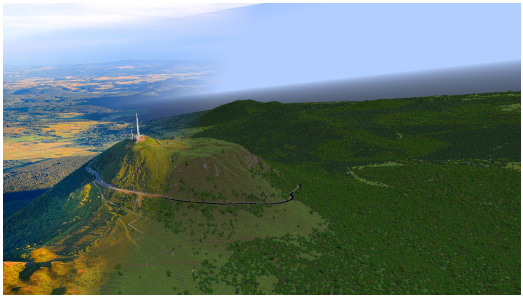
Francegen

Recreating France in Minecraft, block by block: a study of GIS, Open Data, and AI-assisted development

Daniel "defvs" THIRION

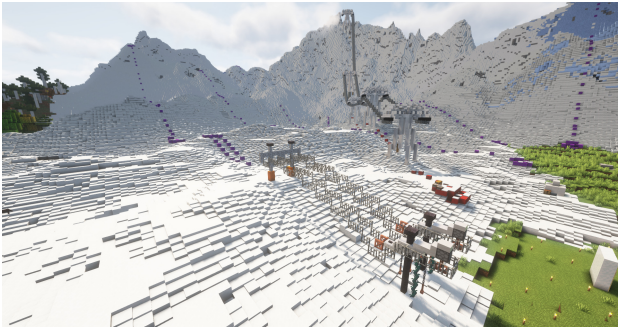


Agenda

Motivation

From Idea to Terrain

- I've always wanted to recreate a massive ski resort in Minecraft.
- But Minecraft's default terrain makes mountains small and slopes short.
- The in-game scale simply doesn't match real-world topography.



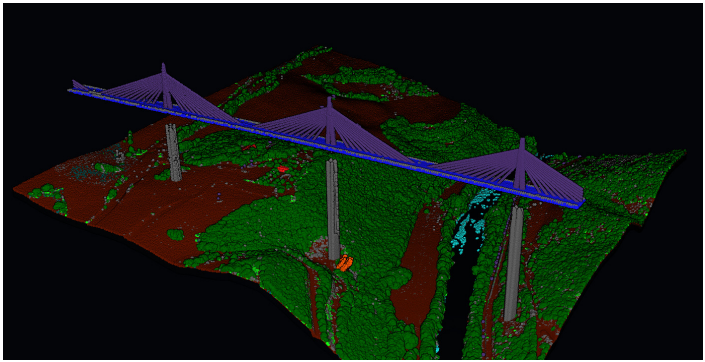
My first ski resort, using Terralith generation mod.

Problem: Tiny Minecraft Mountains

- Real ski resorts span tens of kilometers and thousands of meters of elevation.
- Vanilla Minecraft height is ridiculously small: 320 blocks tall worlds.
- A convincing 1:1 experience needs real elevation data or custom generation.

Trigger: IGN LIDAR-HD

- Discovered IGN's LIDAR-HD project: high-resolution LIDAR for all of France.
- 50 cm DEMs and dense point clouds available under a free open-data license.
- That moment of realization: *"Wait, I can use this for so many things!"*



Viaduc de Millau, France, from LIDAR-HD, © IGN

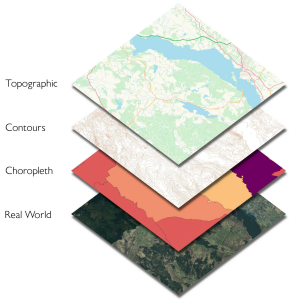
Preliminaries

Minecraft in One Slide

- Sandbox game made of 1 m blocks, with virtually infinite procedural worlds.
- Worlds are stored in chunks and regions; terrain is driven by noise-based generators.
- Perfect playground for geodata, if we can control terrain, biomes, and structures.

GIS Basics

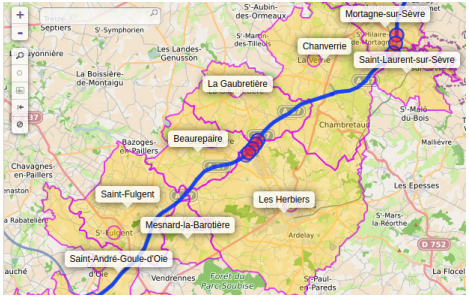
- Geographic Information Systems (GIS) manage spatial data: rasters, vectors, and point clouds.
- Raster DEMs: grids of elevation values (our mountains and valleys).
- Vector data: roads, rivers, buildings, and land-use polygons.
- Point clouds: dense 3D samples capturing roofs, trees, and terrain details.
- Tools and libraries like QGIS make it possible to inspect, slice, and validate these datasets.



