FINAL PROJECT - PROCEDURAL GRAPHICS
INTERESTING 3D LEVEL GENERATOR

Updated Design Doc:

Introduction

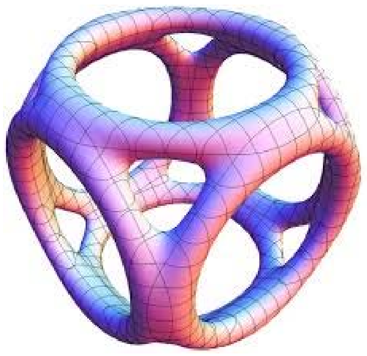
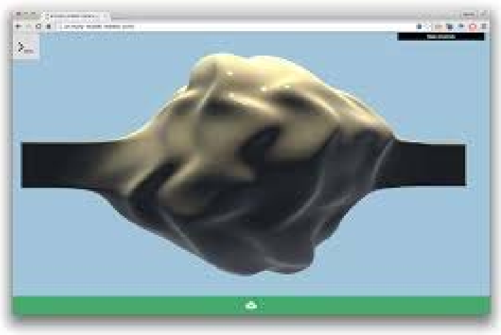
I've been playing games since I was a little kid. Creating interesting levels for games is just a natural direction for my interests to grow into.

Goal

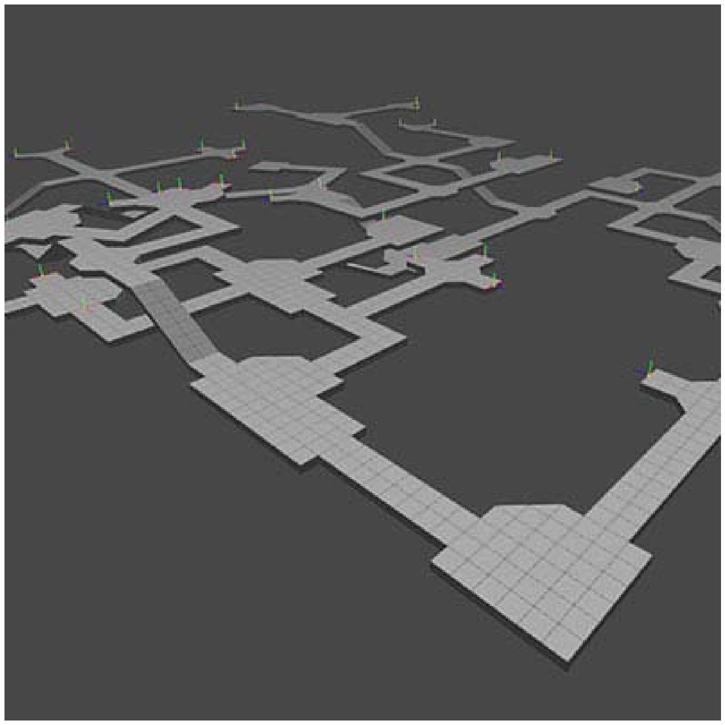
Create a procedurally generated game level, picture the 2D level maps of diablo but in 3D. Then use voxelized implicit surfaces to actually make the generated levels look more interesting (uneven floors, decorative shapes adorning the rooms, etc). That is the minimum shippable product. If I have more time I will then deal with collision handling for the player (FPS camera).

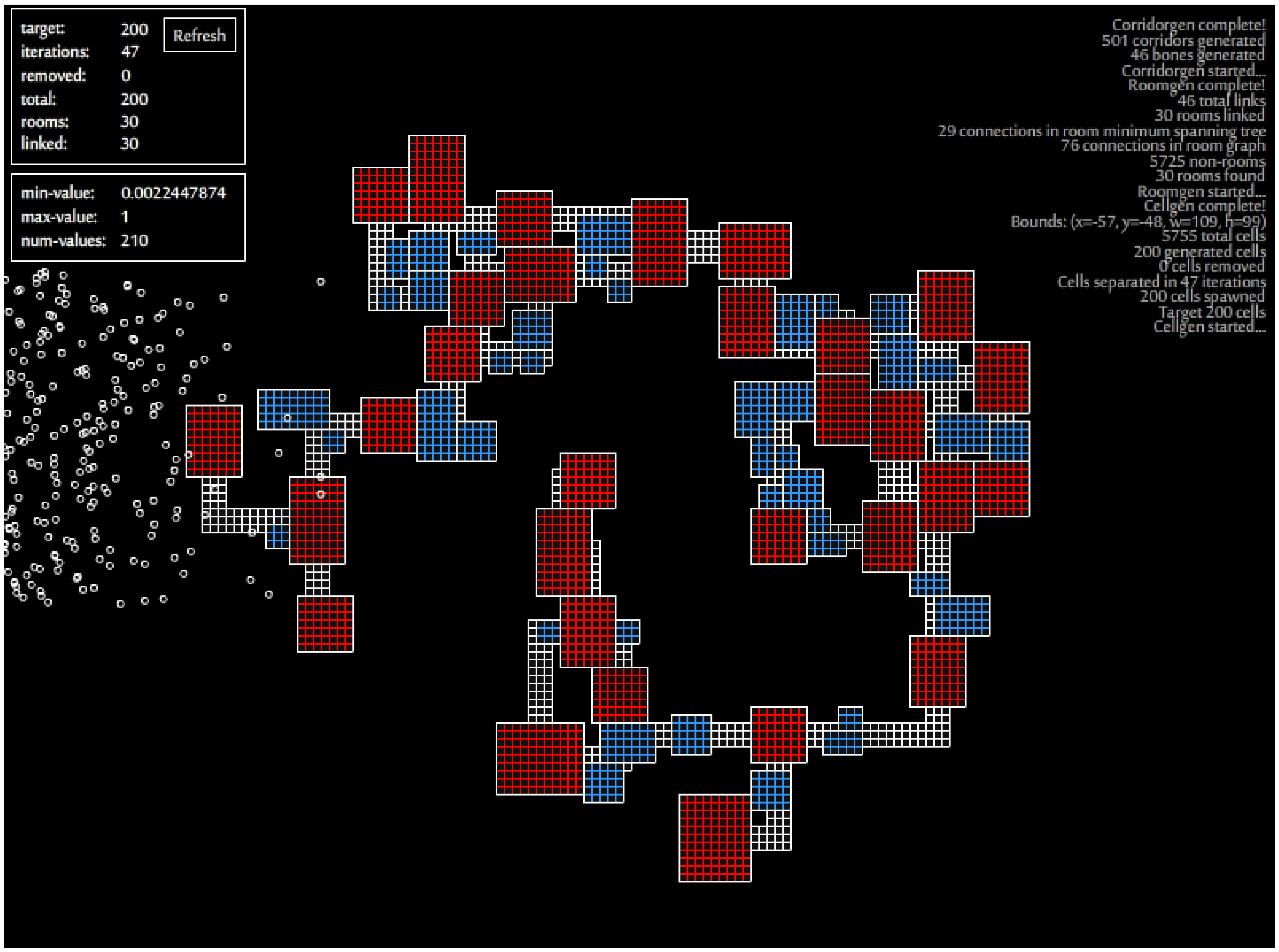
Inspiration:

Implicit surfaces

Level generation





```
target:       200
iterations:   47
removed:      0
total:        200
rooms:        30
linked:       30

min-value:    0.0022447874
max-value:    1
num-values:   210
```

```
Corridorgen complete!
501 corridors generated
46 bones generated
Corridorgen started...
Roomgen complete!
46 total links
30 rooms linked
29 connections in room minimum spanning tree
76 connections in room graph
5725 non-rooms
30 rooms found
Roomgen started...
Cellgen complete!
Bounds: (x=-57, y=-48, w=109, h=99)
5755 total cells
200 generated cells
0 cells removed
Cells separated in 47 iterations
200 cells spawned
Target 200 cells
Cellgen started...
```

Specification

-> procedural 3D game level (multiple stacks of 2D maps).
-> implicit surfaces to create terrain on the resulting map floors
-> custom fog shader

The specifications below will be implemented if I have enough time in the final week
-> collision handling so player can't walk through geometry.

