Faceted Ground and Feature Corridors for a Persistent, Live-Edited World

The `twc_terrain` Technical Reference, Thistlewright Chronicles

THISTLEWRIGHT CHRONICLES ENGINEERING, Internal technical reference, September 2026



Fig. 1. A 2 × 2 km region built by `twc_terrain`, seen from 150 m, 26 m and 3.5 m. Lowlands use a jittered 16 m lattice; mountains use broad 64 m facets; the cart road is a terrain feature cut into the ground with a sunk bed and bevelled banks. The figure on the road is 1.8 m tall. All geometry is derived at runtime from the heightfield, facet classes and feature list.

We describe `twc_terrain`, the terrain system of Thistlewright Chronicles, a single persistent world that is edited live for years and streamed to native and WebGL2 clients. The target look is hand-made low-poly art: large flat-shaded facets, irregular outlines, broad planes on mountains, and roads that are sunk into the land with bevelled banks. Every triangle is derived at runtime and deterministically on the server and all clients, because the world is edited while it runs. The persistent state consists of a 4 m heightfield, one facet-class byte per 64 m cell, and a list of terrain features, which are world objects that own a footprint. From these, a pure per-chunk function builds a jittered-lattice ground mesh with per-face palette colour, merges mountain cells into broad facets, cuts every feature into the ground along a watertight seam, and stores two bytes of per-face surface state from which a shader renders puddles and snow. In a 2 × 2 km prototype, broad facets reduce mountain triangles by 12× and local peaks by 14×. All seams are closed and manifold, a road adds about 2.5 triangles per metre, and weather changes touch only shader uniforms.

CCS Concepts: • **Computing methodologies** → **Mesh geometry models**; *Rendering*; • **Applied computing** → Computer games.

Additional Key Words and Phrases: terrain, low-poly, heightfield, procedural meshing, roads, streaming, weather

1 INTRODUCTION

Persistent online worlds accumulate years of placed objects, player history and edits. Thistlewright Chronicles is one such world, with a single canonical copy that its operator reshapes while players are present. Its terrain has to meet three demands at once. The first is a strong art direction: the ground should read as hand-made low-poly art, with large flat-shaded facets, irregular outlines, broad planes on mountains, and engineered features such as roads that are visibly sunk into the land. The second is liveness: the ground is edited at any time, so every mesh must be derivable at runtime from authoritative data, on the server and on every client, with identical results. The third is longevity and platform reach: the data format must stay stable for years, and the renderer must run in WebGL2.

Classic terrain level-of-detail methods [5, 13, 20] optimise smooth shaded surfaces for geometric accuracy. Adaptive triangulations [6, 11] produce irregular facets and concentrate triangles wherever the error is high; their shapes depend globally on the data. Vector-feature terrain systems [2] embed roads into smooth terrain through textures and displacement. Our setting asks for a faceted surface whose structure is local, deterministic and stable under edits, and whose features are part of the mesh.

`twc_terrain` addresses this with a small set of mechanisms that share one principle: the persistent state is minimal, and everything visible is a pure function of it, evaluated per 256 m chunk. We make the following contributions:

- A **faceted jittered lattice** whose vertex positions and diagonals depend only on global integer coordinates. This gives seam-free chunks, constant-time point location and a triangulation that stays fixed when heights are edited (Section 4.2).
- **Facet classes**, an authored per-cell attribute that merges mountain cells into broad planar facets and joins them to the fine lattice without T-junctions (Section 4.3).
- **Terrain features**, persistent objects that own a footprint, freeze their levels at placement and are cut into the ground with a watertight seam and a benched cross-section (Section 5).
- **Surface state**, two bytes per face that let a shader extension render rain and snow with rank-controlled coverage through shader uniforms alone (Section 6).

Section 7 describes the chunk build, streaming and editing in the Bevy engine [1], and Section 8 evaluates a prototype of the system on a 2 × 2 km region.

2 RELATED WORK

Terrain representation. Most real-time terrain systems start from a regular heightfield and differ in how they triangulate it for rendering; Pajarola and Gobbetti [16] and Galin et al. [8] survey the field. ROAM [5] refines a bintree per frame against a screen-space error. Chunked LOD [20] precomputes a quadtree of meshed tiles and hides cracks with skirts, a technique that streamed GIS formats

such as quantized-mesh [3] also use. Geometry clipmaps [13] render nested regular grids around the viewer. Right-triangulated irregular networks [6] and greedy-insertion Delaunay meshes [11, 18] adapt triangle density to the data. These methods aim for a smooth, accurate surface at minimal cost. Our lattice aims for a deliberately faceted surface whose facet size is an art-direction parameter, and we reuse chunking and skirts for level of detail. Jittering a regular lattice to break visible structure follows the same reasoning as jittered sampling in image synthesis [4].

Roads and water in terrain. Bruneton and Neyret [2] render and edit vector features such as roads and rivers on large terrains by modifying the heightfield and textures on the GPU. Galin et al. generate road paths that respect slope and cost [10] and organise them into hierarchical networks with junctions and bridges [9]. Génevaux et al. [12] derive terrain from a river network with consistent flow. Our corridors take the road geometry as given, from an operator or a generator in that tradition, and focus on embedding it into a faceted mesh: a cross-section benched into the slope, banks sized by the height they cover, and a seam that shares vertices with the surrounding ground. River levels follow the same monotonicity constraint that hydrology-based generators enforce.

Surface weather. Tatarchuk [19] describes an artist-directed rain pipeline with wet surfaces, puddles and reflections for a city scene. Fearing [7] models snow accumulation physically, and Ohlsson and Seipel [15] render accumulated snow in real time from surface orientation and exposure. We adopt the exposure-and-orientation idea and reduce it to precomputed per-face scalars, which suits a flat-shaded mesh and WebGL2.

3 SYSTEM OVERVIEW

3.1 Design goals

G1 Look. Large flat-shaded facets with irregular outlines and a restricted palette; broad planes on mountains; the underlying grid stays hidden at every distance.

G2 Runtime derivation. Every mesh is derived at runtime from authoritative data.

G3 Determinism. Server and clients derive identical geometry, since placement, collision and ground contact depend on it.

G4 Features. Roads, rivers, yards and pads have designed cross-sections cut into the ground, and they survive later edits of the surrounding land.

G5 Locality. Edits rebuild only the chunks they touch, in milliseconds, while the old meshes stay on screen.

G6 Weather. Rain and snow change the appearance of the ground itself.

G7 Boring code. WebGL2 with vertex and fragment shaders only, and a plain grid wherever a plain grid will do.

3.2 Architecture

`twc_terrain` separates *authoritative* from *derived* data (Figure 2). Authoritative data are written to the world's event log: heightfield samples with their facet classes, and terrain feature records (Listing 1). Derived data are the meshes, colliders, per-face colours and surface state that the renderer, physics and placement tools use. Derived data are recomputed wherever they are needed.

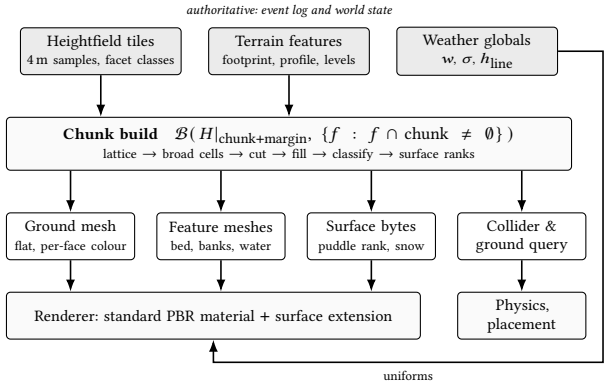


Fig. 2. Data flow. Only the heightfield tiles (with facet classes) and the feature list persist. A chunk build is a pure function of the heightfield around the chunk and the features that touch it; its outputs are recomputed on the server and on every client. Weather globals reach the shader as uniforms.

```
pub struct TileId { pub x: i32, pub z: i32 }     // 512 m tiles
pub struct HeightTile {
    pub s: Box<[u16; 129 * 129]>,                // h = 0.01 s − 200 m
    pub facet: [FacetClass; 8 * 8],              // one per 64 m cell
}
pub enum FacetClass { Fine, Broad }

pub struct TerrainFeature {
    pub id: ObjectId,              // shared with all world objects
    pub footprint: Footprint,      // Corridor | Polygon | Point
    pub profile: ProfileId,        // catalog entry, frozen
    pub edge: EdgeRule,            // Bank | Retain | None
    pub attachments: Vec<Attachment>, // Water, Kerb, Sheet
    pub priority: u8,
    pub levels: FrozenLevels,      // cm, set at placement
}
pub struct SplineNode {
    pub pos: WorldPosition, pub elevation_cm: i32, pub width_cm: u32,
}
```

Listing 1. Authoritative records (abridged).