Overpass API and QL

- Overpass API is a query language and endpoint for extracting OpenStreetMap data.
- Overpass QL lets you filter by tags and geometry: roads, rivers, buildings, POIs, and more.
- Queries are small “programs” that describe which ways/nodes/relations to return.
- Example: `way["waterway"="river"];` selects all river ways in the current bounding box.
- francegen uses Overpass QL to pull vector data that later gets converted into blocks.

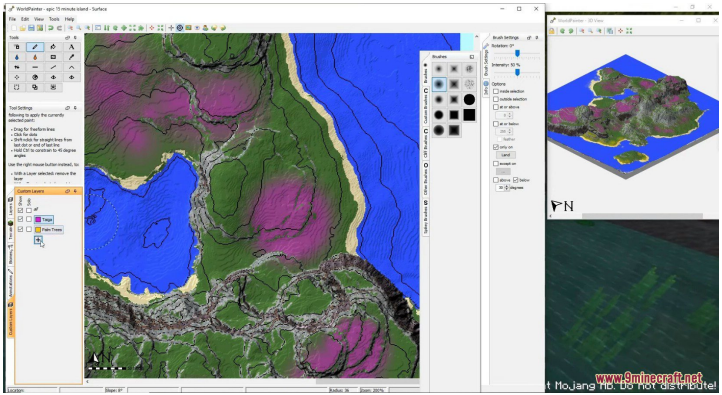
```
area
  [admin_level=6]
  [name="Vendée"]->.dept;

// Tracé de l'A87 en Vendée
way
  [highway]
  [ref="A 87"]
  (area.dept);
out geom;

// Communes traversées
>;
is_in;
rel(pivot)
  [boundary=administrative]
  [admin_level=8]
  (area.dept);
out geom;

{{style:
relation {text:name}
}}
```



First Tests

First Prototype: WorldPainter

- Started with WorldPainter, a GUI tool to design Minecraft worlds.
- Imported IGN DEM heightmaps to generate realistic terrain.
- Painted biomes manually from satellite imagery to approximate real landscapes.



Result: Promising but Painful

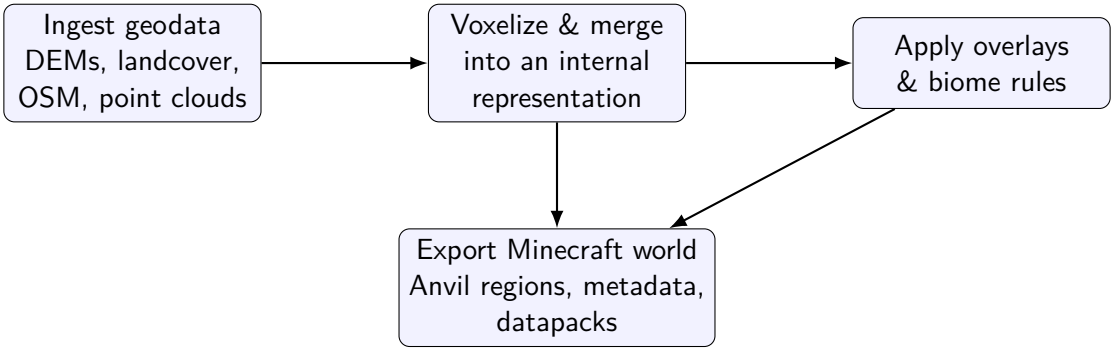
- Already looked surprisingly good: recognizable mountains and valleys.
- But biomes and details were tedious to paint by hand.
- World generation for a single region could take 12+ hours.
- Clear conclusion: this needed to be automated and reproducible.

