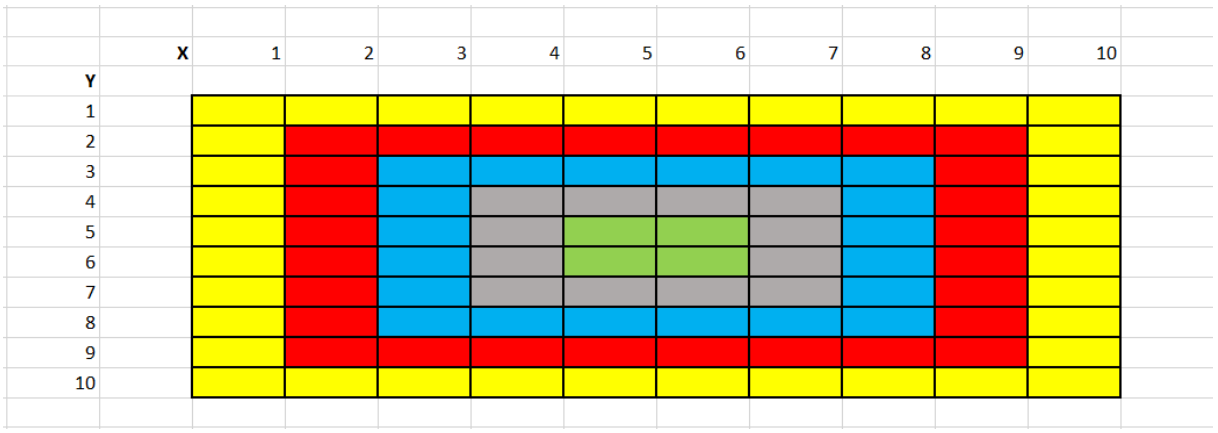
Design document for ”A LUA ASCII MapGenerator” (LuaLike)

by

Jesper S

The finished product and how it works:



Each rectangle is called a “chunk” and each colored part is called a “section”. Every chunk is connected to its own sections with open paths. The connections are between the closest chunks, I.E chunk (y=2, x =5) is connected to (y=2, x=4) and (y=2, x=6).

Every section is connected to its closest section(s) with two connections, one top or bottom and one right or left, I.E the grey section connects to the blue and the green sections. The connections between sections are not open, they have a closed door. If this where to be the basis for a game, you would find the keys for the grey section in the green section and so on.

There are right now two types of special rooms. A starting room (spawns in a random chunk in the green section) and one ending room (spawns in a random corner of the yellow section). The ending room has a smaller room inside it with a closed door and treasure. If this where to be the basis for a game, maybe it would open when you have killed a boss in each section?

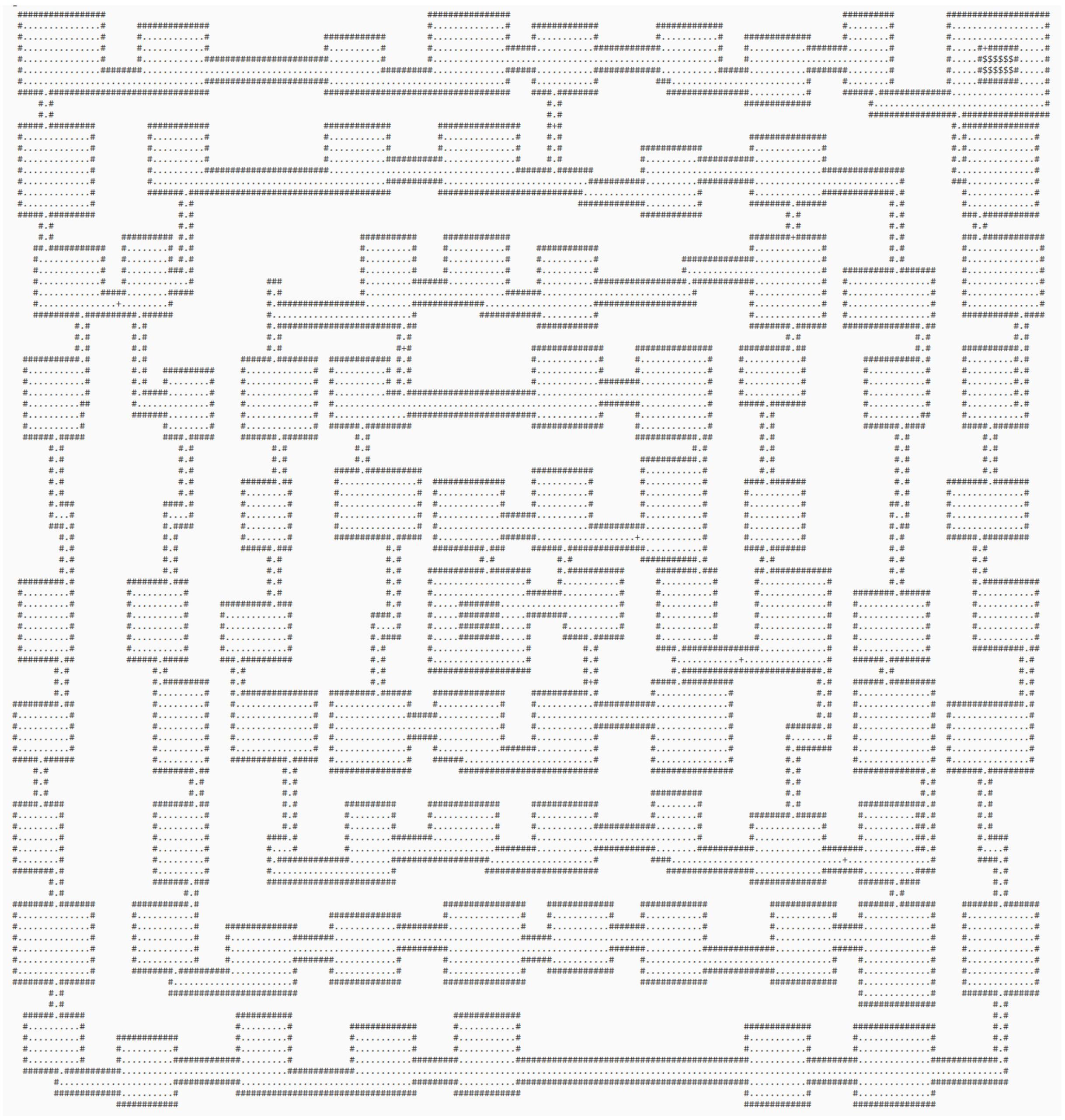
Otherwise, it spawns a room in each chunk that right now has a maximum height/width of 80% and a min of 40%. With the averages leaning to 60%.