This split keeps live editing small and uniform. A brush stroke produces a fact that lists new sample values; placing a road produces a fact that records a footprint and its frozen levels; removing the road produces a fact that deletes the record. Each peer applies the fact and rebuilds the affected chunks. The heightfield under a feature stays intact, so the natural ground reappears exactly when the feature is removed.

4 FACETED GROUND

4.1 Heightfield

The natural ground is stored in square tiles of 512 m addressed by `i32` coordinates. A tile holds 129×129 samples at 4 m spacing, with the last row and column duplicating the neighbour's first so that a tile samples independently. A sample is a `u16` s with height $h = 0.01\,s - 200$ m, covering -200 m to 455.35 m at 1 cm resolution; a raw tile is 33 KB. The heightfield is sampled bilinearly through two prefiltered versions: $\tilde{H}$ (Gaussian, $\sigma = 4$ m) for fine lattice vertices and $\hat{H}$ ($\sigma = 16$ m) for broad facets. Prefiltering at the scale of the

lattice suppresses the aliasing spikes that point sampling of 4 m detail at 16 m spacing produces. The heightfield is finer than the lattice so that features have accurate ground to sit on and a finer lattice near the camera remains possible.

4.2 Jittered lattice

The ground is a lattice with spacing $\Delta = 16$ m. Vertex (i, j), with i, j global integers, is placed at

$$\mathbf{x}_{ij} = \Delta\,(i,\ j) + J\,\big(2\,\textsc{Hash}(i, j, 1) - 1,\ 2\,\textsc{Hash}(i, j, 2) - 1\big), \quad (1)$$

with jitter amplitude $J = 0.38\,\Delta$, where $\textsc{Hash}(i, j, k) \in [0, 1)$ is a 32-bit integer avalanche hash of the packed coordinates and a salt k. Its height is $\tilde{H}(\mathbf{x}_{ij})$. Each cell chooses its diagonal with $\textsc{Hash}(i, j, 5) < \frac{1}{2}$, which removes the parallel creases that a fixed diagonal leaves (Figure 3).

Because positions and diagonals depend only on global integer coordinates, the lattice has four useful properties. Two chunks that share an edge compute identical vertices, so seams need no stitching. The triangle containing a point is found by rounding to a cell and testing that cell and its eight neighbours, since $J < \Delta/2$. The connectivity is fixed by plan positions alone, so a height edit moves facets and keeps their connectivity. Positions are integer centimetres, and the remaining floating-point work (bilinear interpolation and plane evaluation) uses basic IEEE operations, which Rust leaves uncontracted, so native and WASM builds agree bit for bit.

Non-convex cells. Jitter of 0.38Δ makes about 1% of cell quadrilaterals non-convex. In such a cell only the diagonal through the reflex vertex yields two valid triangles, and the build uses it. An exhaustive check of the extreme jitter configurations shows that a valid diagonal always exists at this amplitude, with a smallest triangle of $0.029\Delta^2$ (7.4 m^2). The test uses plan positions only, so connectivity stays independent of height.

4.3 Facet classes

A uniform 16 m lattice renders mountains as clusters of narrow, sharp facets, while the target art shows a few large planes per mountain face. `twc_terrain` therefore groups the lattice into *macro cells* of $R \times R$ fine cells ($R = 4$, 64 m) and gives each macro cell a *facet class*, Fine or Broad. The class is stored in the heightfield tile, one byte per macro cell. World generation initialises it from landform relief, the steepest slope within 60 m of the cell centre measured on a smoothed heightfield, with a threshold of 0.30. After that it is authored data that the operator can paint like any other terrain attribute.

A Broad cell is two large triangles over the macro corners (Algorithm 1). Macro corners surrounded only by broad cells receive a large jitter of $0.38\,\Delta_M$ ($\Delta_M = 64$ m). Corners that touch any fine cell keep the fine jitter of Equation 1, so the neighbouring fine cells remain valid. Broad vertices take their heights from $\hat{H}$, and the broad diagonal is the one whose midpoint lies closer to $\hat{H}$, so facet edges follow ridges and valleys.

Where a fine cell meets a broad cell, the $R - 1$ fine vertices on the shared edge are placed on the straight broad edge, in plan and in height. The broad triangle on that side then includes them in its boundary and is emitted as a fan from its centroid. All fan triangles lie in the plane of the original broad triangle, so it still renders as one face, and the seam is conforming (Figure 4). Point location remains constant-time: the macro cell is found by rounding and a 3×3 macro-cell search covers the larger jitter. Connectivity inside a broad cell depends on heights through the diagonal choice. An edit on a mountain may therefore flip a 64 m diagonal, which we accept in exchange for ridge-aligned facets.

Algorithm 1 Ground triangles of one macro cell

Input: macro cell (I, J), facet classes κ, prefiltered $\tilde{H}$, $\hat{H}$

function Vertex(i, j) ▷ fine lattice index
  if (i, j) is a macro corner with all adjacent cells Broad **then**
    $\mathbf{p} \leftarrow \Delta(i, j) + 0.38\Delta_M\ (2\textsc{Hash}(i, j, 13) - 1,\ 2\textsc{Hash}(i, j, 14) - 1)$
  else
    $\mathbf{p} \leftarrow$ Equation 1
  if (i, j) lies on a macro edge adjacent to a Broad cell **then**
    return lerp of the edge's two corner vertices at $t = k/R$
  return $(p_x,\ \hat{H}(\mathbf{p})$ **if** it touches a Broad cell **else** $\tilde{H}(\mathbf{p}),\ p_z)$
if $\kappa(I, J) =$ Fine **then**
  for each of the $R \times R$ fine cells **do**
    $d \leftarrow \textsc{Hash}(i, j, 5) < \frac{1}{2}$; use the reflex diagonal if non-convex
    emit two triangles
else
  $a, b, c, d \leftarrow$ macro corners
  $diag \leftarrow \arg\min_{(p,q)\in\{(a,d),(b,c)\}} |\hat{H}(\frac{p+q}{2}) - \frac{p_y+q_y}{2}|$
  for each of the two triangles t **do**
    $\rho \leftarrow$ corners of t with the fine vertices of every edge shared with a Fine cell
    if $|\rho| = 3$ **then** emit t **else** emit fan from centroid(t) over ρ

4.4 Face colour and lighting

Every face emits three unshared vertices carrying its normal, its palette colour and its surface bytes. This triples the vertex count relative to an indexed mesh and keeps every per-face attribute exact. The palette class comes from the face's centroid height, its slope, a low-frequency dryness mask D and its height above water (Algorithm 2). Thresholds are perturbed by a per-face hash ξ, so class boundaries follow facet edges (Figure 5). Rock is assigned from landform-scale slope, which produces coherent rock regions on steep ground, and from individual faces steeper than 0.62. Faces that belong to one broad facet, including fan triangles and pieces produced by feature cuts, inherit that facet's centroid and hash, so each facet has a single colour.

The palette is authored for Bevy's default PBR lighting, with vertex colours as albedo and roughness near one. The reference lighting in our figures is a warm, low directional sun $\mathbf{l} \propto (-0.55, 0.42, 0.55)$ with shadows, a cool hemispheric ambient term weighted by $0.55 + 0.45\,n_y$, and exponential distance fog. A low sun makes the facets legible, because each face receives a single value of $\mathbf{n} \cdot \mathbf{l}$ and neighbouring faces differ in brightness.

