Procedural Generation of Urban Environments with Minimal Urban Planning Rules

Dmitry Novozhilov

January 2026

Abstract

Procedural generation is a powerful technique for synthesising complex content from compact rules and random seeds. In the context of urban environments this approach allows one to create vast, unique cityscapes on demand, useful for games, simulations and urban planning studies[17†L19-L27]. We present a modular prototype for procedural city generation that transforms high-level parameters—population, numbers of hospitals and schools, a primary transport mode and a random seed—into a complete three-dimensional model of a city. Our generator adheres to a handful of basic urban planning guidelines, such as minimum green space per inhabitant[26†L7-L10] and the distribution of educational and healthcare facilities[25†L825-L834]. The implementation employs C++ for performance and a Python wrapper for ease of use, and includes integration tests verifying determinism and rule compliance. This paper summarises the design, algorithm and implementation of the generator.

1 Introduction

Cities arise from a complex interplay of geography, economics and societal forces. Modelling such complexity procedurally requires simplifying assumptions and rules that nevertheless produce believable results. Prior work on procedural city generation has explored grid-based and agent approaches[17†L17-L25], multi-zone models inspired by the Burgess concentric zone theory[18†L67-L75], and noise-driven growth akin to terrain generation in Minecraft[21†L92-L100]. Our goal is to design a generator that, while simplified, produces cities that obey some well-known urban planning principles and remain reproducible under a fixed seed.

2 Input Parameters

The generator accepts several high-level parameters:

- **Population.** The number of inhabitants in the city influences the spatial extent of development. A larger population implies a larger buildable area and more infrastructure. The gross density is assumed implicitly, but can be tuned via the grid size and radius.
- **Number of social facilities.** Users specify how many hospitals and primary schools should exist. These facilities are distributed across residential and commercial zones to ensure coverage; in practice one hospital per about 100 000–120 000 people is recommended[14†L318-L326] and one primary school per 6 000–10 000 residents[25†L825-L834].
- **Primary transport mode.** The dominant mode of transport (car, transit or walking) influences the road network layout. In this prototype we generate a simple cross and two ring roads, but the parameter is provided for future extensions (e.g. generating a transit line when the mode is “transit”).

- **Random seed.** Procedural content should be repeatable; given identical parameters and a seed, the same city must be produced. Our generator uses a 32-bit seed to initialise all random processes.
- **Grid size and radius fraction.** The city is discretised into a square grid of `gridSize` × `gridSize` cells. A radial boundary defined by `radiusFraction` of half the grid extent delimits the developed area; outside cells remain undeveloped (zone "None").

3 Urban Planning Rules

While a complete simulation of urbanism is beyond scope, we encode several basic planning rules drawn from the literature:

Green space per inhabitant. World Health Organization guidelines recommend at least 8 m^2 of accessible green space per inhabitant in cities[26†L7-L10]. Our generator calculates the total area of green cells given an assumed cell area (100 m × 100 m) and the population; if the initial zoning yields too few green cells, random residential or industrial lots are converted to parks until the target is met.

Education and health. Access to schools and healthcare should be ubiquitous. Primary schools are typically located within 500–1000 m of residences and serve roughly 6 000–10 000 people[25†L825-L834]. Hospitals should serve about 100 000–120 000 people[14†L318-L326]. We simply place the user-specified number of facilities at random within residential and commercial zones; more sophisticated placement (e.g. ensuring maximum distance constraints) is a topic for future work.

4 Algorithm Overview

The generation pipeline follows the steps below:

1. **Initial zoning.** Each grid cell inside the radial boundary receives a noise value from a multi-octave hash-based noise function. The value is mapped to one of four zones: residential, commercial, industrial or green, based on predefined thresholds. Cells outside the radius remain undeveloped. Building heights are sampled stochastically from zone-specific ranges (e.g. 2–6 floors for residential, 5–20 for commercial).
2. **Green space enforcement.** The number of green cells is compared to the required amount derived from the population and cell area. If insufficient, random residential or industrial cells are converted to parks until the requirement is satisfied.
3. **Facility placement.** Hospitals and schools are placed by sampling random eligible lots (residential or commercial) without replacement until the requested counts are met. Facilities are flagged so tests can verify their counts.
4. **Road network.** The road system comprises a cross of orthogonal boulevards through the city centre and two concentric ring roads at 50% and 90% of the developed radius. Each road is represented as a set of line segments; more complex networks could be generated following the radial or grid patterns discussed in prior work[17†L17-L25].
5. **Output.** The final city is serialised into two formats: a Wavefront OBJ mesh where each non-green lot becomes a cuboid, and a JSON summary containing high-level counts for testing and analysis.

5 Implementation

The core of the generator is implemented in modern C++ and organised into three main classes:

- `Config` – holds all user-supplied parameters.
- `City` – stores the resulting grid of buildings, the list of facilities and the road segments. Methods allow for OBJ and JSON serialisation.
- `CityGenerator` – exposes a static `generate` method that performs the pipeline described above using the configuration.

Noise is generated using a simple hash of grid coordinates and the seed. Although less smooth than Perlin or Simplex noise, this method has no external dependencies and suffices for the prototype. A fractal sum of four octaves introduces multi-scale variation. The generator uses the `std::mt19937` engine for deterministic random choices, ensuring repeatability. A command-line interface built in `main.cpp` parses arguments and writes the outputs to a specified directory, while a Python wrapper simplifies invocation from scripts.

6 Testing

Integration tests written in Python validate key properties of the implementation. The tests compile the C++ code, run the generator with various configurations and inspect the JSON summaries. They verify that identical seeds yield identical summaries (determinism), that the counts of hospitals and schools match the configuration, and that the green space allocated meets or exceeds the recommended minimum per capita[26†L7-L10]. Future tests could enforce maximum walking distances to facilities and compare generated road networks against reference patterns.

7 Conclusion and Future Work

We have presented a modular procedural city generator that produces reproducible urban models from a handful of parameters. By embedding simple urban planning rules—minimum green space, basic facility provision and a rudimentary road network—the generated cities avoid some of the pathologies of purely random constructions. Nevertheless, there is tremendous scope for refinement: incorporating more realistic noise, simulating growth dynamics[22†L23-L39], enforcing service access constraints, and adding variety in building geometry. We hope this prototype serves as a foundation for further experimentation and as a teaching aid for procedural content generation.

References

[1] Sahinoglu, B. Y., & Celikcan, U. (2022). A grid-based multi-zone Burgess approach for fast procedural city generation from scratch. *Mugla Journal of Science and Technology*, 8(1), 8–18.[17†L17-L25][18†L67-L75]

[2] World Health Organization. (2024). Guidelines for healthcare facilities. *BMJ Open*, 14, e000000.[14†L318-L326]

[3] MDPI Sustainability Journal. (2023). Role of urban planning standards in improving lifestyle in a sustainable system. Sections on infrastructure planning standards and green space requirements.[25†L825-L834][26†L7-L10]

[4] Cybrancee Blog. (2025). How Minecraft terrain generation works. Retrieved from the author's website.[21†L92-L100]