Techniques

Level Generation:

-> 2D maps:

- Using sampling and rejection to layout the slabs in a given space.
- Then use a fake voronoi generation technique to create a graph. The fake voronoi technique consists of first starting with 3 connected nodes, and then for every new node you want to add to the graph you find the 2 closest neighbours from the existing graph and form edges between them.
- We can improve the above technique a bit more by sorting the positions of the slabs along one axis. This makes the connections look more like a voronoi pattern.
- This graph is highly connected so we randomly remove connections.
- The graph can end up with intersecting edges in a few edge cases, so we carry out 2D line intersection tests and remove any intersections if they exist.

-> Walkways between slabs:

- Because of the graph we created above we have edge and node data, in other words we know which points to connect. So between any 2 points we could lay out planes to connect them, but this is boring.
- Instead we can use instancing to place many many tiny cubes along the line segment and then randomly remove cubes to make it look like a crumbling walkway.
- We also need to give the walkway some width so we take the cross product of the direction of the line segment and the y axis (up) to get a horizontal ortho-normal direction for width. Add instanced cubes not just on the line segment but for a certain width for each line segment.

-> 3D Level:

- We can create multiple 2D maps at different heights.

-> Interlevel Connecting Paths:

- This is a similar problem to the one we solved in "Walkway between slabs" section; But now it's in 3D.
- For every layer we pick a random node/slab as the starting point of our path between layers.
- For the end point of that line segment we search through the nodes in the layer above for rooms that are beyond a certain distance from the randomly picked starting node (so paths don't go straight up and there is more complexity in connections), and form a list of these "toNodes".
- Pick a random "toNode", and using the random starting node we have a 3D line segment.
- Create a similar instancing cubes setup for these paths as we did with the walkways.
- Remove random lines, and also carry out 3D intersection tests and remove any intersecting paths, if they exist.

-> Path shifting for Paths:

- To make the paths connecting walkways seem more organic and prevent janky looking paths that start at the center of each cell and end at the center of the other one (this is a problem as a player can never go to a higher layer, they will be stuck underneath the “toNode”).
- We need to shift paths to the edges of the cells they are connecting. Simply offset by the width and length in the correct direction.
- To add organic paths we should shift by both the width and length.

-> Fog:

- Created in the shader, with global control of density, color, and a on/off switch.
- A good approximation of fog fall-off is: $e^{-(fogDensity^2\ x\ distance^2)}$
- Fog also appears to have different densities at the periphery of our vision, so we need to account for rimColor.
- Resource: http://in2gpu.com/2014/07/22/create-fog-shader/

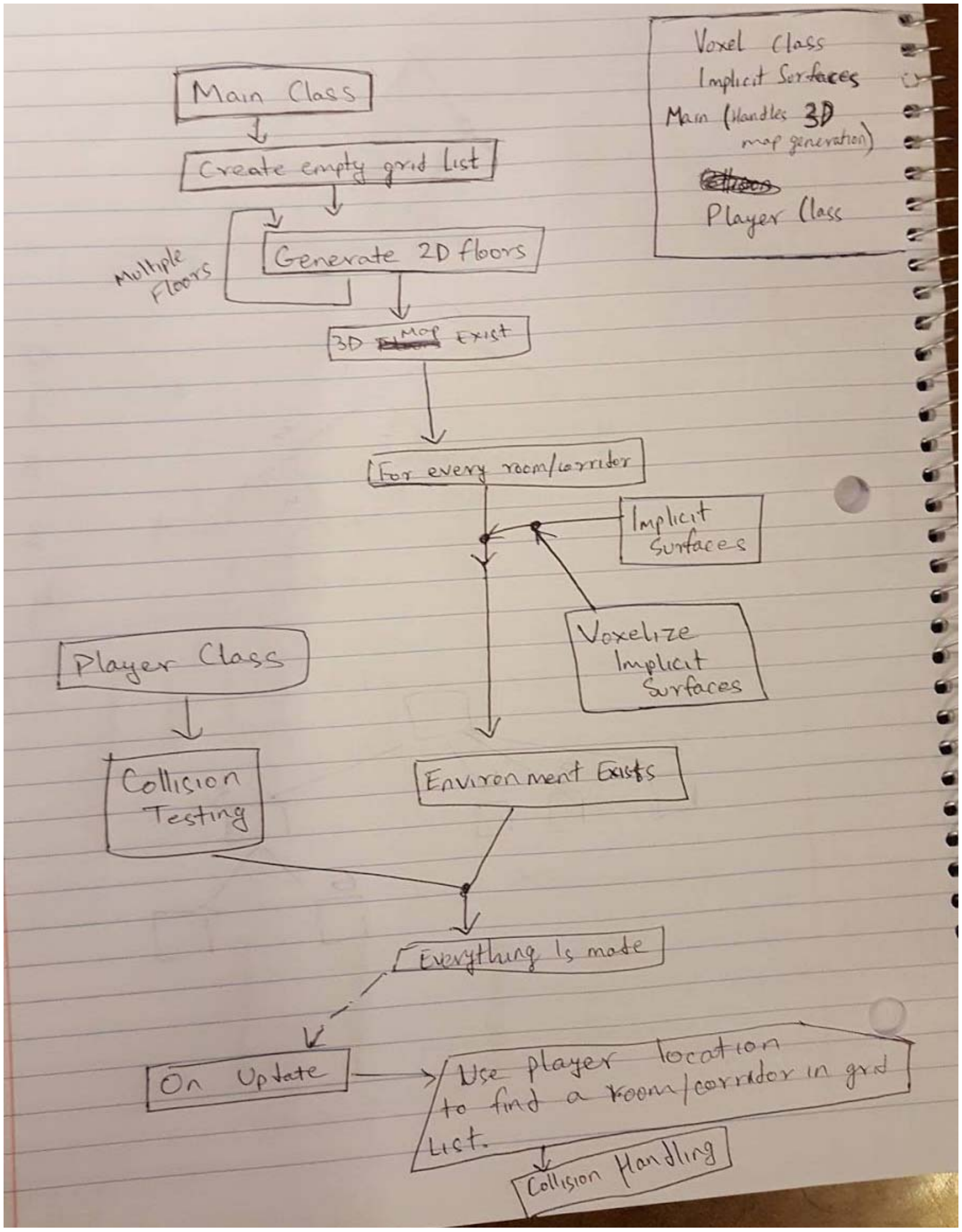
-> Terrain:

- Terrain was created in the shader.
- Create an elevation map and a moisture map using smoothed 2D noise with different seed values (or different noise functions).
- Use the elevation map to deform vertices in the shader.
- Create a moisture map (similar to the elevation map).
- Use the float values from the elevation and moisture as uv values to determine colors from gradients.

-> Grid based Acceleration:

- Takes too much memory, and so was never used for anything.