Implementing francegen

Design Choices

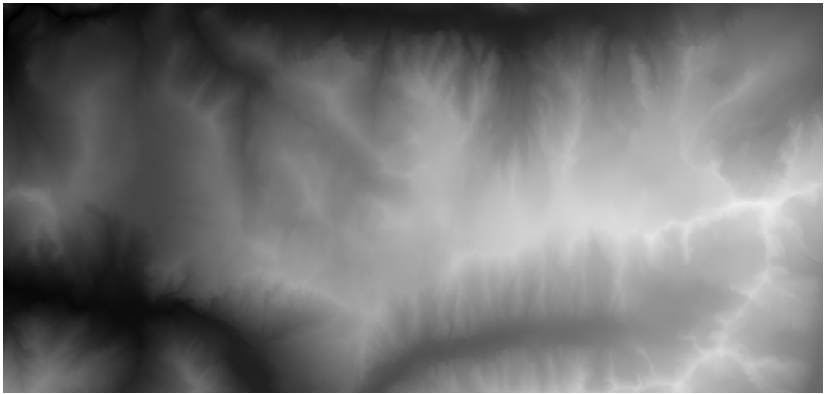
- Language: **Rust** — efficient, safe, and with a rich ecosystem (crates).
- Core crates: `fastnbt`, `fastanvil`, to work with Minecraft region files; `tiff` and `geo` for GIS; `rayon` for easy multithreading implementation.
- CLI-first design to make batch runs and scripting easy.
- Not only for France, but France first. Every aspect fully configurable with JSON files.
- Coding assistant: OpenAI Codex VSCode extension and GPT-5-Codex series of models.

High-Level Workflow



Terrain from DEMs

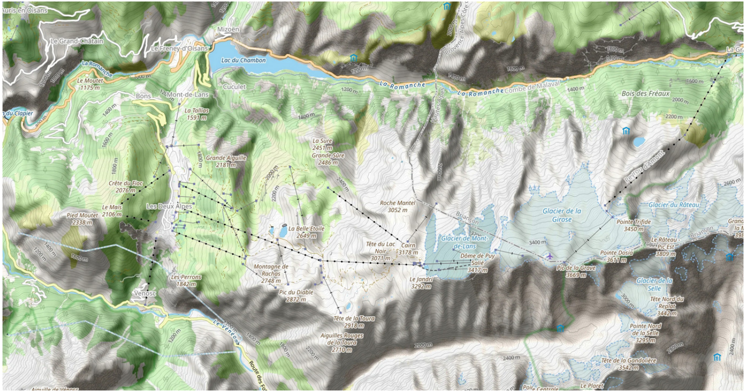
- Input: GeoTIFF heightmaps in a shared projection (e.g. LAMB93).
- Resample and normalize elevations to Minecraft's XYZ coordinate range.
- Generate base terrain and cliffs at 1:1 horizontal and vertical resolution.



Les 2 Alpes heightmap, © IGN

OpenStreetMap Data

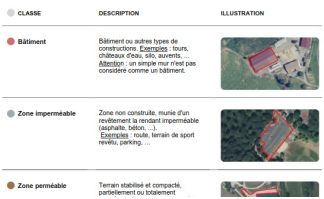
- Download OSM data via the Overpass API using Overpass QL queries.
- Extract roads, rivers, lakes, boundaries, and building footprints.
- Map these features to block palettes and structures in the world.

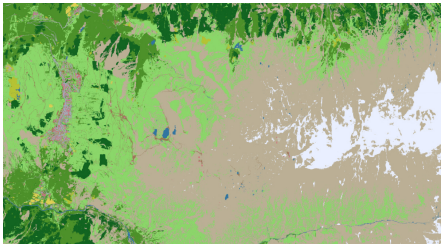


Les 2 Alpes on OpenStreetMap, © OpenStreetMap Contributors

Landcover and Biomes

- Fetch WMTS tiles to obtain landcover and landuse information.
- Match tile colors to semantic classes (forest, urban, water, snow, . . .).
- Convert landcover classes into Minecraft biomes and surface blocks.

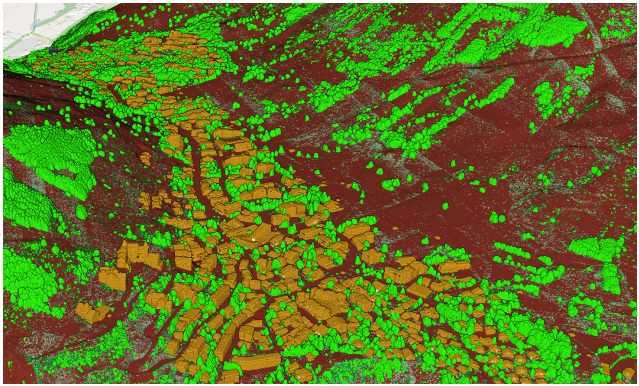




Les 2 Alpes landuse, © IGN

Point Clouds to Buildings

- Ingest COPC point clouds to capture detailed 3D surfaces.
- Voxelize points into columns to approximate roofs, facades, and trees.
- Use point clouds directly when OSM building heights or shapes are missing.