4.5 Ground query

Gameplay needs the height of the triangle that is drawn at a point. A built chunk keeps a uniform 4 m bucket grid over its final triangles, ground and feature alike, and GroundHeight(x, z) is a bucket

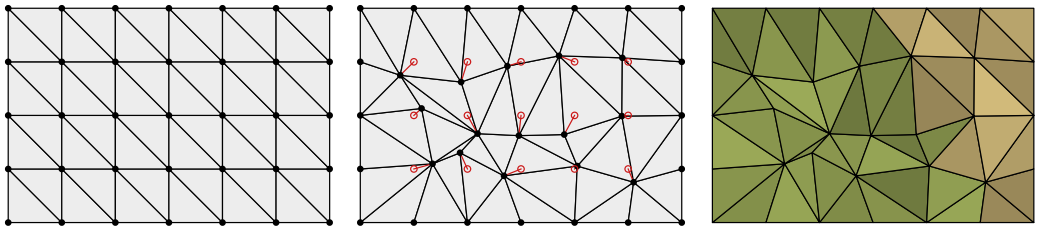


Fig. 3. Faceted jittered lattice. Left: a regular lattice with a fixed diagonal reads as a grid at any distance. Middle: each vertex is displaced by a hash of its integer lattice coordinates (red circles mark the original positions) and each cell picks its diagonal by hash. Right: the triangles are flat-shaded, each with its own palette class and tone.

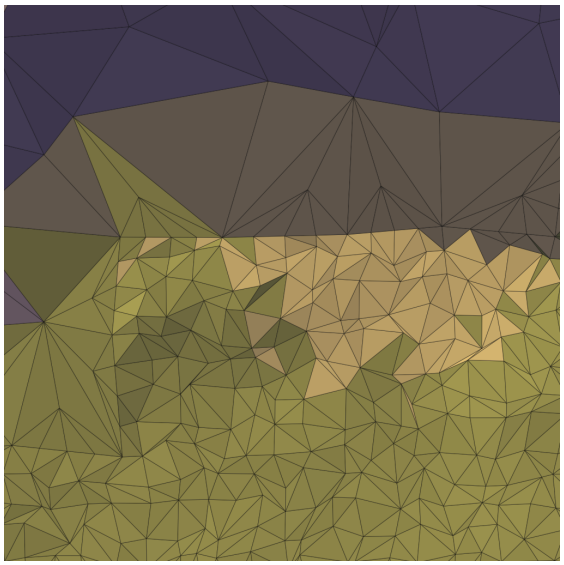


Fig. 4. A facet-class boundary in plan (320 m window). Broad cells (top) are two 64 m triangles; fine cells (bottom) are 16 m jittered cells. Along a shared edge, the fine vertices lie on the broad edge, and the broad triangle becomes a coplanar fan through them, so the seam has no T-junctions.

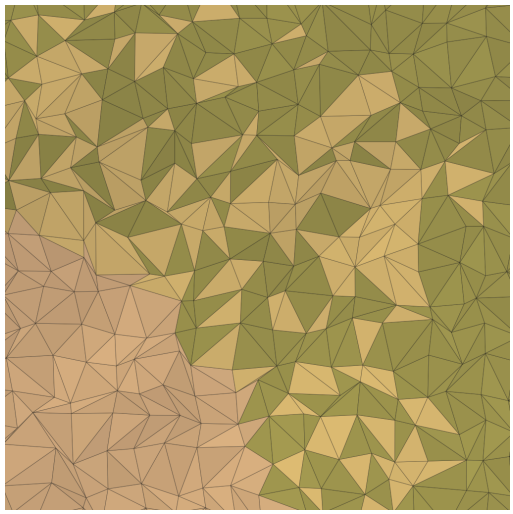


Fig. 5. A 256 m lowland patch of fine cells from above near the lake shore. Palette classes (grass, dry grass, shore) follow facet edges, and their boundaries are ragged because each threshold is perturbed per face.

Algorithm 2 CLASSIFY(t): palette colour of one face

```
c, ξ ← centroid and HASH(⌊c⌋_cm) of t, or of its broad facet
s ← 1 − n_y;  s_L ← landform slope at c
if c_y < h_water + 2.5 m then
    k ← SHORE
else if s_L > 0.42 + 0.08(ξ − 1/2) or s > 0.62 or c_y > 205 m + 30 m ξ then
    k ← ROCK
else if D(c) + 0.12(ξ − 1/2) > 0.60 then
    k ← DRY
else
    k ← GRASS
return PALETTE[k] · (0.93 + 0.14 ξ)
```

lookup, a point-in-triangle test and a plane evaluation. Before a chunk is built, the query uses the lattice directly with the cell search of Section 4.2. The same triangle list feeds the collider.

5 TERRAIN FEATURES

A 9 m road drawn on a 16 m lattice by colouring faces inherits the facets' outline (Figure 6, top). Roads, rivers, yards, building pads and similar things are therefore *terrain features*: persistent world objects, with the same identifiers, event log and placement tools as every other object, that also own a footprint. Inside the footprint a feature supplies its own profile mesh, and along the footprint's edge it meets the ground with a watertight seam.

5.1 Grammar

Every feature is described by five parts (Table 1). The *footprint* is a spline corridor, a polygon or a point. The *profile* is a catalog entry such as "cart road 9 m"; entries are frozen once published so that years of placed features keep their shape, and a new look becomes a new entry. The *edge rule* is a bank with slope ratio ρ and maximum width, a vertical retaining face, or none. *Attachments* add derived geometry such as water surfaces, kerbs, ditches and waterfall sheets. The *priority* resolves overlapping footprints.



Fig. 6. Top: a road drawn by recolouring whole 16 m ground faces inherits the lattice's sawtooth outline. Bottom: the same road as a terrain feature owns a corridor, cuts the ground along it and supplies its own bed and banks.

Table 1. Common features expressed in the grammar.

Feature	Footprint	Profile	Edge	Attachments
Cart road	corridor	9 m bed, −0.8 m	bank 1:1.6	–
Paved street	corridor	lanes + crown	retain	kerbs, ditches
Stream / river	corridor	channel	bank	water surface
Waterfall	river segment	stepped bed	retain	sheet, foam
Pond, lake	polygon	basin	bank	water surface
Yard, plaza	polygon	flat pad	bank / retain	–
Building pad	polygon	flat pad	bank	–
Fence line	corridor	low berm	bank	–

Features and ordinary objects are divided by a size rule. Anything that changes where one can walk or where water can flow is a feature. Anything smaller than about one lattice cell is an ordinary object with a sunk plinth: a well, a boulder, a fence post. A fence line is a feature as a whole, with a low berm profile.

5.2 Frozen levels

When a feature is placed, it samples the natural ground along its footprint and stores the elevations it needs in centimetres: the bed level at each spline node for a road, the water-surface level for a river, the pad height for a yard. From then on the feature's own geometry is a function of its record. Brush edits are masked inside footprints, and edits outside a footprint move only the line where the banks meet the ground. The road is an intent that the surrounding land conforms to, and changing its levels is an explicit edit of the road.

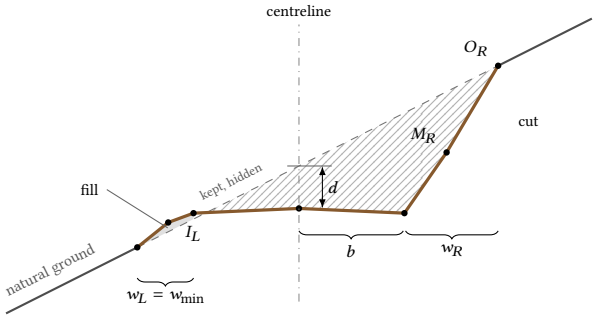


Fig. 7. Corridor cross-section on a 20% side slope (2.5× vertical exaggeration). The bed of half-width b sits a depth d below the *smoothed* centreline ground, so it is benched into the slope with a cut uphill and a small fill downhill. Each bank runs from the inner bed edge I through a jittered mid row M to the outer edge O on the ground, and its width grows with the height it covers.

