Minecraft Settlement Generator V2 Description

Alexaline Thomas[1], Eduardo Hauck[2], Claus Aranha
University of Tsukuba, Department of Computer Science, Japan

UrbanSettlementGeneratorV2[3] is based on the generator[4] made by Eduardo Hauck (ehauckdo) for the 2019 iteration of the competition. It aims at generating modern cities in one biome of a Minecraft map. It generates houses, apartment buildings and greenhouses connecting all of them with roads. The buildings are generated according to a downtown/outskirts division, with apartment buildings placed towards the center and the other structures placed towards the edges of the map.

Houses have their wall separated by pillars placed in their corners and have pitched roofs. The blocks used for their construction depend on the biome in which they are built. They have one oak door as the entrance on one random side, and glass windows on the walls perpendicular to the door wall. Houses are furnished with carpets, a table in the center of the room, a chandelier on the ceiling, and a bed, bookshelf and couch in the corners.

Greenhouses are small glass dome with a random cultivated plant inside (well watered). The glass dome is made of white stained-glass and, depending on the biome other glass colors for aesthetic purpose. They can be accessed from both the front and back thank to three per four blocks opening. They exist to ensure that the settlement inhabitants have a minimum food supply.

Apartment buildings have clay walls (of a random color), 4 to 10 floors, with one or two apartments on each floor and a furnished roof. The entrance to each building leads to a small corridor with a door to a small apartment and, a stairwell for reaching other floors and the building lobby (with mail boxes and small plants). The apartments' furniture follows a similar fashion as in the houses, and windows are on the opposite wall to the building entrance. Some apartments have small balcony build with local blocks. The roof features some tables, a small garden and an antenna.

The generator takes a number of steps in order to generate the city. First, it takes the space received in the perform function and generates a height map, a 2D matrix containing the height of the first valid ground block for all coordinates x, z of the map. It then analyses the surface of the map in order to gather information about it (what are the ground blocks and what type of tree and plants are in the area) and determine in which biome is the area and what are the local tree type (those two

[1] thomas.alexaline@gmail.com
[2] eduardohauck@gmail.com
[3] https://github.com/Megacruxis/UrbanSettlementGenerator
[4] https://github.com/ehauckdo/UrbanSettlementGenerator

parameter will determine the blocks used for the construction of the city’s structure). It then gets the width and depth of the space, and selects a percentage of the central area to become the city center. The remaining areas are divided into 4 sections to become neighborhood.

The center and the 4 neighborhoods go through binary or quadtree (chosen randomly each time) space partitioning 100 times and the generated partitions are added to a list. Each partition becomes a building lot, and the ones that have water, lava, etc in their perimeter or do not reach the set minimum size are eliminated as invalid. The list is then ordered according to the steepness of the lots. The steepness is calculated as the sum of differences between the most occurred height (according to the height map), and the height of every other block in the lot. Therefore, a steepness of 0 means a building lot with a flat surface.

The construction starts by pulling a building lot from the list, checking if it does not intersect with any previously pulled lot, and building a structure on it. This procedure is repeated until reaching a certain minimum number of lots for each of the 5 sections.

To generate a building, the first thing to be done is to check whether the lot terrain is flat. If not, earthworks are carried out to ensure that the perimeter is at the same height level. This is done by finding the most occurred height, and then flattening the entire lot to match that. The blocks (the surface and the underground blocks) used for the flattening depend on the settlement’s biome.

Once a lot is flattened, generation proceeds at that height. The generation of buildings, greenhouses and houses are mostly hardcoded. For the houses, it is as follows: first, a random width and depth between a certain range, and the maximum height of the house are set. Then, the land is cleared changing trees etc to air blocks. The generator try to keep as much tree as possible by only deleting those which have some of their blocks (trunk or leaf) inside the area where the house is going to be constructed. From that, floor and walls are generated, an orientation (N/S/E/W) is decided based on the position of the house, always facing the center of the map. The orientation will dictate the location of the entrance door, the entrance to the lot at the edge of it where roads will "connect", as well as the position of the windows. The orientation also sets the direction of the pitched roofs. Once the house is completed, the interior is furnished. For apartment buildings, generation happens similarly.

Once all the buildings have been placed, the last step is connecting them with roads. To perform this, we consider each lot as a node in a graph, and we create a minimum spanning tree to connect them. Distances between nodes are calculated with Manhattan distance. Once we have the minimum spanning tree, we use A* to actually find the path to generate the road. The heuristic function that gives the cost between the current and the target point is the Manhattan distance. The cost function between each node is $1+n^2$, where n is the difference of height between the current block and the next. We employed this function to try and find a path that is not too steep between two points (e.g. going around a mountain instead of climbing it).

The A* returns a sequence of coordinates 1 block wide where the pavement is to be generated. We complete the road on both sides of this path to make it 3 blocks wide. When doing that, we verify if the neighbouring block is not an invalid block (e.g. water), and if it is on top of air, we fill the blocks below recursively until getting to a valid ground block.