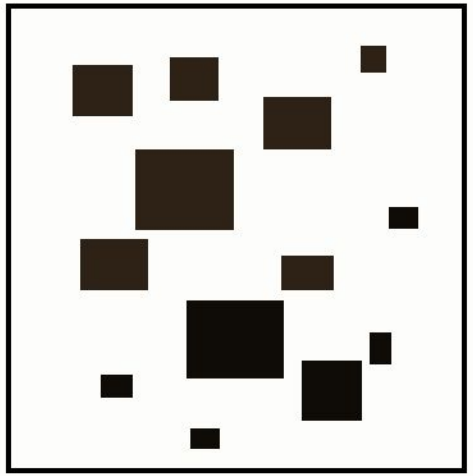
Dungeon Generation With Properties of Wholeness

Dana Stokes & Eiler Poulsen

Problem statement

Attempting to construct dungeons based on a subset of the 15 Properties of Wholeness.

More specifically, we are interested in generating a high-level overview of a dungeon, (showing mainly only positive and negative space).



Literature

Christopher Alexander’s research: 15 Properties of Wholeness

Current research focuses on city generation (city, building, and floor plan)

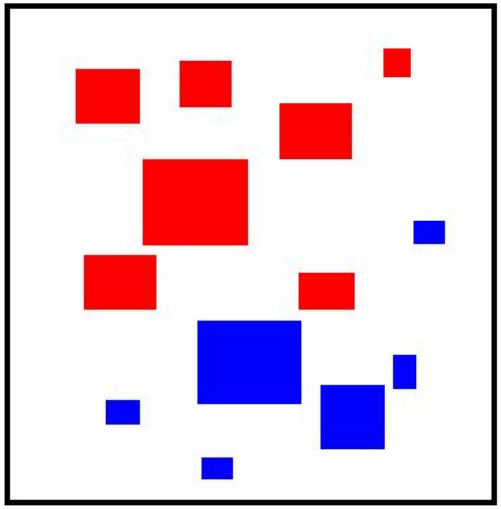
Nothing found focusing on dungeon generation

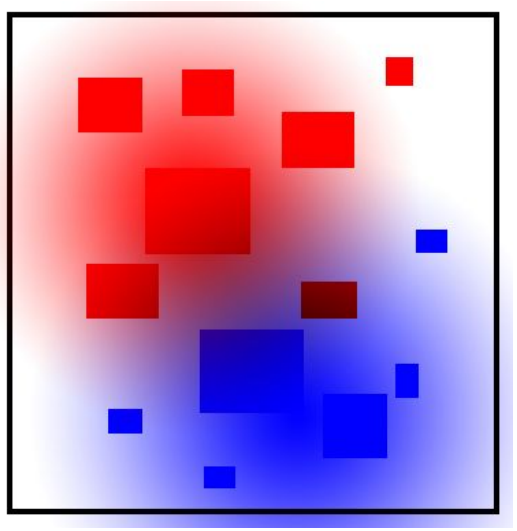
Properties

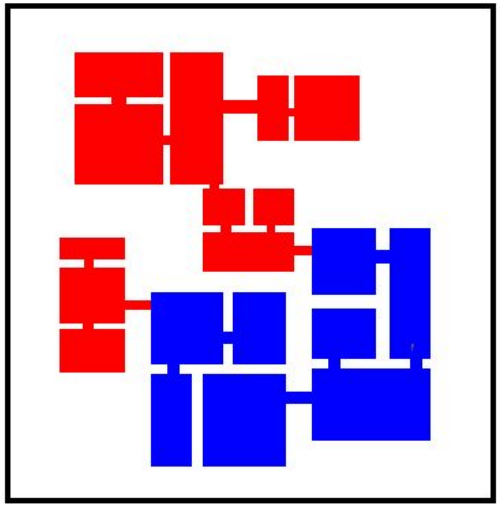
- Strong Centers
- Good Shape
- Level of scale
- Alternating Repetition
- Gradients

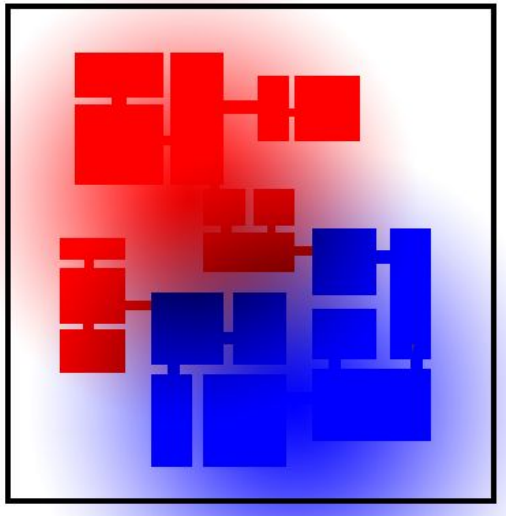
(Excluded: Local symmetry

Applicable? Non-separateness)









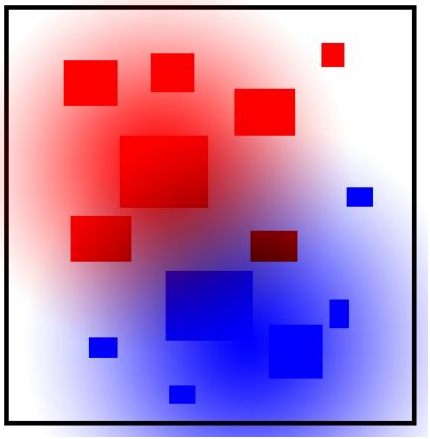
Algorithm overview

1. Generate **hotspots**
2. Use hotspots to determine **heatmap**
3. Use heatmap for **container** generation
4. Generate **rooms**
5. (Generate **corridors**)