5.3 Corridor cut

A corridor feature is built in three stages: solve, cut and fill. Figures 7 and 8 show the result in section and in plan.

Solve (Algorithm 3). The spline is resampled into stations every 5 m with tangent $\mathbf{T}_k$ and plan normal $\mathbf{N}_k$. At placement, the bed elevation e_k is the natural ground height along the centreline, smoothed by a Gaussian of $\sigma = 2.5$ stations and lowered by $d = 0.8$ m, and it is stored in the record. Smoothing benches the bed into side slopes the way real roads are built. The inner bed edges lie at half-width $b = 4.5$ m with a hashed wobble of ±0.5 m, which gives the road the same irregular outline as the ground.

Each bank's width follows from the height it covers,

$$w = \mathrm{clamp}(\rho\,|\Delta h|,\ w_{\min},\ w_{\max}), \tag{2}$$

with $\rho = 1.6$, $w_{\min} = 2.4$ m and $w_{\max} = 10$ m, plus a hashed wobble of ±0.6 m. Since Δh depends on where the bank lands, the width is found by a three-step fixed-point iteration. A deep cut becomes a wide bevel; where $w_{\max}$ is insufficient, the edge rule switches to a retaining face. The outer edges O^L and O^R bound the footprint polygon Π.

Cut (Algorithm 4). Each ground triangle whose bounds meet Π is classified. Triangles wholly inside are removed and triangles wholly outside are kept. A straddling triangle is clipped to $t \setminus \Pi$ with a general polygon clipper [21] in integer centimetres, so every machine obtains identical pieces. Each piece is a simple polygon, triangulated by ear clipping [14], and its vertices take their height from the plane of the source triangle. Pieces inherit the source triangle's colour. Outside Π the remaining ground is therefore the original surface, subdivided.

Seam. The boundary of the clipped pieces along Π consists of Π's own vertices and the points where Π's edges cross ground-triangle edges. Densifying O^L and O^R with exactly those crossings gives

Algorithm 3 Corridor solve

Input: spline S; stored bed levels e (none at placement)

$P_k, \mathbf{T}_k, \mathbf{N}_k \leftarrow$ resample S every 5 m
if placing **then**
  $g_k \leftarrow \textsc{GroundHeight}(P_k)$ ▷ natural ground
  $e_k \leftarrow \sum_m G_m\, g_{k+m} - d$ ▷ G: constant table
  store e in the feature record (cm)
for each station k **do**
  $\omega_k \leftarrow 0.5\,(2\textsc{Hash}(k, 22) - 1)$
  $I_k^L \leftarrow P_k + (b + \omega_k)\mathbf{N}_k$; $I_k^R \leftarrow P_k - (b - \omega_k)\mathbf{N}_k$
for side $\pm \in \{L{:}+1,\ R{:}-1\}$, station k **do**
  $O_k \leftarrow I_k \pm w_{\min}\mathbf{N}_k$
  for 3 iterations **do**
    $\Delta h \leftarrow |\textsc{GroundHeight}(O_k) - e_k|$
    $w \leftarrow \mathrm{clamp}(\rho\,\Delta h,\ w_{\min},\ w_{\max}) \pm 0.6\,\omega'_k$
    $O_k \leftarrow I_k \pm w\,\mathbf{N}_k$
$\Pi \leftarrow \textsc{Polygon}(O_0^L \ldots O_K^L,\ O_K^R \ldots O_0^R)$

Algorithm 4 Cut ground along Π

Input: chunk ground triangles $\mathcal{G}$; footprint Π; outer edges O^L, O^R

$\mathcal{E} \leftarrow \emptyset$
for $t \in \mathcal{G}$ with bounds$(t) \cap$ bounds$(\Pi) \neq \emptyset$ **do**
  if $t \cap \Pi = \emptyset$ **then continue** ▷ kept as is
  $\mathcal{E} \leftarrow \mathcal{E} \cup$ edges(t); remove t
  for polygon $q \in t \setminus \Pi$ **do** ▷ fixed-point clip
    for $u \in \textsc{EarClip}(q)$ **do**
      lift u onto plane(t); emit with the colour of t
$O^L \leftarrow \textsc{Densify}(O^L, \mathcal{E})$; $O^R \leftarrow \textsc{Densify}(O^R, \mathcal{E})$
function $\textsc{Densify}(O, \mathcal{E})$
  return O with every crossing of a segment of O with an edge in $\mathcal{E}$ inserted in order, heights from $\textsc{GroundHeight}$

the corridor an outer edge with the same vertices as the cut, at the same heights, since both lie on the same ground planes. The seam is watertight by construction.

Fill (Algorithm 5). The bed consists of three rows of vertices (left edge, centre, right edge) at height e_k, with ±0.12 m of hashed noise on the centre row so the bed breaks into shallow facets. Each bank consists of two strips: from the inner bed edge to a mid row at a hashed 45 ± 20% of the bank's width and height, and from the mid row to the densified outer edge. The rows of the outer strip have different vertex counts and are joined by a zipper that merges them by station. Banks use a darker variant of the bed colour. Bed and bank faces receive the same classification, lighting and surface state as the ground.

5.4 Networks, rivers and removal

The following parts are specified and have not yet been prototyped. Where footprints overlap, the higher-priority feature subtracts its footprint from the lower one before either is cut, so each point of ground belongs to at most one feature. Where corridors meet, a *junction node* owns the overlap region, fills it with a flat pad at a level interpolated from the incident beds, and each incident corridor ends at the pad's boundary. A bridge is an ordinary placed object; the road under it is split into two corridors that end at the abutments.

A river is a corridor network whose nodes store a water-surface level w_i, with the bed at w_i minus the channel depth. Levels are non-increasing downstream and tributaries join at the level of the river they enter (Algorithm 6), the same monotonicity that hydrology-based terrain generation enforces [12]. A drop larger than δ_{fall} between consecutive nodes makes the segment a waterfall: the bed steps, the edge rule becomes a retaining face, and the water attachment emits a vertical sheet with foam at its foot. The direction of the spline parameter is the flow direction.

Algorithm 5 Fill corridor

$C_k \leftarrow P_k + (0,\ 0.12(2\textsc{Hash}(k, 23){-}1),\ 0)$
$\textsc{Strip}(I^L, C)$; $\textsc{Strip}(C, I^R)$ ▷ bed at height e_k
for side $\in \{L, R\}$ **do**
  $f_k \leftarrow 0.45 + 0.2(2\textsc{Hash}(k, 24) - 1)$
  $M_k \leftarrow I_k + f_k\,(O_k - I_k)$ ▷ O before densify
  $\lambda_k \leftarrow 0.35 + 0.3\,\textsc{Hash}(k, 26)$
  $y(M_k) \leftarrow e_k + \lambda_k\,(\textsc{GroundHeight}(M_k) - e_k)$
  $\textsc{Strip}(I, M)$; $\textsc{Zipper}(M, O^{\text{dense}})$
function $\textsc{Zipper}(A_{0..K}, B_{0..n}$ with stations $\sigma_j)$
  $i, j \leftarrow 0$
  while $i < K$ **or** $j < n$ **do**
    if $j < n$ **and** $(i = K$ **or** $\sigma_{j+1} \le i + 1)$ **then**
      emit (A_i, B_j, B_{j+1}); $j \leftarrow j + 1$
    else
      emit (A_i, B_j, A_{i+1}); $i \leftarrow i + 1$

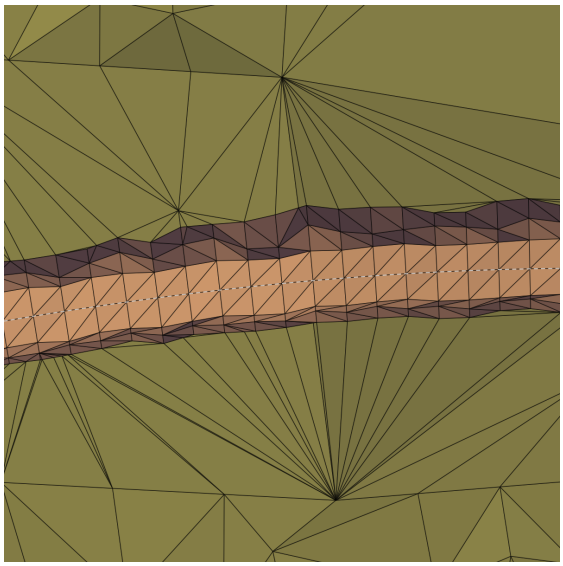


Fig. 8. Plan view of a 90 m window of the sidehill road of Figure 11. Ground triangles are clipped along the outer edge, and the corridor is filled with bed strips (light) and bank strips (dark) whose outer vertices are the cut's vertices. The uphill bank (top) is wider because it covers more height. Dashed: centreline.

