












# Markov chain parameters

## Initialization

```
SetDirectory[NotebookDirectory[]];
Needs["Tilemap`"]
cityFiles = FileNames["*.txt", "../data/processed/elements"];
mapMatrices = ExtractMapMatrix[#] & /@ cityFiles;
tileSet = ExtractTileset[Flatten[mapMatrices]];

charToTileAssociation = <|"T" -> , "g" -> , "H" -> , "p" -> , "D" -> ,
    "e" -> , "W" -> , "l" -> , "M" -> , "r" -> , "R" -> |>;

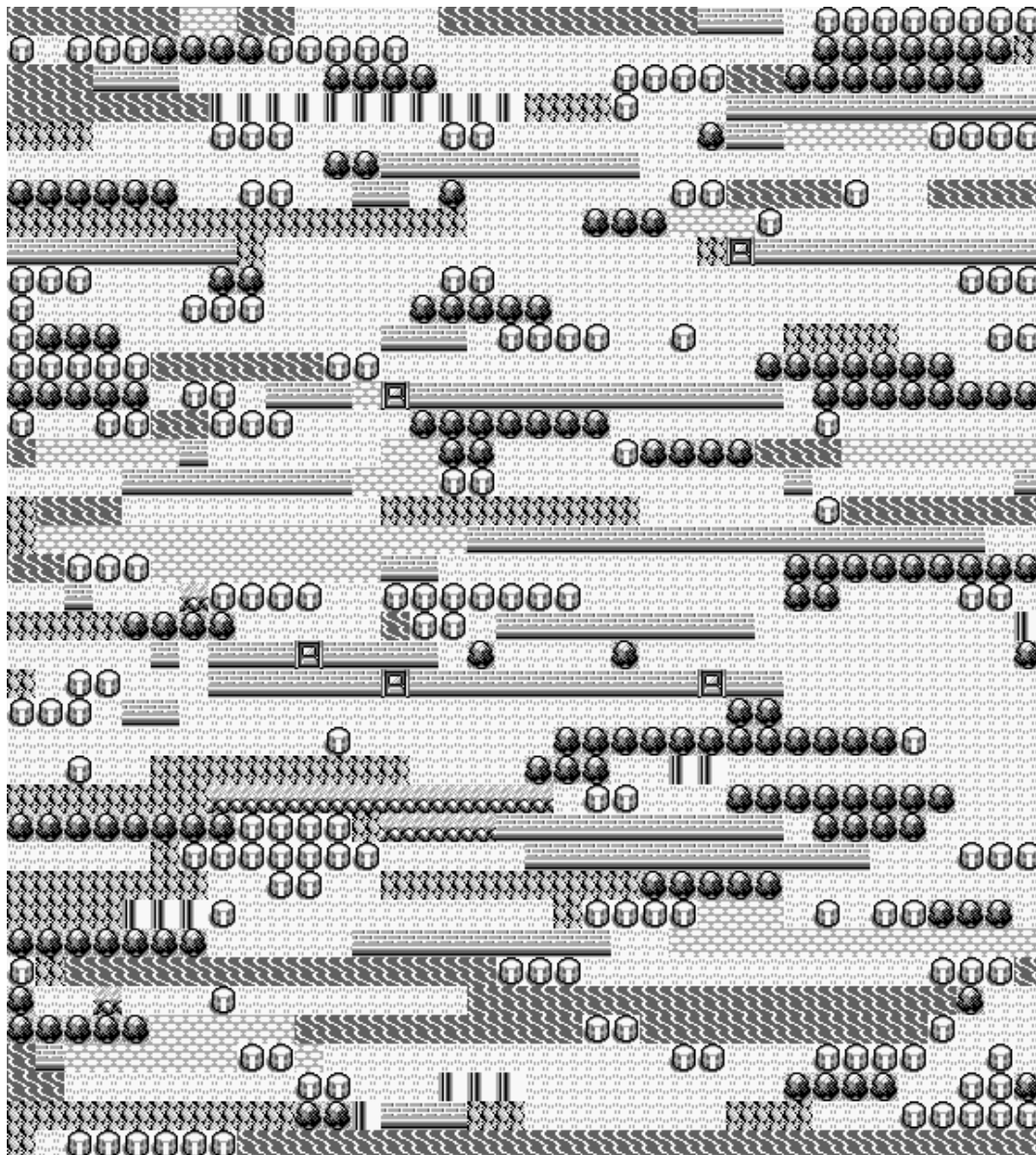
CreateMap[kernel_List] := Module[{
    probabilities = MapDistribution[kernel, #, tileSet] & /@ mapMatrices,
    totalProbabilities,
    map},
    totalProbabilities =
        Total[Keys[#]] -> tileSet & /@ Merge[probabilities, Plus];
    map = GenerateMap[kernel, Array[# &, {40, 36}], totalProbabilities];
    ImageAssemble[map /. charToTileAssociation]
```