Design

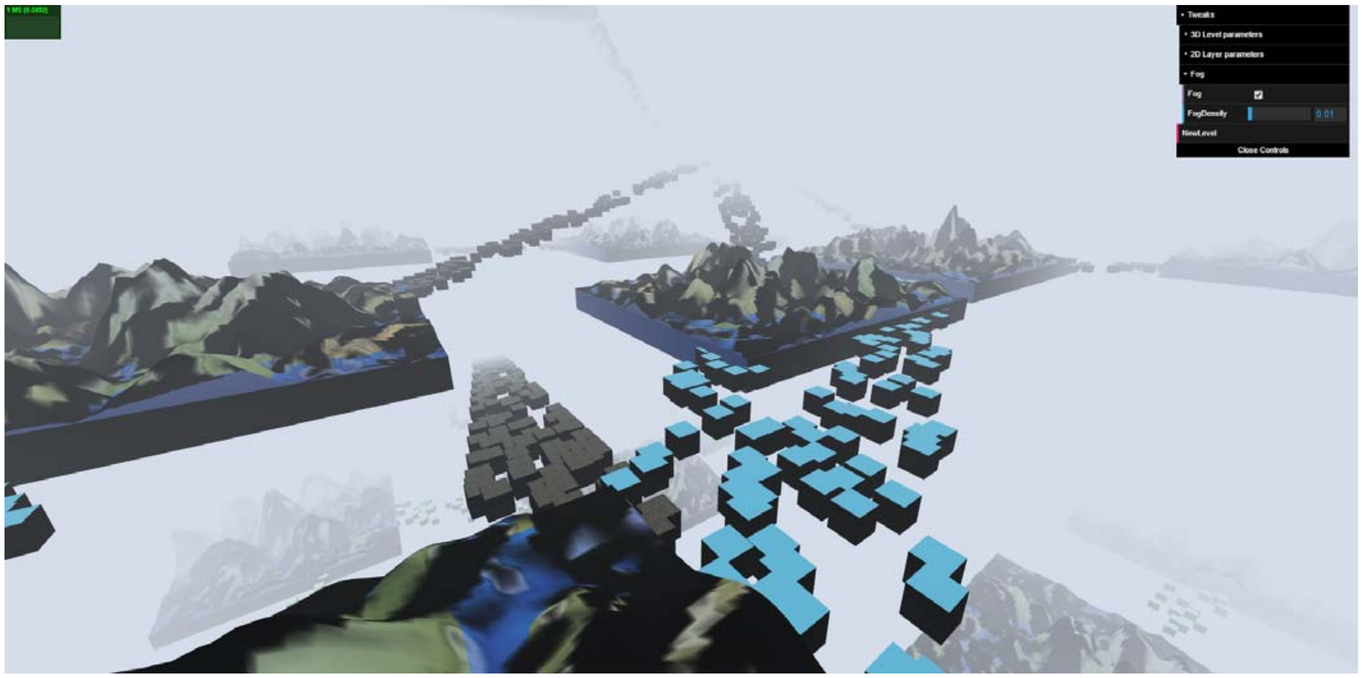


TimeLine

Week 1: 2D procedural level generation
Week 1.5: 3D level (stacks of connected 2D levels)
Week 2.5: Implicit surfaces (Terrain) to make the rooms prettier
Remaining Time: Debugging collision handling

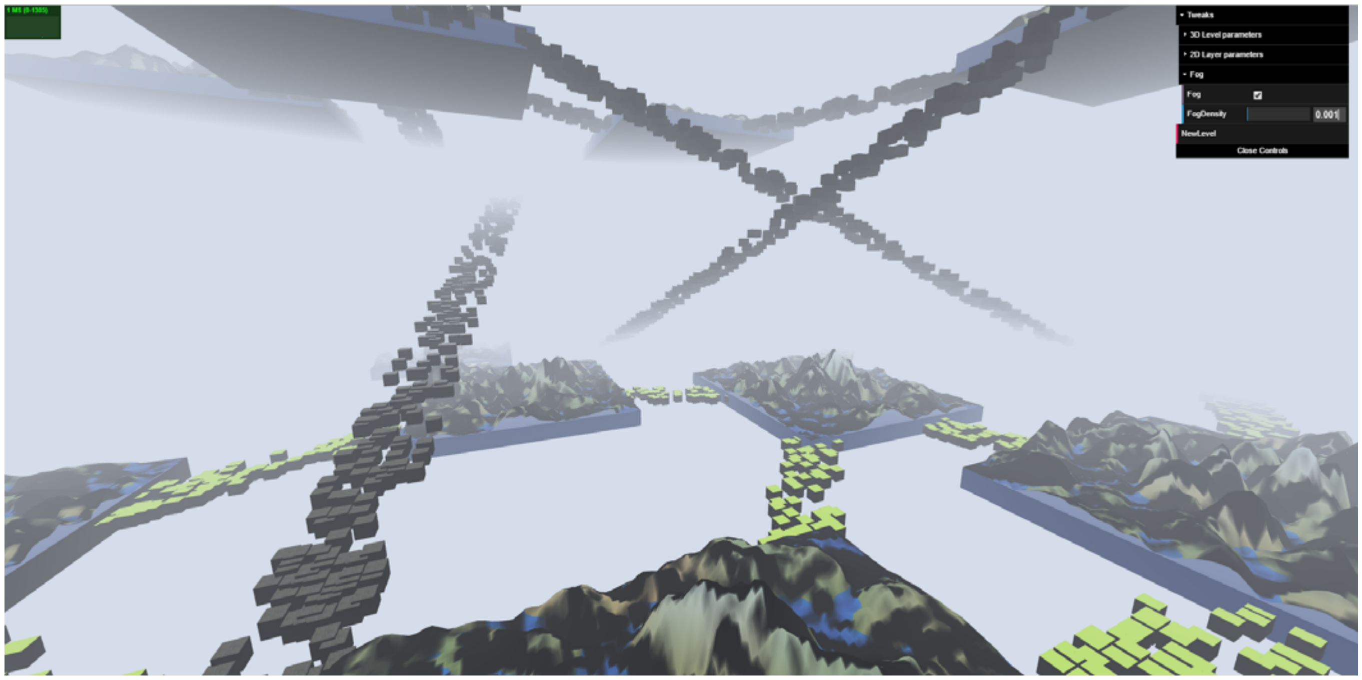
Results:

Fog:



Fog ✓

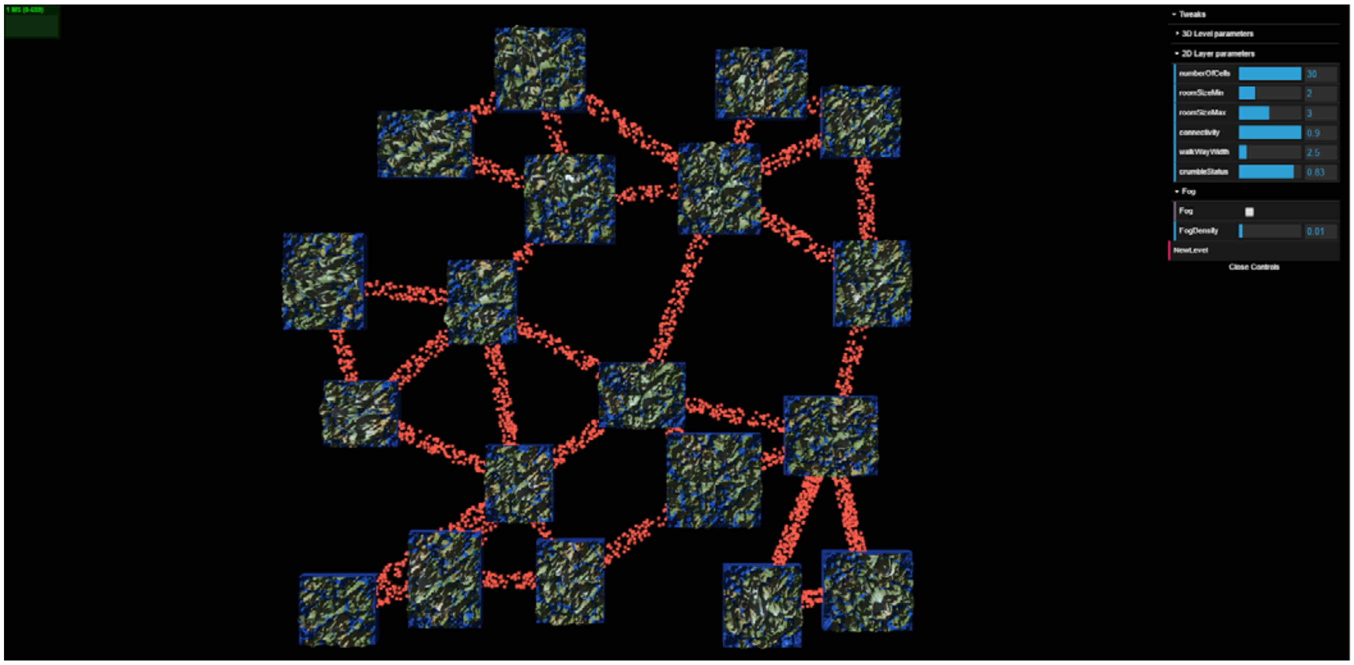
FogDensity 0.01

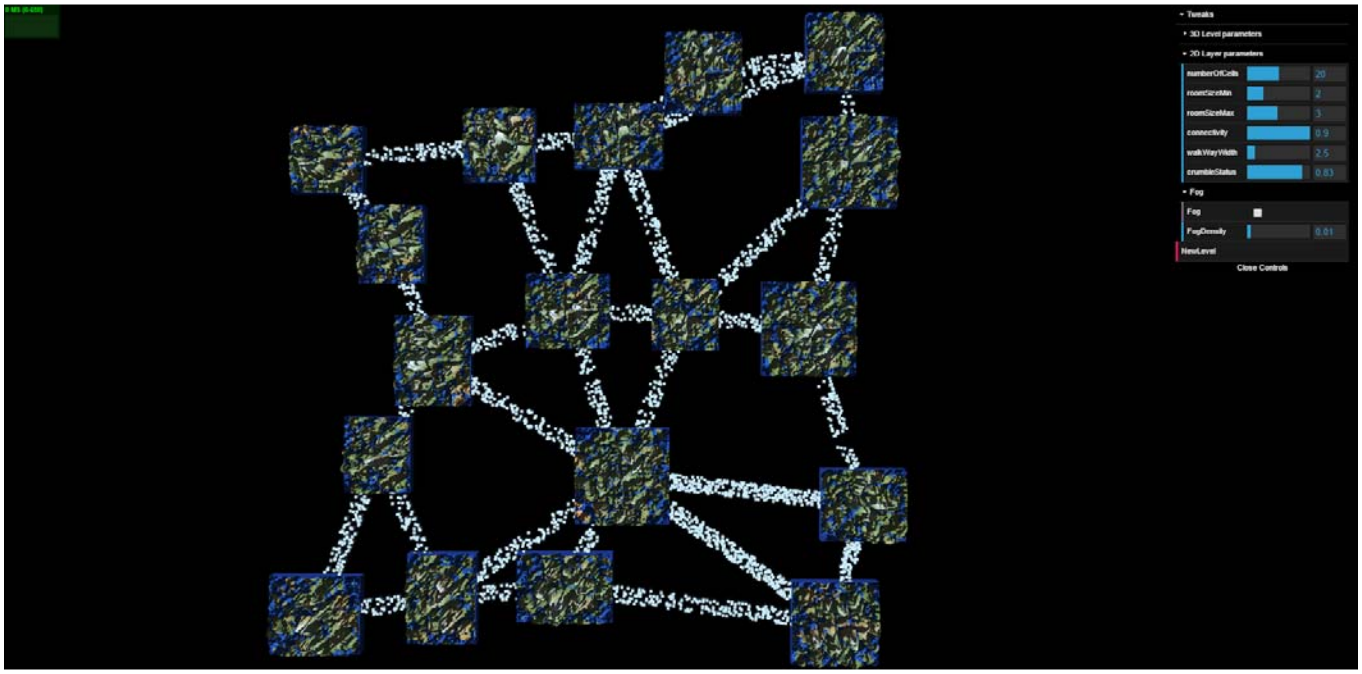


Fog ✓

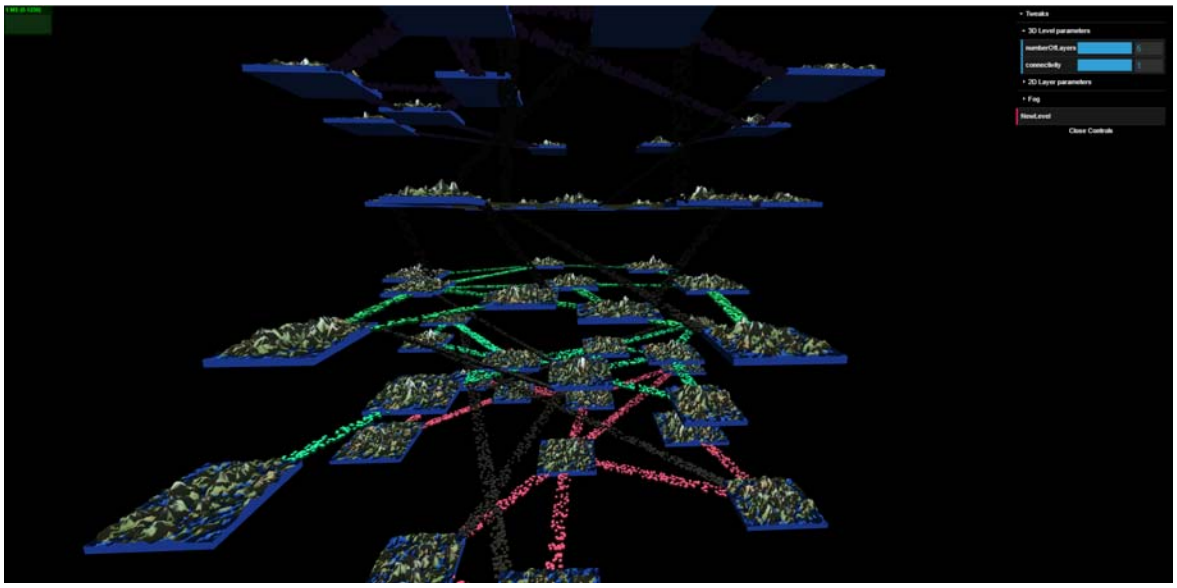
FogDensity 0.001

Voronoi 2D Map:


Tweaks
3D Level parameters
2D Layer parameters
numberOfCells 30
roomSizeMin 2
roomSizeMax 3
connectivity 0.9
walkWayWidth 2.5
crumbleStatus 0.83
Fog
Fog
FogDensity 0.01
NewLevel
Close Controls