Algorithm 6 River levels (on placement or edit)

for node i in topological order from the sources **do**
  $w_i \leftarrow \min(w_i,\ \min_{u \in \mathrm{up}(i)} w_u)$
  for $u \in \mathrm{up}(i)$ **do**
    mark segment (u, i) FALL **if** $w_u - w_i > \delta_{\mathrm{fall}}$
for each confluence of tributary end t into segment (a, b) **do**
  $w_t \leftarrow \mathrm{lerp}(w_a, w_b, \mathrm{param}(t))$

Deleting a feature is a fact like any other. The chunks it touched are rebuilt from the untouched heightfield, and the natural ground reappears.

6 SURFACE STATE AND WEATHER

Weather in `twc_terrain` is surface state: two bytes per face computed at build time and three global uniforms, wetness $w \in [0, 1]$, snow amount $\sigma \in [0, 1]$ and snow line h_{line}. Each frame, the shader decides per face whether it is dry, wet, a puddle or snow. Changing the weather changes only uniforms.

6.1 Per-face potentials

Puddle potential. A puddle forms on a face that is nearly level and lies below its surroundings:

$$\phi_{\mathrm{flat}} = \mathrm{clamp}\Big(\tfrac{n_y - 0.975}{0.022}, 0, 1\Big), \quad \phi_{\mathrm{basin}} = \mathrm{clamp}\Big(\tfrac{\bar{H}(\mathbf{c}) - c_y + 0.3}{1.1}, 0, 1\Big),$$
$$p_f = \phi_{\mathrm{flat}}\ \phi_{\mathrm{basin}}, \tag{3}$$

where $\bar{H}$ is the natural ground blurred over 24 m and $\mathbf{c}$ is the face centroid. A road bed sunk 0.8 m below smoothed ground scores high by construction.

Ranks. A fixed threshold on p_f floods every flat face at the same moment. The build therefore sorts the eligible faces ($p_f > 0.05$) of each chunk and material by $p_f + 0.18(\xi - \frac{1}{2})$ and stores each face's percentile rank $r_f \in (0, 1]$ in one byte, with 0 for ineligible faces. The shader turns a face into a puddle when $r_f \geq 1 - c_m\, w$. Coverage then grows linearly with wetness up to a per-material maximum c_m: 10% of ground faces and 32% of road-bed faces at $w = 1$.

Snow potential. Snow collects on high, level and shaded faces [15]. The altitude term is evaluated in the shader, which keeps the snow line global, and only the altitude-independent part is stored:

$$s'_f = 0.55\ \mathrm{clamp}\Big(\tfrac{n_y - 0.80}{0.2}, 0, 1\Big) + 0.15\, a_f + 0.22\,(\xi - \tfrac{1}{2}), \tag{4}$$

where $a_f \in [0, 1]$ measures how far the face points away from the sun in plan. It is quantised to one byte.

6.2 Shading

The surface shader extends Bevy's standard PBR material through the engine's `ExtendedMaterial` mechanism. It reads the two bytes from a spare vertex attribute and modifies albedo and adds a reflection term before standard lighting (Algorithm 7). Everything it needs is available in WebGL2. Puddles reflect a sky approximation weighted by Schlick's Fresnel term [17], remapped so that puddles remain visible from above. Wetness darkens albedo by up to 30%. Snow uses a cool albedo with a blue-tinted ambient term, so shaded snow reads cold. Trampled surfaces hold less snow through per-material offsets δ_m: 0.35 for the road bed and 0.10 for banks.

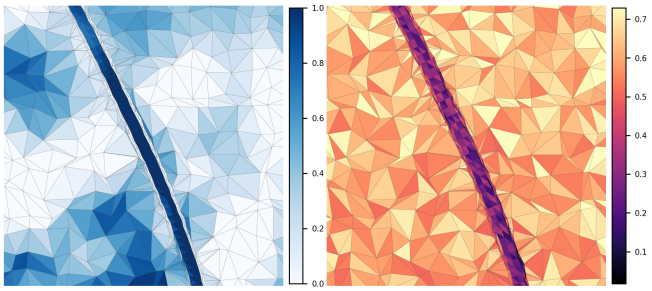


Fig. 9. Per-face potentials around the gentle road (220 m window). Left: puddle potential p_f, highest on the sunk road bed and in ground basins. Right: snow potential at $h_{\mathrm{line}} = 40$ m including the per-material offset, lowest on the road.

Algorithm 7 Surface fragment (flat per-face attributes)

Input: albedo $\mathbf{a}$, normal $\mathbf{n}$, position $\mathbf{p}$, rank r, snow s', material m
Input: uniforms $w, \sigma, h_{\mathrm{line}}$, sky
$s \leftarrow (p_y - h_{\mathrm{line}})/260 + s' - \delta_m$
snow $\leftarrow s > 1.35 - 1.25\,\sigma$ **and** $n_y > 0.62$
if *snow* **then** $\mathbf{a} \leftarrow \mathbf{a}_{\mathrm{snow}}$
$\mathbf{a} \leftarrow \mathbf{a}\,(1 - 0.3\,w)$
$L \leftarrow$ STANDARDLIGHTING$(\mathbf{a}, \mathbf{n})$
if not *snow* **and** $r > 0$ **and** $r \geq 1 - c_m\, w$ **then**
  $\mathbf{v} \leftarrow$ view direction; $\boldsymbol{\rho} \leftarrow \mathrm{reflect}(\mathbf{v}, \mathbf{n})$
  $F \leftarrow \mathrm{clamp}(0.35 + 0.6\,\mathrm{SCHLICK}(\mathbf{n} \cdot \mathbf{v}), 0, 1)$
  $L \leftarrow (1 - F)\,0.45\,L + F \cdot \mathrm{SKY}(\boldsymbol{\rho})$
return L

Table 2. Atmosphere presets used in Figure 13.