Some chunks do not have a room and only has paths/connections. This is for variance. There is a 20% chance that a chunk will be empty.

In the map each character represents something, and this is what they mean: walls are (#), floors (.), closed doors (+), treasure ($) and empty( ).

The mapgenerator can make three different sizes of map, (height=100, width=200), (height=150, width=300), (height=200, width=400). The last one needs 4k resolution to render in the terminal.

Example of (height=100, width=200):



Challenges:

This was my first time creating any project in Lua so the most challenging part was really getting used to Lua and how I could/should use it.

I'm not sure my chosen software design of "classes" is good or even appropriate for Lua, but I chose to do something I was familiar with.

It was hard to debug and test for the most part. Lua just compiles like every time without questioning anything. I had a bug where I called on math.random() in for a function parameter and closed it of weirdly. What I meant to do was

fun(x, math.random(a, b), y, z)

What I did do was

fun(x, math.random(a, b, y, z)),

And it just accepted that. I took a long while to find that bug. Another bug was that I wrote "empy" instead of "empty" for a variable assignment, that was also ok for both Lua and the IDE.

I also confused x and y (that is why the project always refers to y first and x second) when creating the array at first. In the beginning the mapgenerator did square maps, so I did not at first discover my mistake. When doing 2D arrays you make the rows first and the columns second, and I did not think about that. To save time I choose to rename all variables instead of refactoring and remaking parts of the projects.

Continuing this project:

I would make more room, pathmaker and connectionmaker generator “classes” and call on them in the mapgenerator for variance. Maybe more special rooms, maybe you go from left to right instead of inside out in the sections? More sizes and more flexibility on how many chunks exist. And how the sections are done. So instead, the columns could be sections, or the rows, or a mix with what I have to create more variation.

I would also add guard clauses to everything and make unit test (or equivalent).

I would also want to rethink how I create paths in general. They could stop at the edge of the chunk (something like when the door is placed in connections) instead of going inside the other chunk so far. How it is done now does however create some more variance, so I’m not sure.

I would also change how I add walls to the map. Right now the addWalls() function is the reason you have to wait a second or two for a new map to be generated. Without it is almost instant.

Remaking it with the knowledge I now have:

I would make a class that creates a more reasonable array. So that increasing Y goes up in the map. I could do it so that the for-loop that creates it counts down from max to one instead. I would also make an “mapHandler” or something that takes x, y instead of y, x and that class would do the conversion.

But everything in the code is dependent on how it is made right now so that would be a lot of work for it just to make more sense.

I would also do an BSP generated map. With tree and leaf “classes”. The paths would be between sister leaves and connections between parent and child. Startingroom in the leftmost leaf and endingroom in the rightmost leaf. A recursive function that creates leaves until the min height/width is reached. And each leaf saves parent/child/sister leaves.

I elected not to do this early on in the project because I had a lot of bugs, and I was not sure I could make a recursive algorithm with “classes” or how to test/debug this. The mapgenerator that exists now is more reliant on design rather than code to be “good”.