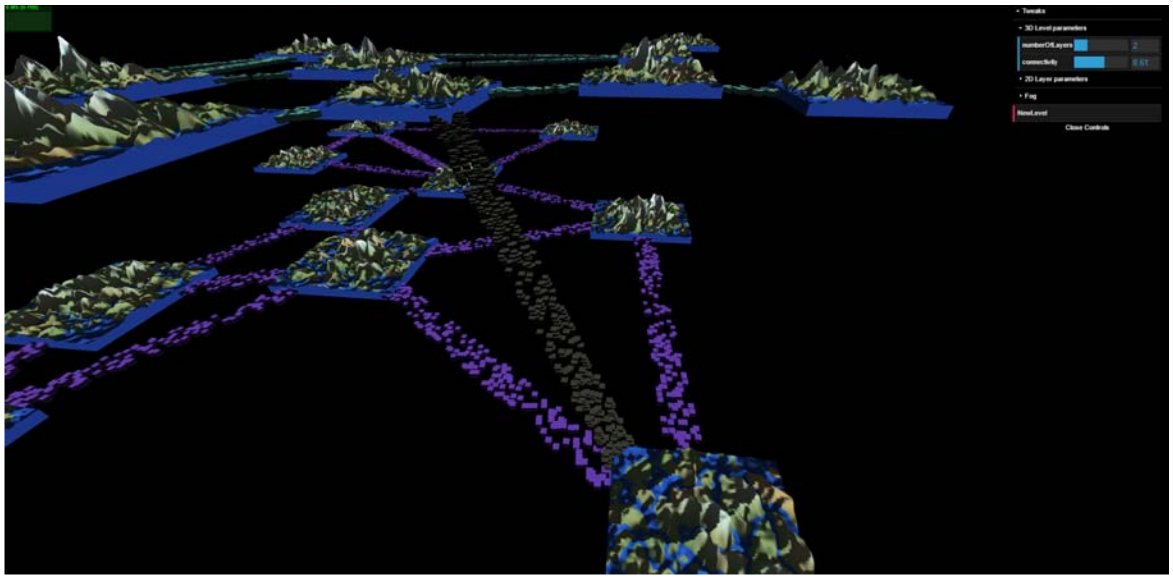

Tweaks
3D Level parameters
2D Layer parameters
numberOfCells 20
roomSizeMin 2
roomSizeMax 3
connectivity 0.9
walkWayWidth 2.5
crumbleStatus 0.83
Fog
Fog
FogDensity 0.01
NewLevel
Close Controls

Multiple Layers:

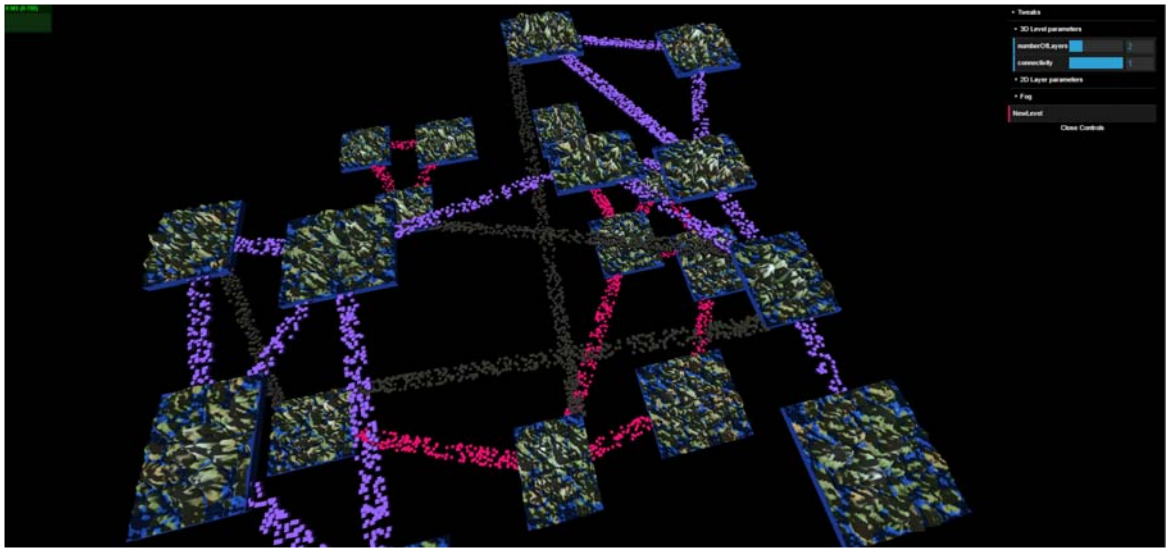


numberOfLayers 5
connectivity 1

Layer Connectivity:

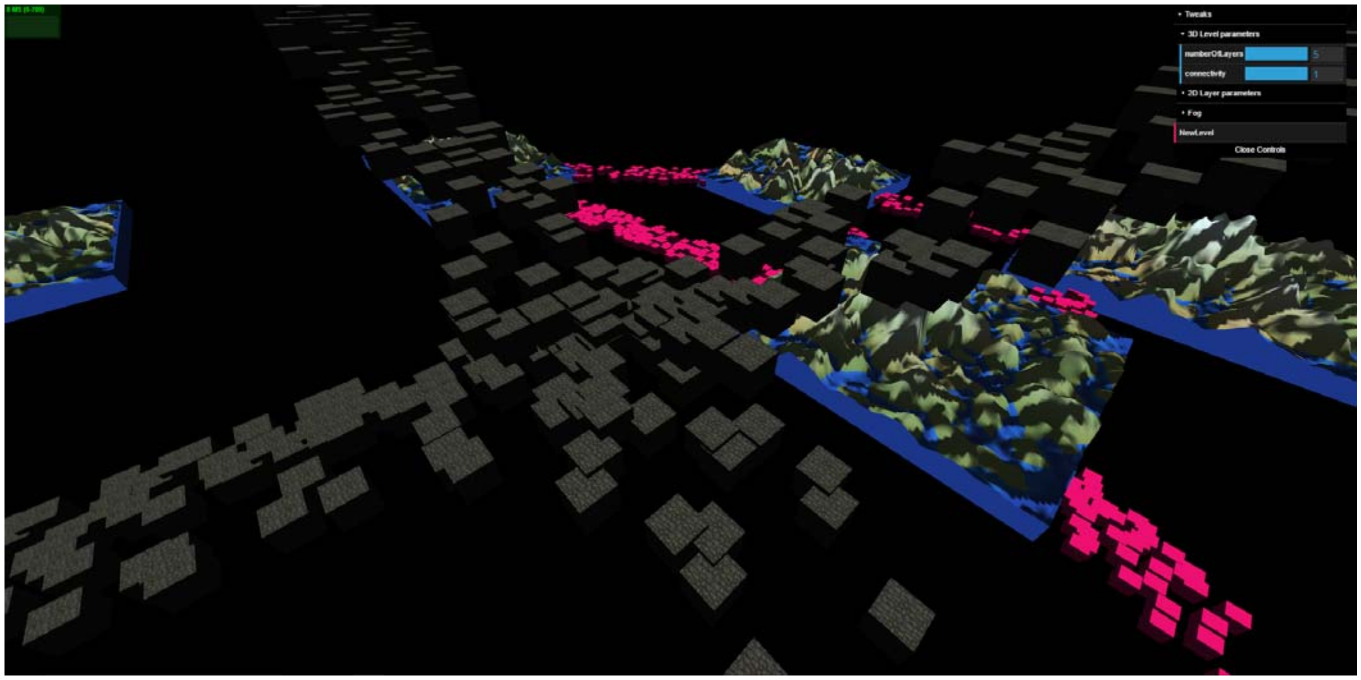


numberOfLayers 2
connectivity 0.61



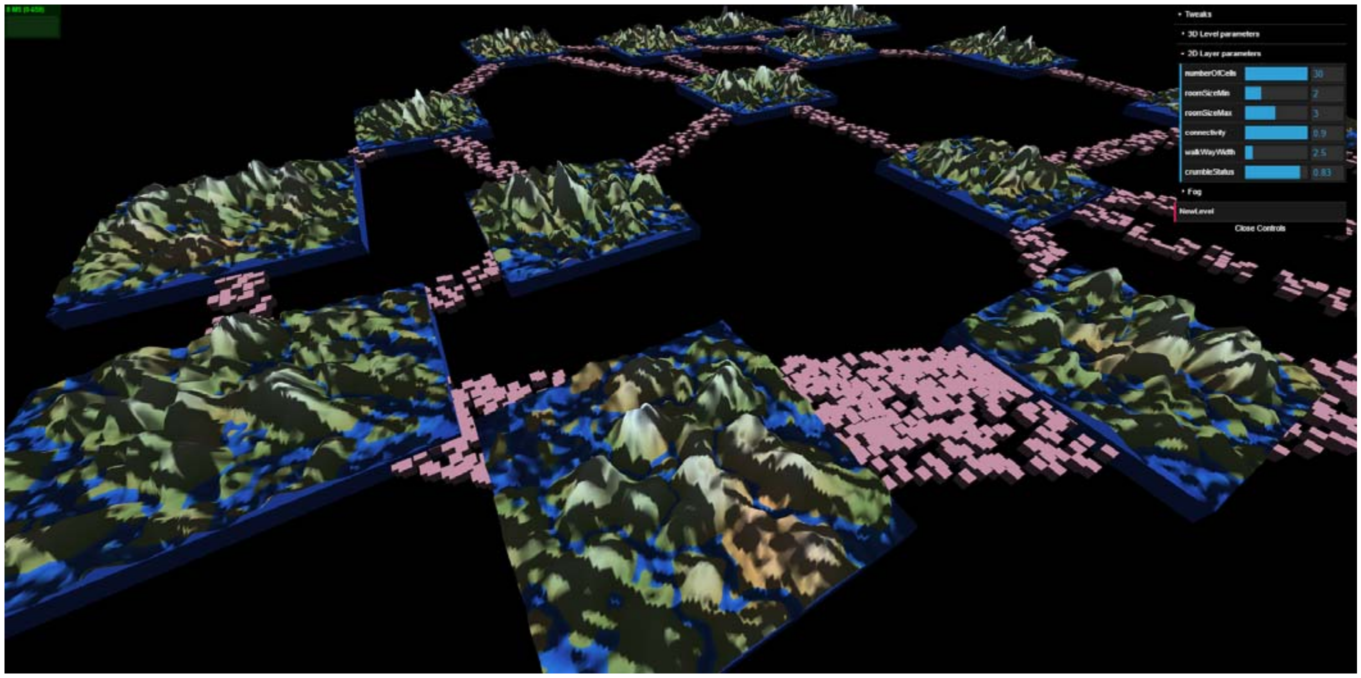
numberOfLayers 2
connectivity 1

Crossroads Emergent Behaviour:



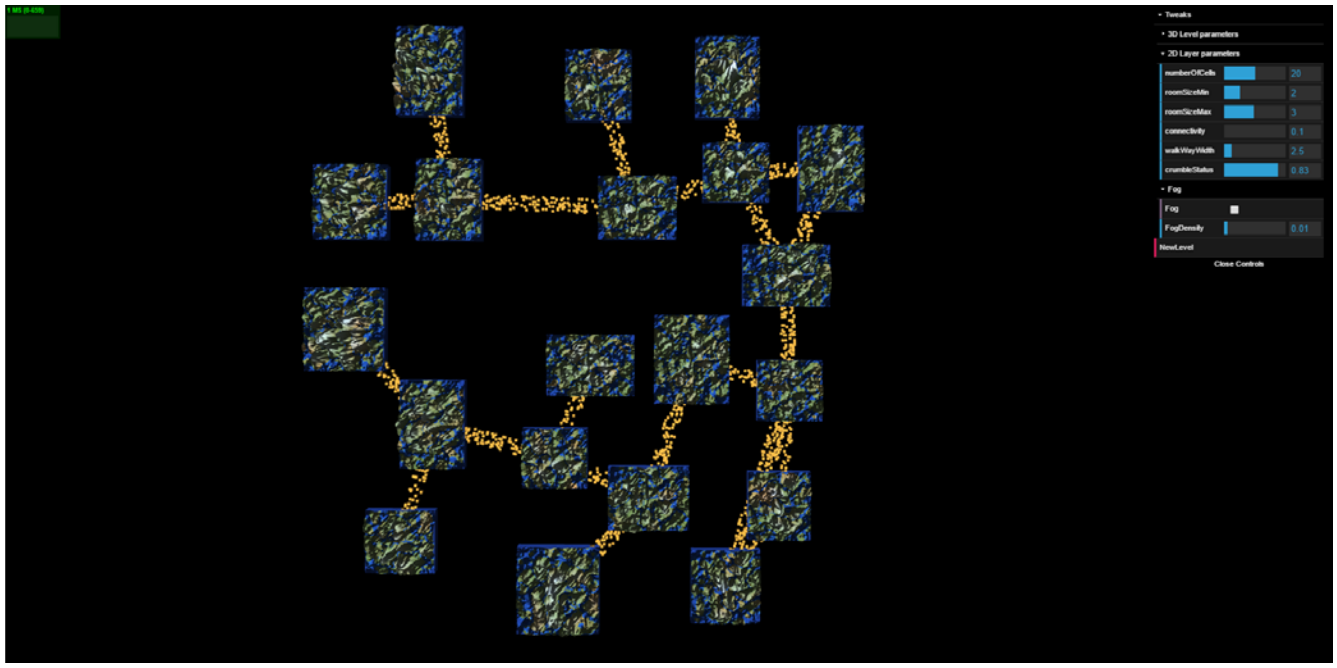
numberOfLayers 5
connectivity 1

Terrain:



numberOfCells 30
roomSizeMin 2
roomSizeMax 3
connectivity 0.9
walkWayWidth 2.5
crumbleStatus 0.83

2D Connectivity:

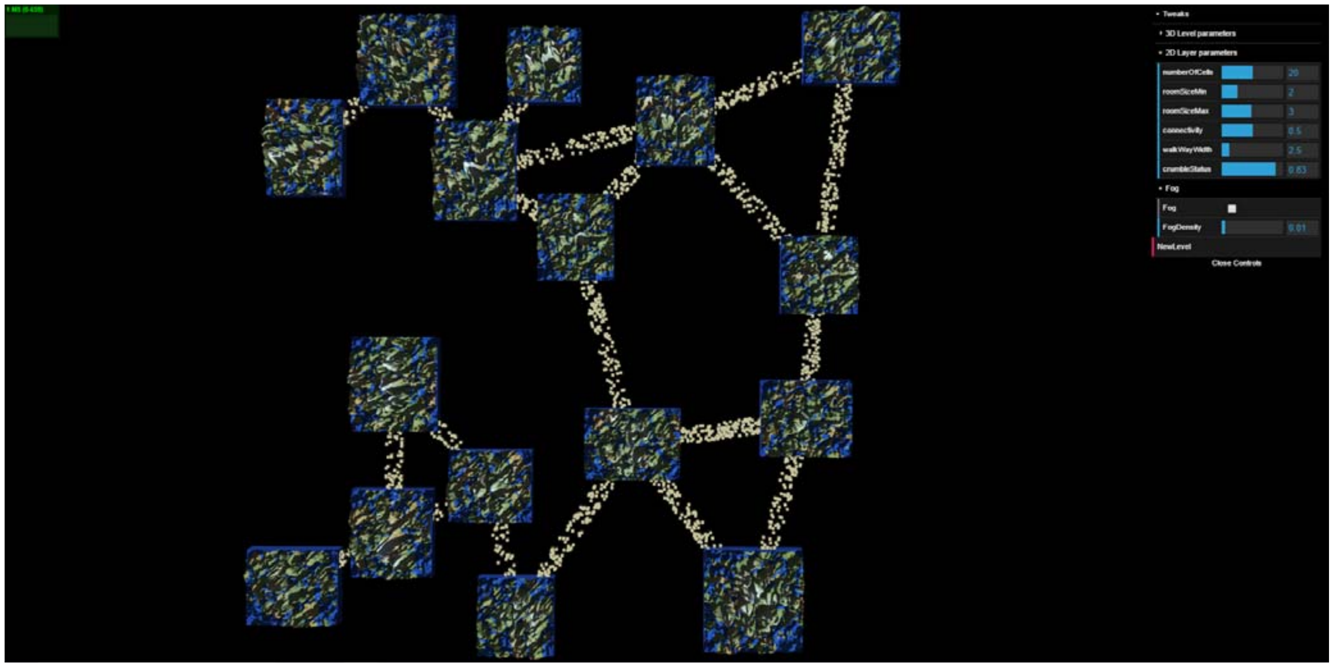


Tweaks
3D Level parameters
2D Layer parameters

numberOfCells	20
roomSizeMin	2
roomSizeMax	3
connectivity	0.1
walkWayWidth	2.5
crumbleStatus	0.83

Fog

Fog	☐
FogDensity	0.01



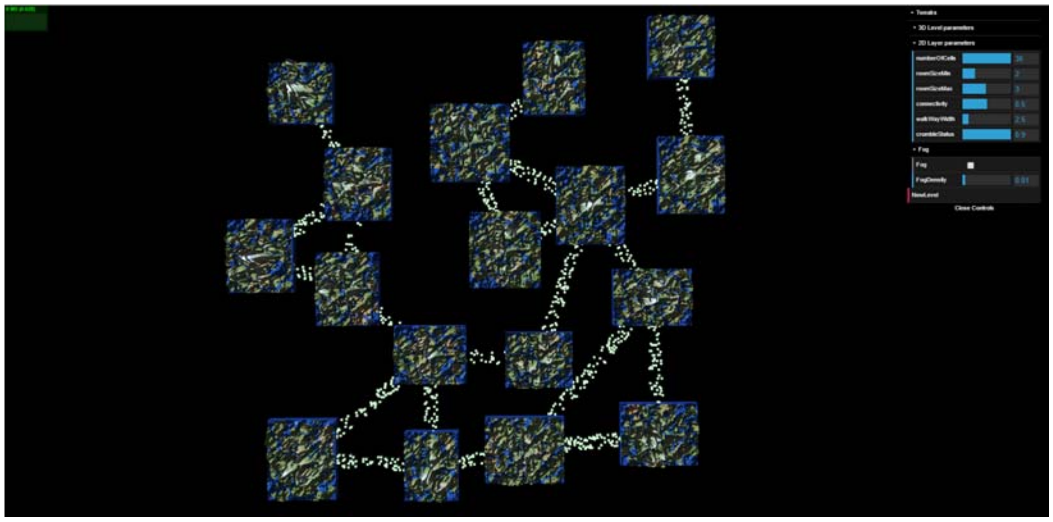
Tweaks
3D Level parameters
2D Layer parameters

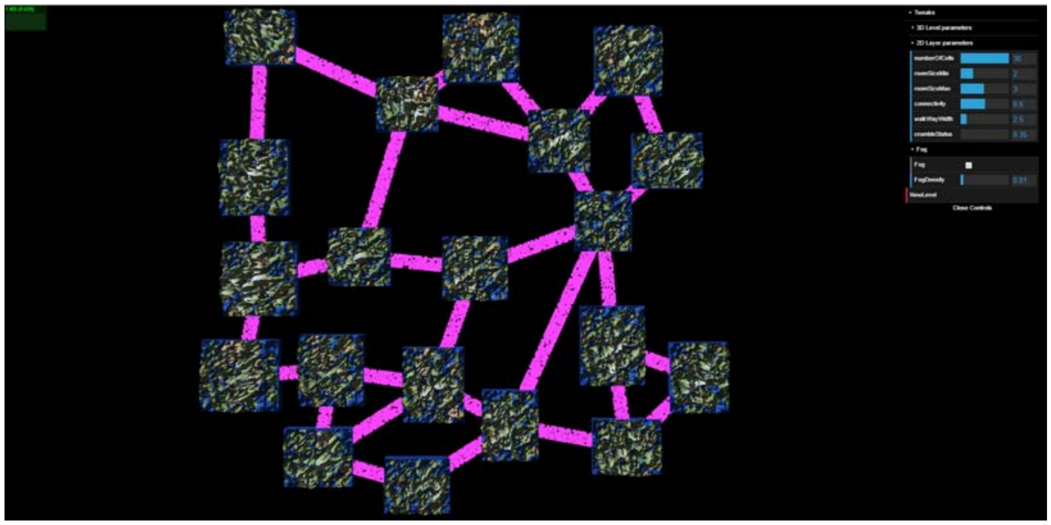
numberOfCells	20
roomSizeMin	2
roomSizeMax	3
connectivity	0.5
walkWayWidth	2.5
crumbleStatus	0.83