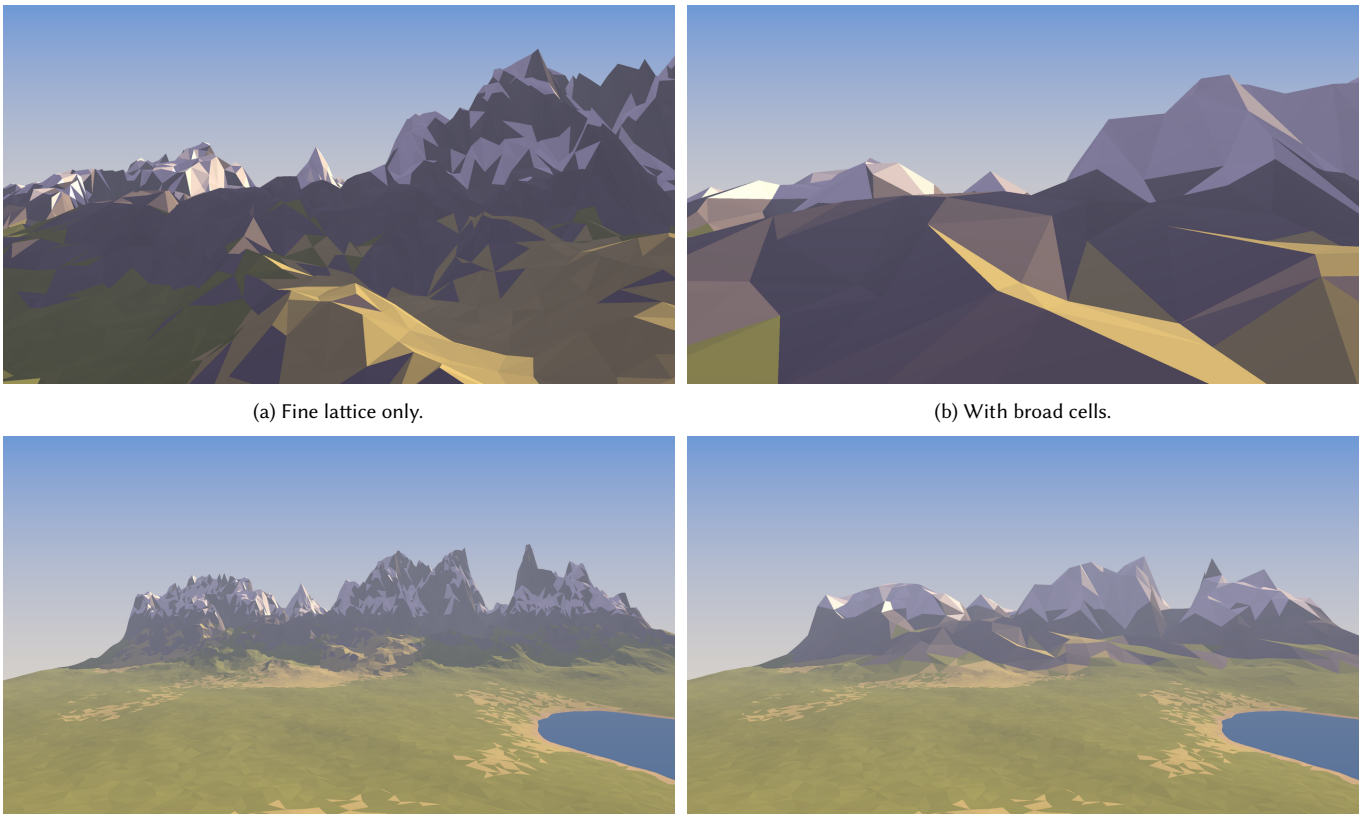
	clear	rain	snow
Sun intensity (relative)	2.1	0.45	1.8
Ambient intensity (relative)	0.70	1.35	0.80
Fog distance	6500 m	900 m	4500 m
Wetness w / snow σ	0 / 0	0.85 / 0	0 / 0.62

Atmosphere. Each weather state also sets sun and ambient colour and fog distance (Table 2). Rain adds camera-local particle streaks, and its 900 m fog matches what a person sees in rain while keeping the rain state cheap at distance.

7 IMPLEMENTATION

Crate structure. The core of `twc_terrain` depends only on `glam`, so the server, which runs `bevy_ecs` headless, links the same mesh-building code as the clients (Table 3). A feature-gated `bevy` module converts build outputs into mesh assets, owns the material extension and performs chunk swaps.

Chunk build. A chunk is 256 m square: 4×4 macro cells or 16×16 fine cells. Its build $\mathcal{B}$ reads the heightfield over the chunk plus a margin of one macro cell and the maximum bank width, and the features whose footprint bounds intersect the chunk, found through



(a) Fine lattice only.

(b) With broad cells.

(c) Fine lattice only, far view.

(d) With broad cells, far view.

Fig. 10. Mountains with and without facet classes. The fine lattice alone produces narrow spikes and a noisy silhouette (a, c). Broad cells produce large planes with ridge-aligned edges and coherent rock and snow regions (b, d). Lowland ground is identical in both.

Table 3. Modules of `twc_terrain`. Modules above the rule are plain Rust.

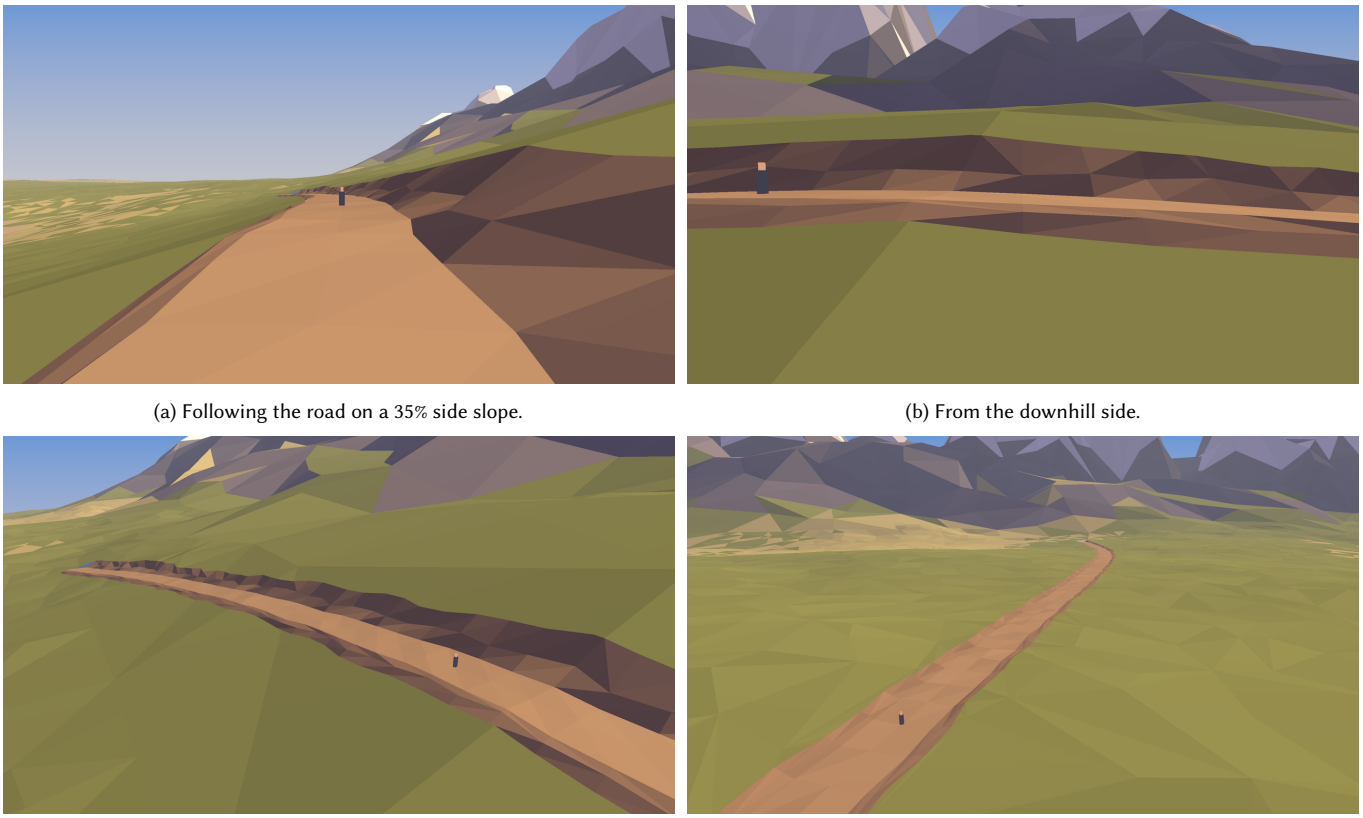
`heightfield`	Tiles, sample encoding, prefiltering, facet classes, edit facts.
`lattice`	Jitter, diagonal rules, broad cells, point location.
`feature`	Footprints, profile catalog, corridor solve, cut and fill.
`network`	Priority, junction pads, river levels, waterfalls.
`surface`	Face classification, palette, surface potentials and ranks.
`build`	The chunk job $\mathcal{B}$; pure and allocation-bounded.
`edit`	Brush application, footprint masking, dirty regions.
`bevy` (feature)	Plugin, mesh upload, material extension, group swap.

the same cell-keyed spatial index as all world objects. It runs, in order: lattice and broad cells (Algorithm 1), cut and fill for each feature in priority order, classification, surface potentials and ranks, bucket grid and collider. The corridor solve depends only on the feature record and is computed in world space, so every chunk sees the same stations, levels and outer edges. Each triangle, ground or feature, belongs to the chunk that contains its centroid, and pieces built by neighbouring chunks meet exactly because all positions are world-space and deterministic.