# Parameter exploring

## One predecessor kernels

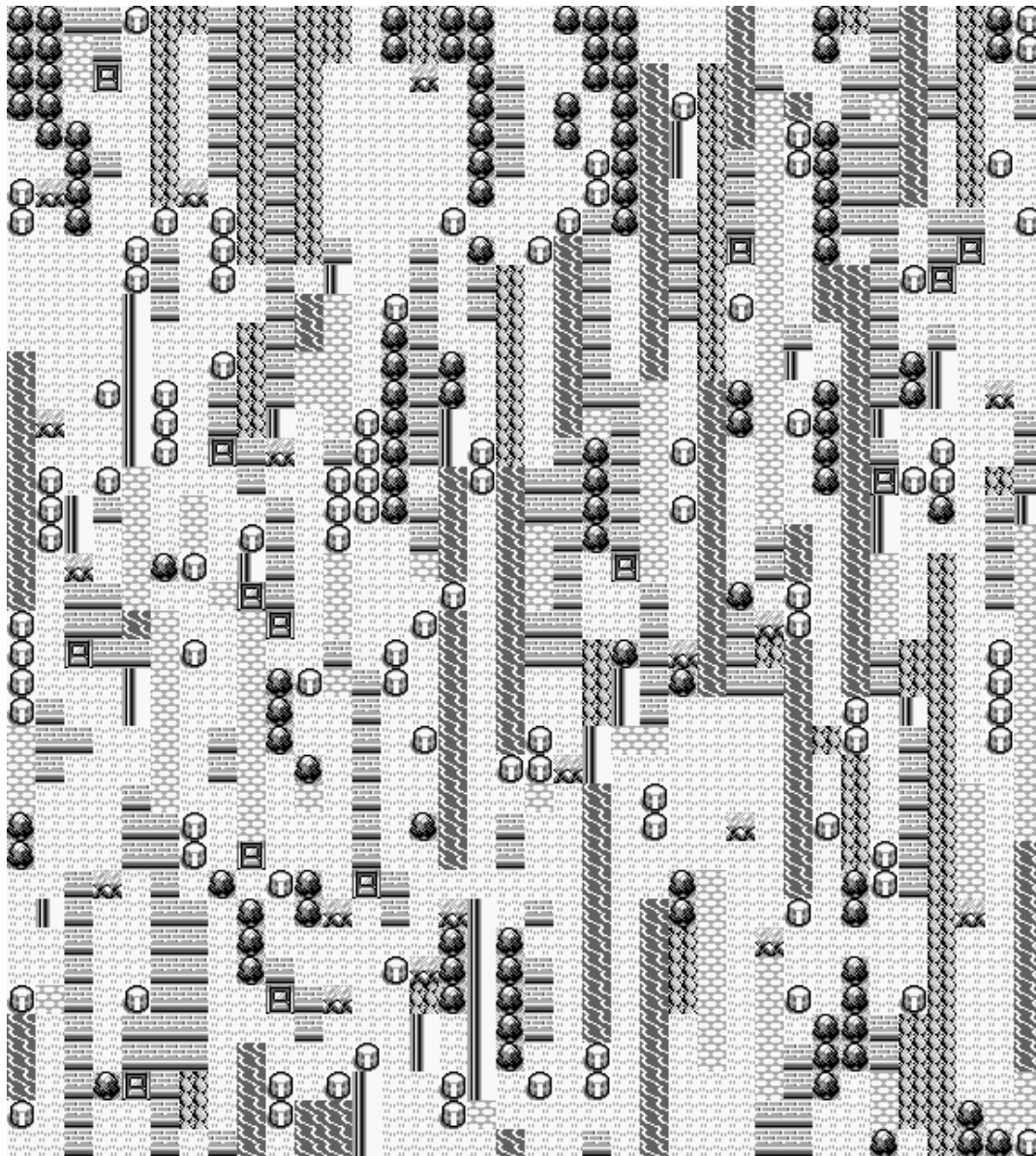
$$\text{kernel} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix};$$