Les 2 Alpes LIDAR-HD, © IGN

Overlays and World Export

- Apply a stack of overlays (layers) to combine terrain, roads, water, and structures from all sources available.
- Write Anvil region files chunk by chunk to disk.
- Finalize the world: `level.dat`, metadata, and a datapack to tweak generation and allow 4096-block-high worlds.

Configuration and Customization

- Everything is customizable through JSON configuration files.
- Top and bottom layer blocks, biomes, for each possible layer gathered from the sources.
- Example: fetch `way["waterway"]` from OverpassQL and paint resulting ways with water, width of 2m.

```
],
"osm": {
  "enabled": true,
  "bbox_margin_m": 500.0,
  "layers": [
    {
      "name": "waterways",
      "geometry": "line",
      "width_m": {
        "default": 4.0,
        "min": 2.0,
        "max": 18.0,
        "sources": [
          { "key": "width" }
        ]
      },
      "layer_index": 0,
      "query": "(way[\"waterway\"~\"^(river|stream)$\"]({{bbox}}););",
      "style": {
        "surface_block": "minecraft:water",
        "top_thickness": 4,
        "subsurface_block": "minecraft:dirt"
      }
    },
    {
      "name": "water_polygons",
      "geometry": "polygon",
      "layer_index": 0,
      "query": "(way[\"natural\"=\"water\"]({{bbox}});relation[\"natural\"=\"water\"]({{
```

Various challenges and Fixes

- **“The world is flat, no trees!”**
  Mark chunks as not fully generated so vanilla Minecraft populates trees and features.
- **“Random springs and lava lakes on the surface.”**
  Use a datapack to adjust or disable unwanted vanilla features.
- **“OSM buildings are too simple and often lack heights.”**
  Fall back to LIDAR point clouds to recover realistic building volumes.

Benchmarks

Hardware: AMD Ryzen 9 3900X, 64 GB DDR4, RTX 3070

- 10 km × 10 km **Puy de Dôme**:
  - ∼45 minutes offline generation with francegen.
  - ∼15 minutes additional generation with Chunky in-game.
  - ∼700MB
- 20 km × 8 km **Les 2 Alpes**:
  - ∼2.25 hours offline generation.
  - ∼2 hours online generation
  - Taller mountains, 600m∼3600m, more blocks to write.
  - ∼6.7GB
- 4 km × 4 km **Paris Champs-Elysées**:
  - ∼7 minutes offline generation
  - ∼4 minutes online generation
  - Small mountains. Main bottleneck: lots of buildings and LIDAR points
  - ∼120MB
- Performance is mostly bounded by network, IO and memory.