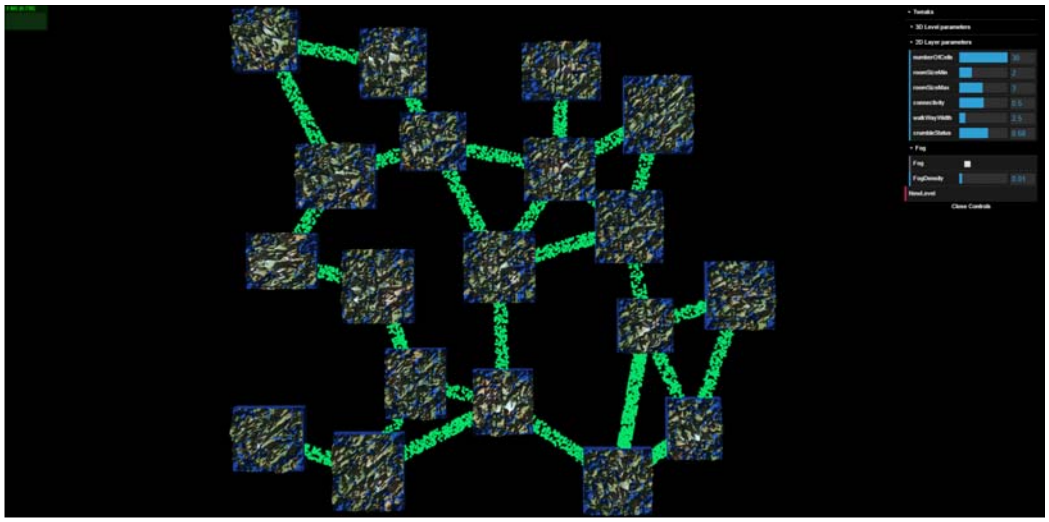
Fog

Fog	☐
FogDensity	0.01

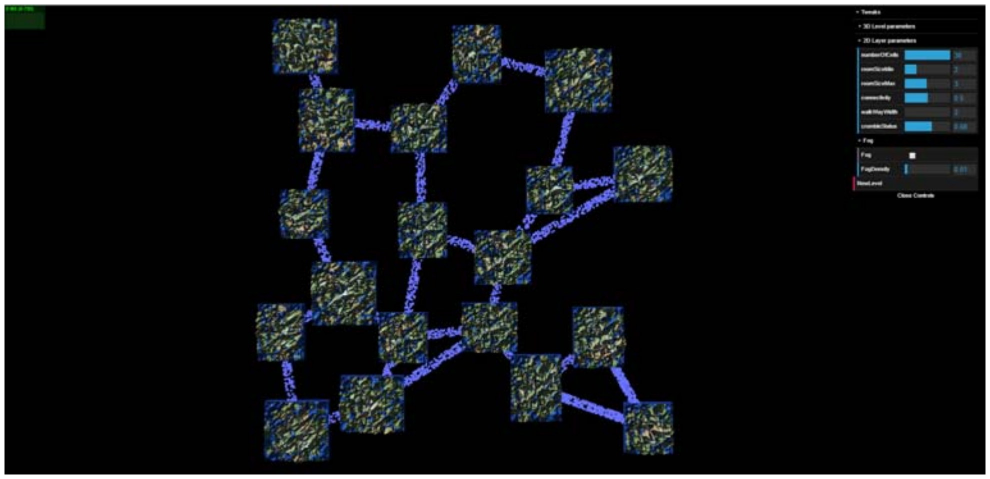
Crumble Status:

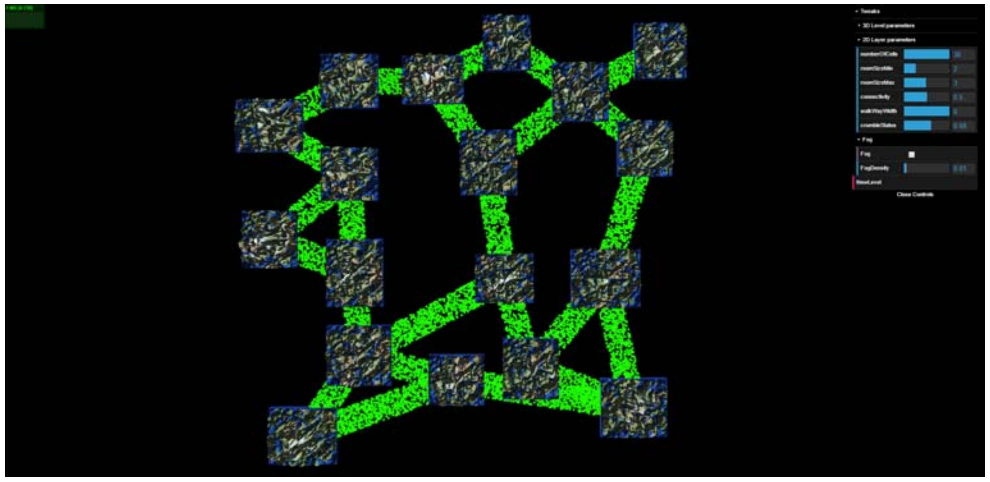

Tweaks
3D Level parameters
2D Layer parameters
numberOfCells 30
roomSizeMin 2
roomSizeMax 3
connectivity 0.5
walkWayWidth 2.5
crumbleStatus 0.9
Fog
Fog
FogDensity 0.01
NewLevel
Close Controls


Tweaks
3D Level parameters
2D Layer parameters
numberOfCells 30
roomSizeMin 2
roomSizeMax 3
connectivity 0.5
walkWayWidth 2.5
crumbleStatus 0.35
Fog
Fog
FogDensity 0.01
NewLevel
Close Controls

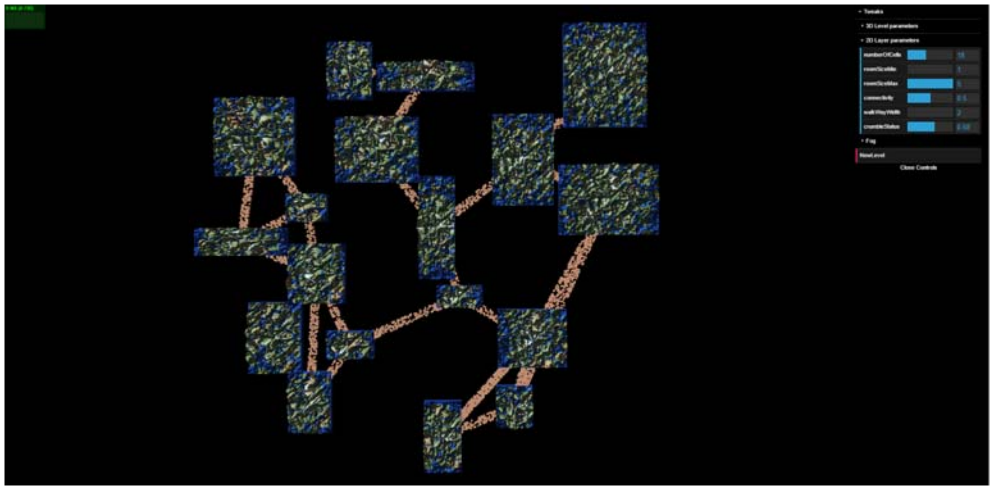

Tweaks
3D Level parameters
2D Layer parameters
numberOfCells 30
roomSizeMin 2
roomSizeMax 3
connectivity 0.5
walkWayWidth 2.5
crumbleStatus 0.58
Fog
Fog
FogDensity 0.01
NewLevel
Close Controls

Walkway Control:





Variable Room Size:



Evaluation:

- How Well did I do? Fair (maybe even pretty good :D :P). Changed a lot of stuff from the original design doc but made it work and look relatively impressive.
- (I didn't take screenshots, so no Work in Progress comparisons)
- I changed the techniques I wanted to use for the creating the 2D maps to something that was less computationally expensive and gave more interesting voronoi-ish maps; however I lost the complexity of non-straight paths (the original algorithm merged tiny rooms into the walkways/corridors).
- I wanted to use iq's volcanic terrain shader: https://www.shadertoy.com/view/XsX3RB ,but it was too complicated to decode. I ended up using some other terrain generation technique.
- Introduced a custom fog shader.
- Initially tried voxelizing different noises in the cpu for terrain, but voxelized noise doesn't look good at all.
- Went through many algorithms for map generation and terrain generation before settling on one I liked.

Future Work:

- Make a kD tree acceleration structure for collision handling (grid acceleration takes too much memory)
- Character Controls
- Collision Handling with a character
- Trampolines
- Fill up empty background space with something (skyboxes didn't look nice)

Acknowledgement:

- Fake Voronoi: Discussion with Trung
- Terrain: http://www.redblobgames.com/maps/terrain-from-noise/
- Fog: http://in2gpu.com/2014/07/22/create-fog-shader/