CreateMap[kernel]



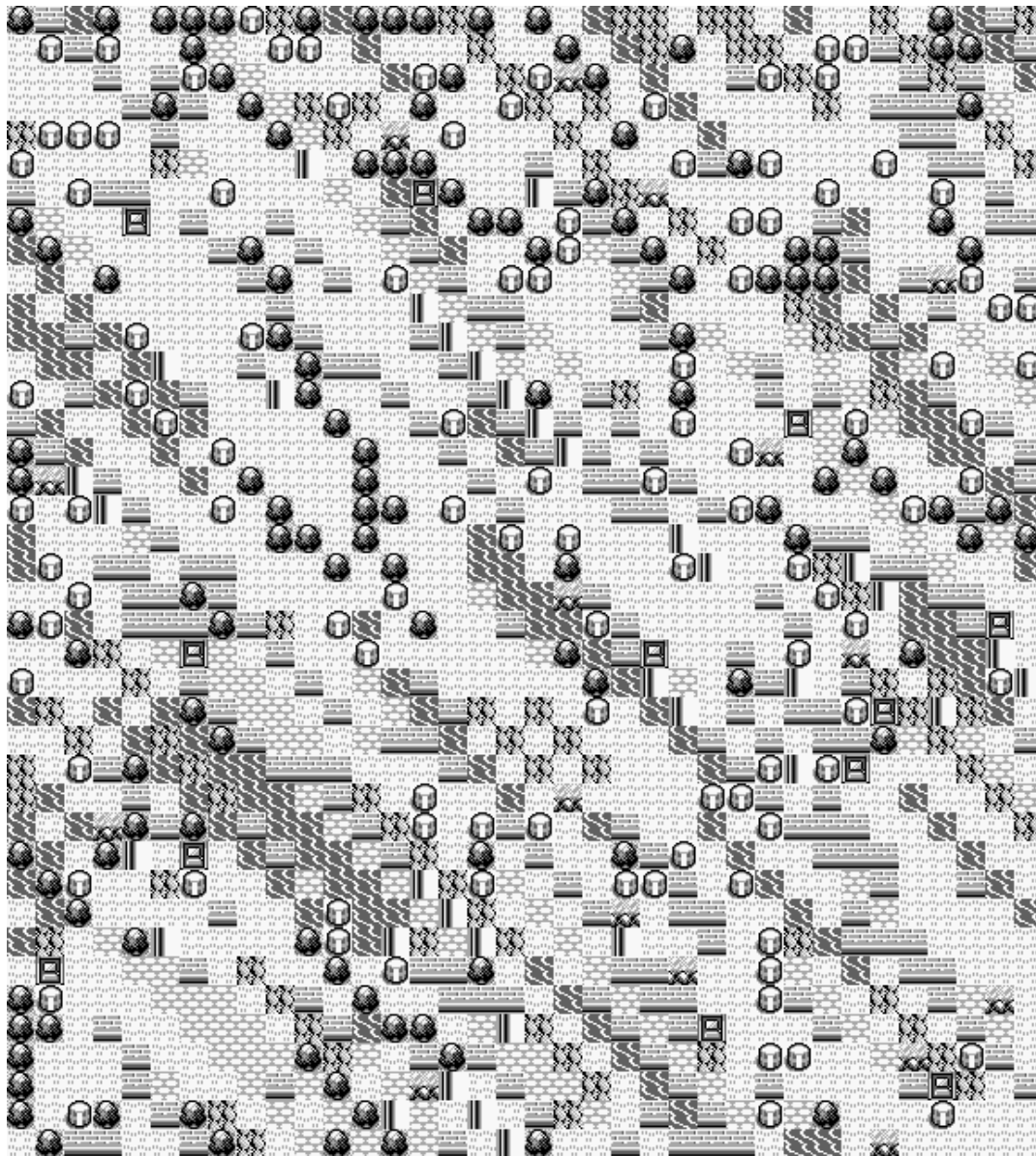
```
kernel =  $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix};$ 
```

```
CreateMap[kernel]
```



```
kernel =  $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix};$ 
```

```
CreateMap[kernel]
```



As we can see with one predecessor the generated maps keep a spatial structure similar to position of the predecessor. Either horizontal, vertical or diagonal.

