

# Mario Level Generator Assignment: IMGD 4100

Mikel Matticoli and Diana Kумыkova

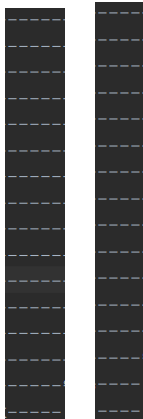
## Optional Extras implemented:

- (1) Manual analysis of files, (2) automatic (programmatic) analysis of files, (3) automatically created transition table

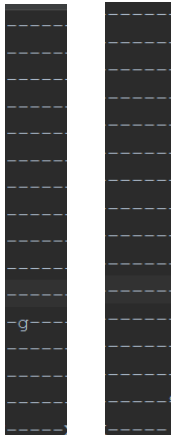
## Known issues:

## Data representation:

We chose to represent our data for each level within a text file that contains the following information about each “chunk” of the map: the chunk number, height (in characters), the chunk string/representation as written in the original level, and the probability that any of the other chunks in the file will come immediately after in the form of a row of probabilities. For example, if chunk 1 had a 50% chance of coming after chunk 4, then column 1 of the probability row for chunk 4 would contain the value of 0.50, etc. For the manual analysis of files chunks were taken as vertical slices of the map, and chosen roughly based on their “uniqueness” as judged by the person analyzing, but there was some leeway given in order to make the process more bearable. For example, the following 2 chunks were treated as similar enough to be considered the “same” for probability’s sake:



While the following 2 were considered unique and had their own probability rows:



Corresponding to the data in the text files are the data structures in the code: a hashmap of chunk strings keyed to id numbers, and a 2D array that mirrors the probability table that we created in class. The array is organized so the index matches the chunk ID specified in the hashmap, and the value at any given point equals the probability for the Y-coordinate chunk to appear after the X-coordinate chunk, i.e.  $[1][2] = .25$  means there is a 25% chance that chunk 2 appears after chunk 1 in the generated map.

The files analyzed manually were pulled from the hopper and notch folders.

**Reflection on strengths and weaknesses:**

An obvious weakness of our generator is that by only judging chunks by taking vertical slices we leave out the potential to create even more varied upper and lower halves of the map generated. The variety of topology in the output level is limited to vertical slices of the input levels, which breaks with a very small dataset. A benefit of this however is that it consistently generates playable levels - any given chunk will never be followed by a chunk that is not known to work in that order within an existing playable level.

**What we would improve if we had time:**

One limitation of our implementation is that the analyzer only processes a single file. Our first step given the time would be to allow multiple level files as inputs, so that more varied levels can be generated. This would allow us to synthesize patterns across multiple level generators and develop more complex and varied levels. An additional enhancement would be specifying chunk width. To limit scope, we decided the width of a chunk would be a vertical slice of width 1, but it would be interesting to see how the output would change with different sized chunks.