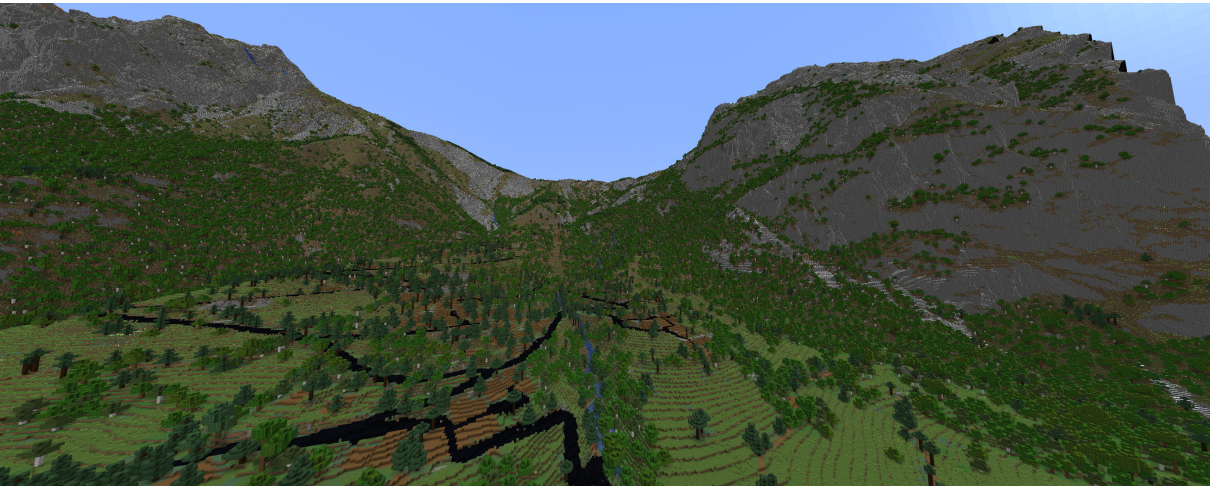
In Minecraft

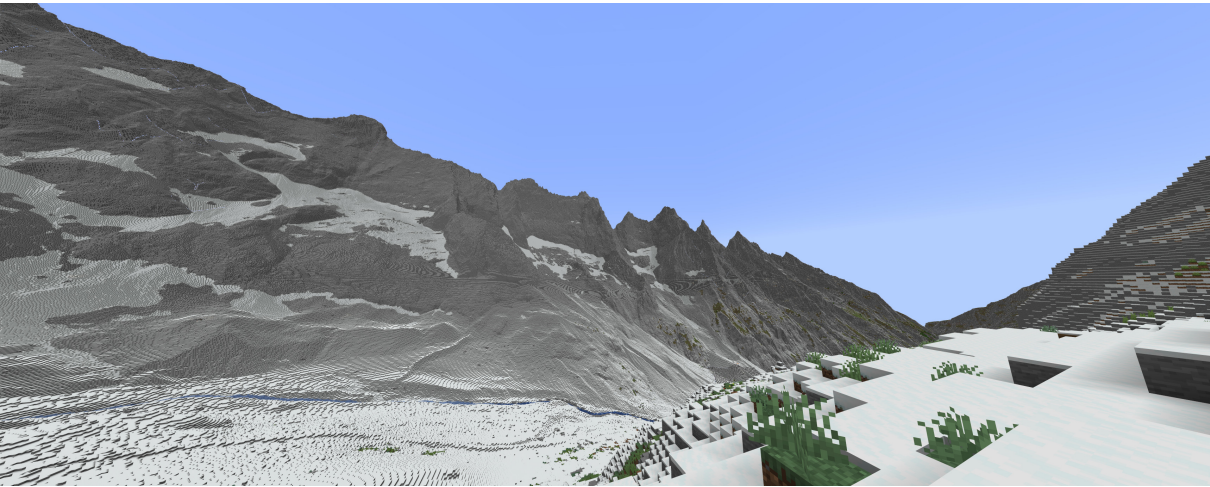
Mods and Environment

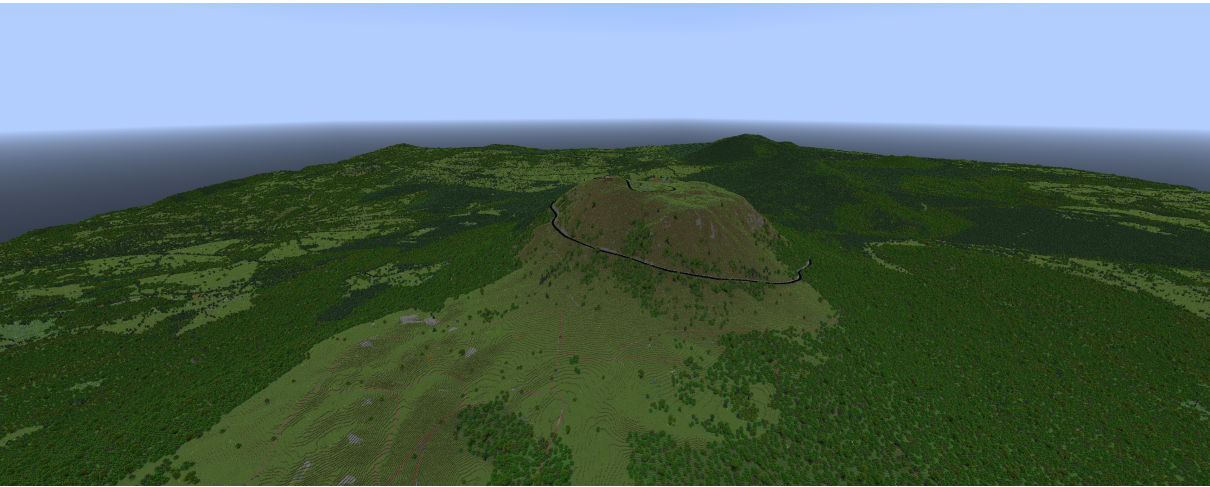
- Mods: Voxy (LODs), C2ME (chunk performance), Chunky (distant world generation).
- Hardware: AMD Ryzen 9 3900X, 64 GB DDR4, RTX 3070.
- Goal: be able to locate myself in the world from my own knowledge only.