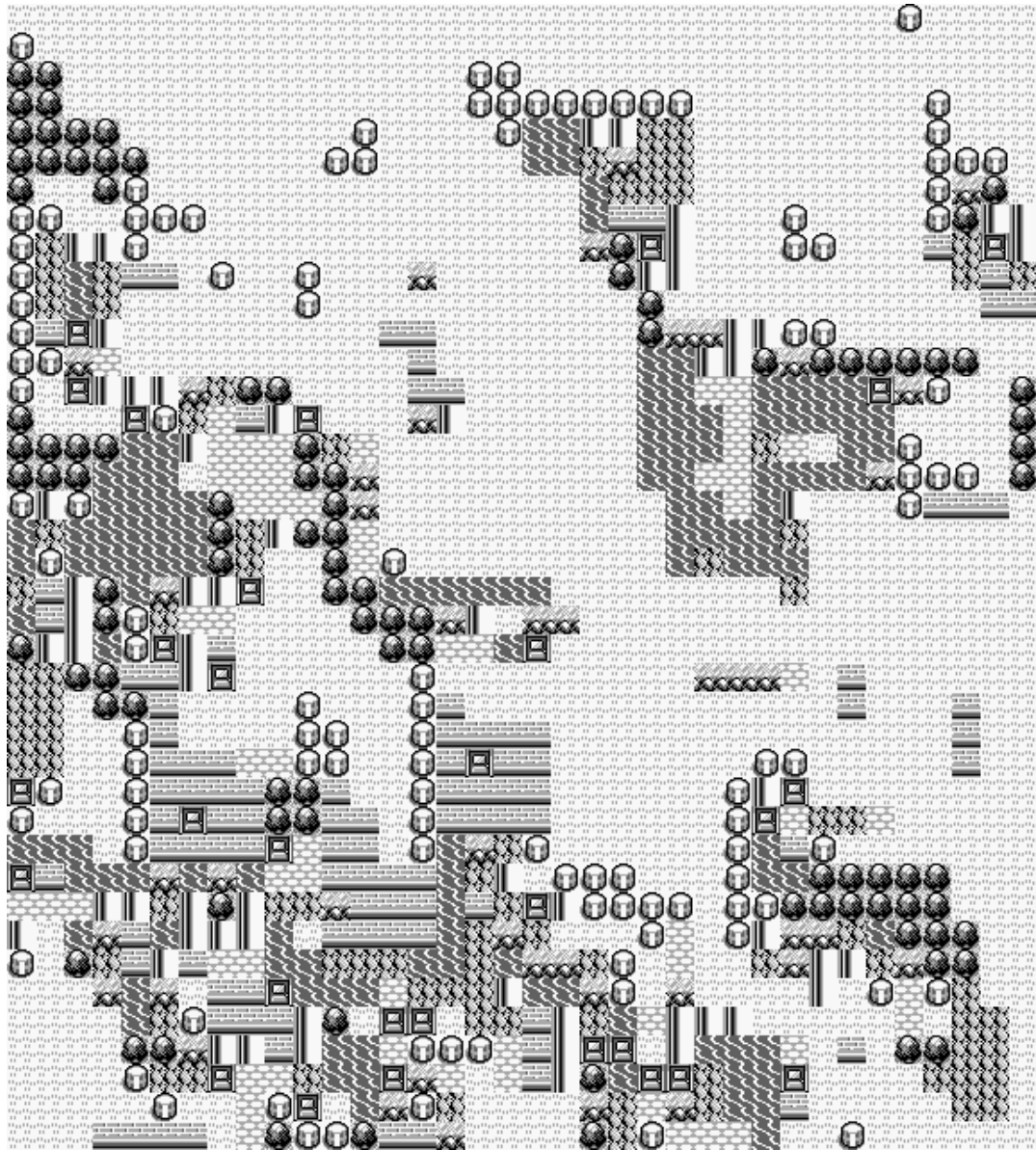
The generated maps do not make much sense since they only take one dimension into account.