Hotspots

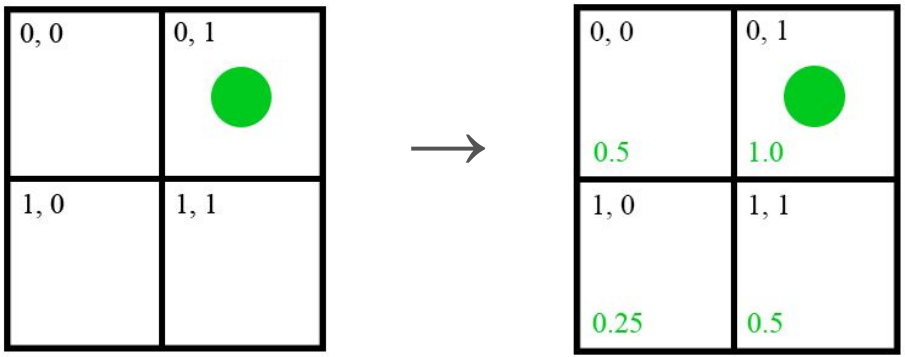
Hotspots have:

1. Location
2. Room generation characteristic such as color
3. Function to determine its influence(from 0-1) at a location (x,y).
    - Eg.: Heat falls off linearly according to the distance to the hotspot.
        - Specifically: *heat * (1 - (distance/maxdistance))*
    - Other options such as quadratic falloff



Heatmap

Call the hotspot's heat function for each location and save the result.



Now we can just look up the characteristics of a location instead of having to query all of the hotspots every time.

Containers: Inverse transform sampling

Probability mass function (=a discrete probability density function), normalized to 1

1	2	3	4
0.25	0.50	0.125	0.125

↓

(A discrete) cumulative distribution function

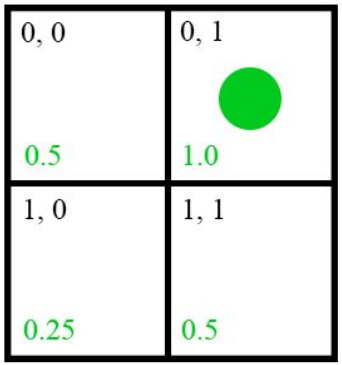
1	2	3	4
0.25	0.75	0.875	1

Sampling:

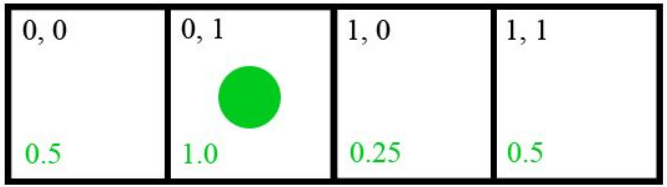
.1	→	1
.2	→	1
.25-.75	→	2
.75-.875	→	3
.875-1	→	4

Generating random numbers between 0.0 and 1.0 returns outcomes 1, 2, 3, or 4 at a frequency in accordance with their probability mass.

Container location by inverse transform sampling

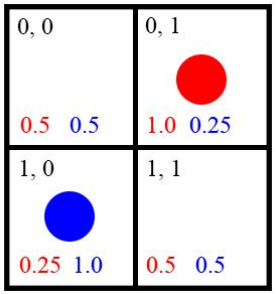


=

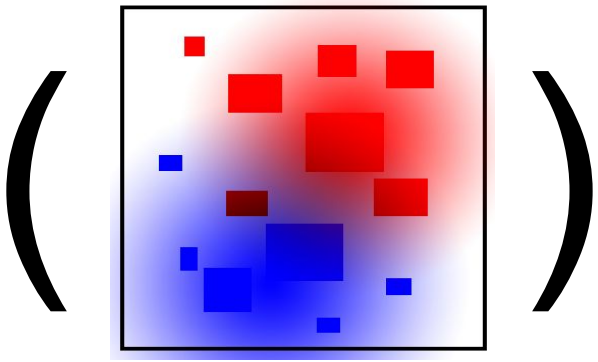


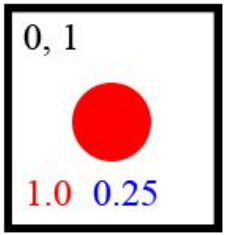
Probability mass function
(see previous slide)

Container characteristics by inverse transform sampling (eg. color)

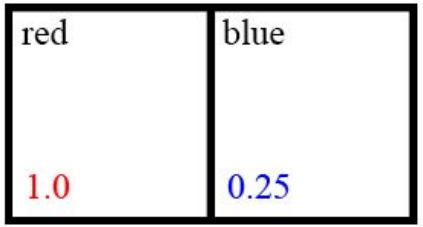


Heatmap for (e.g.) color, with hotspots of red and blue.





We already know the location, and just want to sample a color given that location (in this example, (0, 1)).



Probability mass function.

Rooms

Stochastic space partitioning, as in the book: recursively divide the space randomly down to a minimum size, and place rooms within those subspaces.

Naively implementing this makes for ugly rooms. Tweaks were made so rooms always occupy at least 1/2 of the space on the shorter axis and ⅓ of the space on the longer axis.

Good Shape.

(Corridors)

Construct minimum spanning tree with Kruskal’s algorithm: adds edges from shortest to longest iff the edge doesn’t create a cycle, stops once all vertices are connected.

Potentially add some more edges so it’s not as sparse.