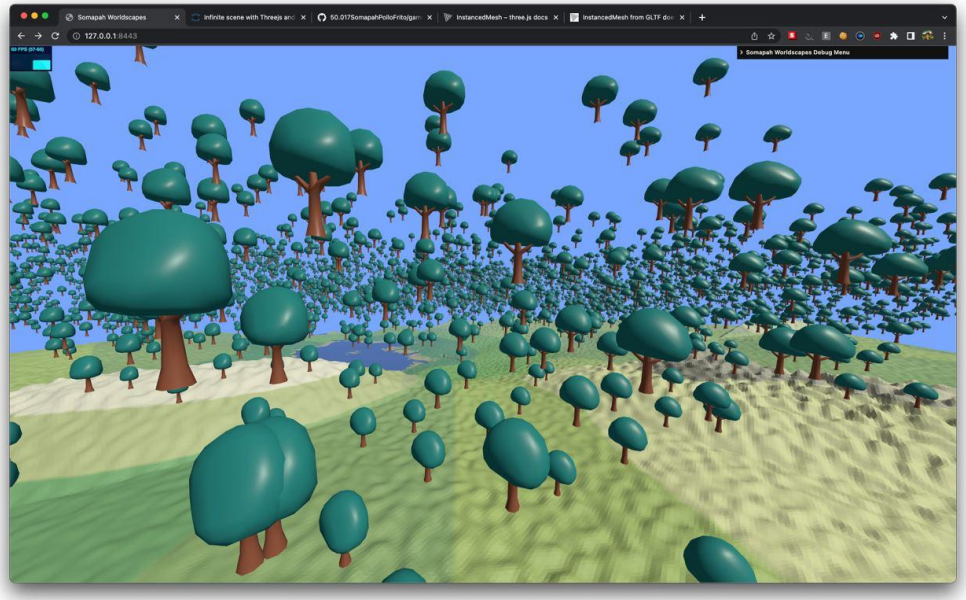
04 Results

Performance (Google Lighthouse)



04 Results

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



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


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