## Two or more predecessors kernels

$$\text{kernel} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix};$$

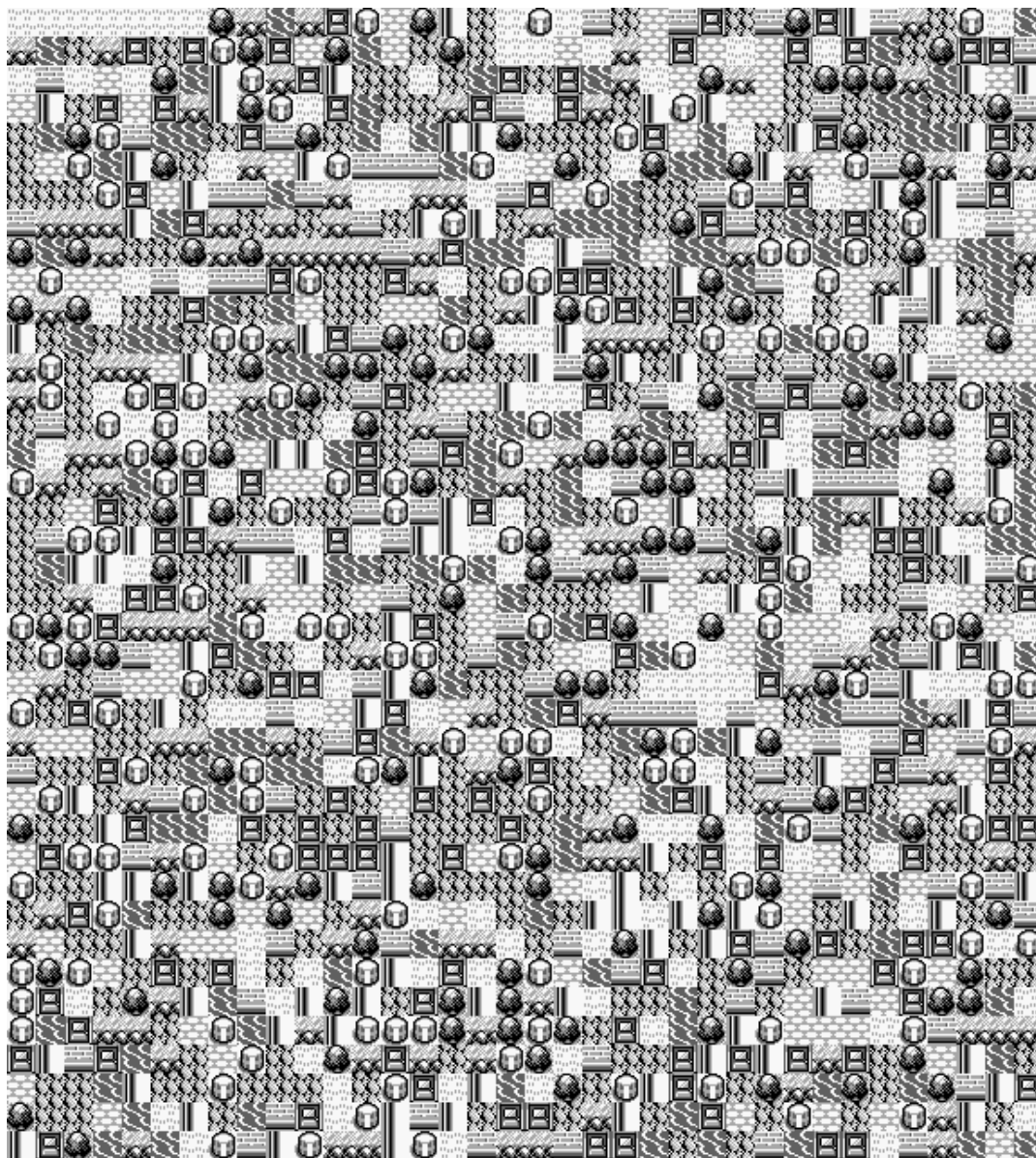
CreateMap[kernel]



Here we can already see an improvement since this kernel takes into account both dimensions. However the overall quality of the map is poor and we can clearly see it was not create by a human designer.

```
kernel =  $\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix};$ 
```