Streaming. Chunks stream by the same cell keys as other objects. A client holding a chunk's heightfield tile and the features touching it builds the chunk locally, so the network carries only facts.

Editing (Algorithm 8). Edits are brush strokes on the heightfield, facet-class paint, and placement or modification of features. Each produces a fact. On every peer, applying a fact marks a dirty region (the edit's footprint dilated by the maximum bank width and one macro cell) and schedules builds for all chunks it touches as one group. The old meshes remain visible until every chunk in the group has finished, and then all are swapped in the same frame, so a road crossing three chunks is always seen whole. The editing client also runs the brush on a local preview copy at 10–20 Hz and rebuilds only the chunks under the brush; the fact is sent when the stroke ends. Native builds run the jobs on Bevy's async compute pool, and WASM builds run them on the main thread between frames.

Level of detail. Distant chunks use a coarser lattice with spacing $2^k\Delta$, jittered by Equation 1 on that level's integer coordinates and organised as a chunked quadtree [20] with skirts between levels [3]. Broad cells are kept at every level up to their own size. Feature cuts are applied at the two finest levels; beyond them a feature is drawn as



(a) Following the road on a 35% side slope.

(b) From the downhill side.

(c) The same road from 30 m up.

(d) A road on gentle ground from 26 m up.

Fig. 11. Corridor cuts at person and overview scale. On the side slope (a–c), the benched bed produces a tall bevelled bank uphill and a shallow fill downhill, cut through broad facets. On gentle ground (d) the same rules produce a shallow sunken lane. The figure is 1.8 m tall.

Algorithm 8 Applying a terrain fact (every peer)

Input: fact ϕ: Samples(R, s), Facets(R, κ) **or** Feature(f)

if ϕ is Samples or Facets **then**
  write into the heightfield tiles over rect R ▷ masked at the source
  $D \leftarrow$ dilate$(R,\ w_{\max} + \Delta_M)$
else
  insert, replace or remove f in the feature index
  $D \leftarrow$ bounds$(\Pi_{\text{old}}) \cup$ bounds(Π_{new})
$\mathcal{K} \leftarrow$ chunks intersecting D
spawn $\mathcal{B}(k)$ for $k \in \mathcal{K}$ as group γ
when all of γ complete: swap meshes and colliders of γ in one frame

a strip of recoloured coarse faces, which is indistinguishable at that range (Figure 1, left). This part is specified and not yet prototyped.

Parameters. Table 4 lists the constants used in this paper. All are crate parameters except the heightfield encoding and the macro cell size, which are part of the persistent format.

Table 4. Constants.

Heightfield	spacing / tile	4 m / 512 m
	encoding	$h = 0.01s - 200$ m
	prefilter $\tilde{H}$ / $\hat{H}$	$\sigma = 4$ m / 16 m
Lattice	spacing Δ / jitter J	16 m / 0.38Δ
	macro cell Δ_M / broad jitter	64 m / $0.38\Delta_M$
	broad relief threshold	0.30
Chunk	size	256 m
Corridor	station spacing	5 m
	bed half-width b / depth d	4.5 m / 0.8 m
	bed smoothing	2.5 stations
	bank ratio ρ / $w_{\min}$ / $w_{\max}$	1.6 / 2.4 m / 10 m
	wobble inner / outer / bed noise	±0.5 / ±0.6 / ±0.12 m
Surface	basin blur $\bar{H}$	24 m
	puddle cover c_{ground} / c_{road}	0.10 / 0.32
	snow offset δ_{road} / δ_{bank}	0.35 / 0.10

8 RESULTS

We evaluated a prototype of the ground, facet classes, corridor cut and surface shading on a synthetic 2×2 km region containing lowland hills, a lake, a mountain range and two roads: a 361 m road across a 35% side slope and an 835 m road over gentle ground. The prototype is single-threaded vectorised Python with NumPy and Shapely, run on one core of a cloud VM, and renders with a software rasteriser using the reference lighting of Section 4.4. All figures in this paper come from it. Timings are upper bounds for the Rust implementation.



Fig. 12. Target concept art (left) and `twc_terrain` output (right). Top: a sunk road with bevelled banks. Bottom: a mountain face built from broad facets.

Table 5. Mountain region (1.54 km^2, 430 of 1024 macro cells classed Broad) with and without facet classes. Planes are maximal groups of edge-adjacent faces within 0.5°; peaks are vertices higher than all their neighbours.

	fine only	broad cells
Triangles in mountain region	12,030	987
Distinct planes per km^2	7,572	477
Local peaks per km^2	100.8	7.2
5th-percentile smallest angle	10.6°	12.0°
RMS deviation from H, mountains	4.4 m	20.4 m
RMS deviation from H, lowlands	0.12 m	0.20 m
Triangles, whole region	32,768	20,112

8.1 Appearance

Figure 12 compares the output with the concept art that defines the target. The road reproduces the sunk bed, the bevelled banks and the irregular outline. The mountain reproduces the large planes and ridge-aligned edges. The palette of the prototype is warmer and more saturated than the concept, which Section 9 discusses. Figure 1 shows the same world at three distances, and the lattice structure stays hidden at all three.

8.2 Facet classes

Figure 10 and Table 5 compare the mountain range with and without broad cells. Broad cells reduce the triangles in the mountain region by 12×, the number of distinct planes by 16× and the number of local peaks by 14×, and they slightly raise the smallest angles. The RMS deviation from the heightfield in the mountains grows from 4.4 m to 20.4 m. This deviation is the intended simplification of the silhouette; gameplay uses the ground query on the rendered mesh, and the heightfield remains the authoritative natural ground. In the lowlands the prefilter adds 8 cm of RMS deviation. The whole region needs 39% fewer triangles.

8.3 Seam integrity

We checked the final ground mesh for open and non-manifold edges. The mesh has no open edges away from the region boundary and no non-manifold edges, and the plan areas of all triangles sum to the region's area to within 10^{-9}, so broad cells, fans and fine cells tile the plane without gaps or overlaps. Of the 16,384 fine cells, 155 (0.95%) are non-convex and take the reflex diagonal; no triangle is inverted. After welding vertices at 1mm, the ground with both roads cut in has no open edges along either corridor. The only open edges are the two end cross-sections of each road, which a junction pad or terminal feature closes in the full system.

Table 6. Corridor measurements for the two test roads.

	side slope	gentle
Length	361 m	835 m
Fill triangles (bed and banks)	936	1,836
Fill triangles per metre	2.6	2.2
Net ground triangles added by the cut	209	423
Bank width, mean / max	3.7 / 7.4 m	2.4 / 3.3 m
Bed elevation range along the road	9.1 m	17.3 m
Open seam edges along the corridor	0	0
Build time, prototype	60 ms	120 ms

Table 7. Budgets at the finest level of detail.

Ground triangles per fine chunk	512
Vertices per fine chunk (unshared)	1,536
Vertex size (position, packed normal, colour, surface)	24 B
GPU memory per fine chunk	≈ 37 KB
Ground triangles per broad macro cell	2 to 18
Surface state per face	2 B
Heightfield and facet classes per 512 m tile	33 KB
Ground build per chunk, prototype	1.4 ms

8.4 Corridors

Figure 11 shows both roads and Table 6 summarises them. A road costs about 2.5 fill triangles per metre plus 0.5 re-triangulated ground pieces per metre, only in the chunks it crosses. On the side slope, the bank-width rule produces uphill banks up to 7.4 m wide. On gentle ground the banks stay near their 2.4 m minimum and the road reads as a sunken lane.