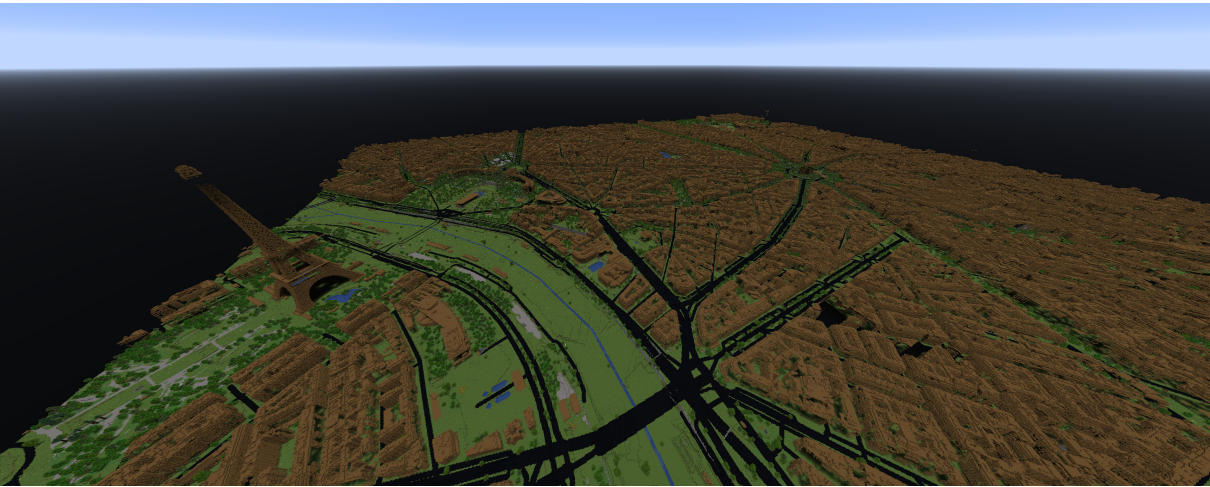
Voxy: `https://modrinth.com/mod/voxy`
C2ME: `https://modrinth.com/mod/c2me-fabric`

Chunky: `https://modrinth.com/plugin/chunky`









Exploring Les 2 Alpes (Live Demo)

- Live demo

AI-Driven Development

Toolset and Workflow

- Coding assistant: OpenAI Codex VSCode extension and GPT-5.1-Codex.
- Low budget: $25 "ChatGPT Plus" monthly subscription.
- Workflow: describe intent, generate code, run, debug, iterate.

Why Rust Works Well with LLMs

- Rust has a large corpus of high-quality open-source code.
- Strong type system and compiler errors provide rich feedback loops.
- Result: fast iteration cycles with surprisingly robust output.

The Importance of Context

- You still need to know what you want to build: you become the project head, not the developper.
- For obscure formats (like Minecraft's internals), good documentation is crucial.
- Example: pasted chunk format details from the Minecraft wiki into the prompt.
- The LLM can write code, but it can't run Minecraft, explore worlds, or judge fun.
- Progress comes from a tight loop: run, observe, and feed concrete feedback back into the LLM.

Outcome of “Vibe Coding”

- All Rust code in this repository was written by LLMs (0 hand-written lines).
- Timeline: under 2 weeks of calendar time, roughly 30 hours of focused work.
- A lot of that time was spent waiting on the coding agent.
- Yet the end result is a complex, working geospatial toolchain with robust code that still, after 2 weeks of work, can be worked on by LLMs without fail.

Conclusion and Future Work

Successes

- Biomes, forests, terrain looks amazing.
- You can easily recognize the place you are in, even down to the specific ski slope!
- These generated worlds are a great starting point for future builds.

Limitations

- Block resolution is low: hard to capture realistic building facades and fine details.
- Lacking texture and material data for true architectural realism.
- Point cloud overhangs are challenging; need better algorithms for cliffs and balconies.
- Ultimately limited by data sources, data quality, and configuration effort.

Future Directions

- Use stairs, slabs, and clever block choices to increase apparent resolution.
- Experiment with better overhang handling for rocky faces and complex buildings.
- Refine presets and configs for more French regions and other countries.
- Personally: build a mod for gondolas, ski lifts, and skiing to complete the resort dream.

Thanks

Questions?

Star my repo: github.com/defvs/francegen