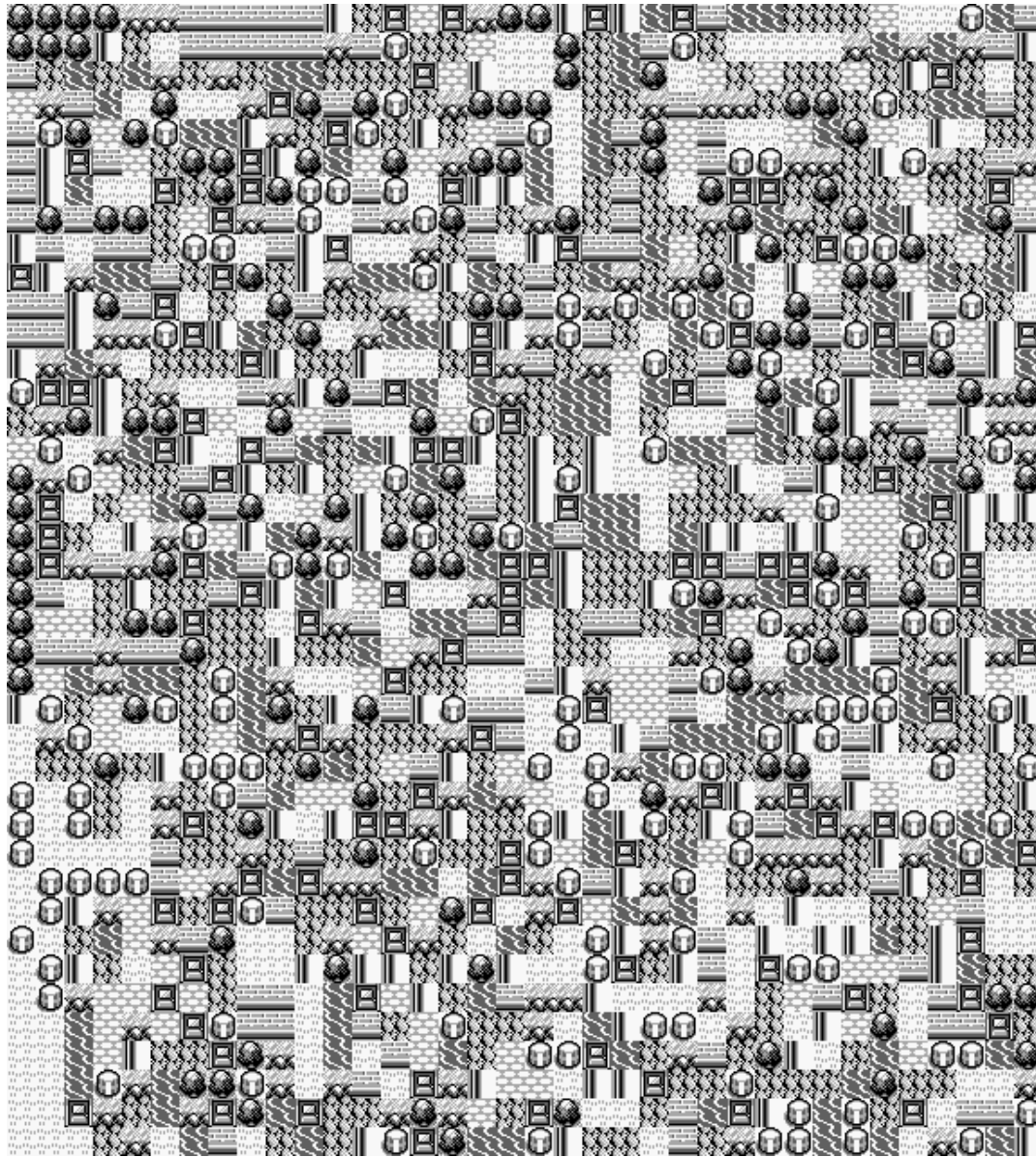
```
CreateMap[kernel]
```



As soon as we increase the dependencies the performance gets much worse, this can be explained by the fact that the number of combinations grows exponentially with the number of predecessors. This makes it very likely for a combination that has no associated data to be generated. This in turn forces the Markov chain to choose a tile at random.

```
kernel =  $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix};$ 
```

```
CreateMap[kernel]
```



Having a diagonal predecessor does not seem to improve the problem that was mentioned above.

## Conclusions

The performance degrades very quickly when the number of predecessors increases.

- The computational complexity grows exponentially with the number of predecessors
- The number of possible permutations of predecessors also grows exponentially. This has the effect of making it very likely for the Markov chain to generate a predecessor combination that has no associated data and as such choose a tile at random

We could solve these problems by either having more data or by reducing the total number of tiles as it would lead to lower number of possible predecessors.

## Next steps

Try to categorize into only two categories:

- Walkable space
- Obstacles

Find ways of expressing it into 3D