Cozy Caves

A D&D Dungeon Generator

Room Generation Module

Written By
Abdulrahman Asfari

Reviewed By
Gavin Lim
Gideon Wilkins

Technical Outline - Layouts

Why Layouts?

The idea behind layouts is that there will eventually be some tool to dynamically create layouts with unique scaling rules. This removes the need to create identical layouts of different sizes, but also requires a much more robust and logic-heavy system to support these custom layouts.

Ideally, in room generation there will be a pool of layouts to choose from. A random layout will be investigated, and the layout will continue scaling until it is either confirmed to be valid or invalid. If it is invalid, then remove it from the pool and reroll another layout. If the layout is valid, then combine all layout partitions and apply unscaled tiles on top, more on both of those terms later.

Layout Data Structure

Tags

A list of string tags that have no effect within the room generation module but can be used in other modules to add extensible behaviour.

Partitions

This is the most significant part of a layout. Essentially, a layout is just a wrapper for a list of partitions. All scaling computation will be done in partitions and then later merged into a room.

Tiles

There are two lists, one for scale exclusion tiles that are applied after scaling occurs, and a list for unscaled tiles that scaling is applied on top of.

There will be methods manipulating either the list of tags, partitions, or the two non-scaling tile lists. There will also be a method to fetch a room based on given dimensions and a leniency parameter. This is where partitions are managed and scaling methods are called.

Partition Data Structure

Scaling Configurations

Lock Ratio - Ensure that X and Y are scaled the same number of times, if any.
Lock X - Prevents any scaling from happening on the X axis.
Lock Y - Prevents any scaling from happening on the Y axis.
Scale Type X - Dictates if scaling on the X axis happens by a set amount or in multiples.
Scale Type Y - Dictates if scaling on the Y axis happens by a set amount or in multiples.
Scale Amount X - Amount to scale by on the X axis if scale type is by a set amount.
Scale Amount Y - Amount to scale by on the Y axis if scale type is by a set amount.
Scale Direction X - Dictates X anchor when scaling: positive, negative, or centre.
Scale Direction Y - Dictates Y anchor when scaling: positive, negative, or centre.

Other

Scaled Count X - Used to ensure lock ratio is upheld.
Scaled Count Y - Used to ensure lock ratio is upheld.
Edges Right - Tracks edges on the right of the partition.
Edges Left - Tracks edges on the left of the partition.
Edges Up - Tracks edges on the top side of the partition.
Edges Down - Tracks edges on the bottom side of the partition.
Max Encountered - Tracks largest X Y values encountered to determine total dimensions.
Min Encountered - Tracks lowest X Y values encountered to determine total dimensions.
Tiles - A map of tiles in the partition.
Scaled Tiles - A map of tiles that can be manipulated and reset.

The most important part of this data structure is that it will have a scale method. It will manipulate the scaled tiles map. It is given information about its parent layout as well as its position, in order to be able to override tiles in previous partitions, in order to emulate writing over a tile. There will also be a method to reset the scaled tile map.

Scaling Logic

Scaling By Set Amount

Edges need to be tracked when scaling by a set amount. This is so the method knows where to add new tiles, as only having the direction is not sufficient and will require iterating through the whole map each time.

- For each edge in the scaling direction, add **scale amount** number of tiles in that direction.
- For each created tile, remove tiles in previous partitions with the same position.
- If the direction is centre, then do this for both directions in the axis but half the scaling amount. Scaling amount must be even for this to be a valid scaling operation.

Scaling in Multiples

- Find origin X/Y using either the minimum or maximum encountered value in the partition, depending on the scaling direction. If the scaling direction is centre, then get all values on the axis and determine the median value (can be two origins here). Store the number of values on the axis.
- Starting with the furthest tiles from the origin, move all tiles in the partition with a non-origin value away from the origin, by an amount equal to the number of values closer to the origin (inclusive) than the current value.
- For each tile in the partition before this stage, add a tile 1 position away, in the scaling direction.
- For each created tile, remove tiles in previous partitions with the same position.