8.5 Weather

Figure 13 shows the three weather states at three distances, and Figure 9 shows the underlying potentials. Rank-controlled coverage keeps puddles in basins and on the road bed at every wetness level. Snow follows the broad facets on the mountains, so each plane is either covered or bare, which matches the art direction.

8.6 Cost

Table 7 lists the budgets. A fully fine chunk costs 512 triangles and about 37 KB of vertex data; broad cells lower both. The prototype builds the lattice and broad cells in 1.4 ms per chunk, and cuts a road at about 0.15 ms per metre. These counts are small for WebGL2, and the build cost fits between frames on a single thread.

9 LIMITATIONS AND FUTURE WORK

The prototype palette is warmer and more saturated than the concept's olive and tan, and it will be re-authored under Bevy's lighting. The concept also breaks road beds into smaller facets than our three-row bed; a five-row bed with more noise is a small change. Snow in

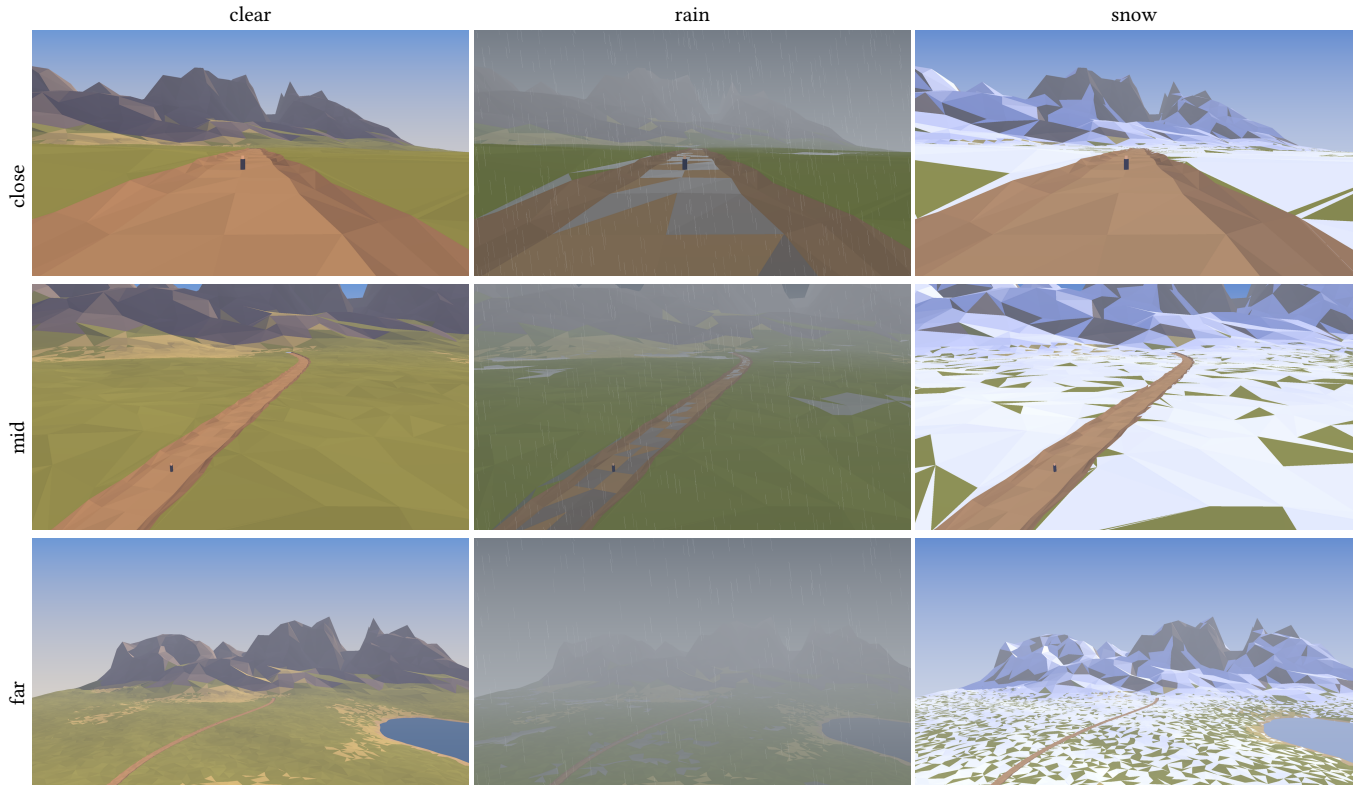


Fig. 13. The same geometry under three weather states, changed through uniforms alone. Rain darkens every face and turns the highest-ranked basin faces into sky mirrors, favouring the road bed. Snow covers level, high and shaded faces first and holds less on the trampled road. In rain the fog limits the view to about 900 m.

shadow drifts toward lavender under the blue ambient tint, and the tint needs a clamp. Per-chunk puddle ranks give each chunk exact coverage, which can show a density step across a chunk border on uneven ground; ranking over a 3×3 chunk neighbourhood would remove it. Facet classes follow the 64 m macro grid, so the boundary between fine and broad regions has a stepped outline in plan, which the relief threshold mostly places on slopes where it is hard to see. Broad cells deviate from the heightfield by design, so a feature placed on a mountain freezes its levels against the rendered surface. Junction pads, priority subtraction, rivers, waterfalls and level of detail are specified in this document and remain to be prototyped and measured.

10 CONCLUSION

`twc_terrain` derives a hand-made low-poly look from a small persistent state: a heightfield with one facet-class byte per 64 m cell, and a list of features that own their footprints. A jittered lattice keyed on global integer coordinates gives seam-free, deterministic, edit-stable facets. Facet classes give mountains broad planes at a twelfth of the triangle cost. Corridor features cut engineered cross-sections into the mesh with watertight seams, and two bytes of surface state per face carry the weather. Each mechanism is a local, pure function of authoritative data, and that property is what lets one world be edited live for years.

REFERENCES

[1] Bevy Contributors. 2026. Bevy Engine 0.19. https://bevyengine.org.
[2] Eric Bruneton and Fabrice Neyret. 2008. Real-time Rendering and Editing of Vector-based Terrains. *Computer Graphics Forum* 27, 2 (2008).
[3] Cesium. 2013. quantized-mesh-1.0 Terrain Format. https://github.com/CesiumGS/quantized-mesh.
[4] Robert L. Cook. 1986. Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics* 5, 1 (1986).
[5] Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. 1997. ROAMing Terrain: Real-time Optimally Adapting Meshes. In *Proceedings of IEEE Visualization '97.*
[6] William Evans, David Kirkpatrick, and Gregg Townsend. 2001. Right-Triangulated Irregular Networks. *Algorithmica* 30, 2 (2001), 264–286.
[7] Paul Fearing. 2000. Computer Modelling of Fallen Snow. In *Proceedings of SIGGRAPH 2000.*
[8] Eric Galin, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedřich Beneš, and James Gain. 2019. A Review of Digital Terrain Modeling. *Computer Graphics Forum* 38, 2 (2019).
[9] Eric Galin, Adrien Peytavie, Eric Guérin, and Bedřich Beneš. 2011. Authoring Hierarchical Road Networks. *Computer Graphics Forum* 30, 7 (2011).
[10] Eric Galin, Adrien Peytavie, Nicolas Maréchal, and Eric Guérin. 2010. Procedural Generation of Roads. *Computer Graphics Forum* 29, 2 (2010).
[11] Michael Garland and Paul S. Heckbert. 1995. *Fast Polygonal Approximation of Terrains and Height Fields.* Technical Report CMU-CS-95-181. Carnegie Mellon University.
[12] Jean-David Génevaux, Eric Galin, Eric Guérin, Adrien Peytavie, and Bedřich Beneš. 2013. Terrain Generation Using Procedural Models Based on Hydrology. *ACM Transactions on Graphics* 32, 4 (2013).

[13] Frank Losasso and Hugues Hoppe. 2004. Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids. *ACM Transactions on Graphics* 23, 3 (2004).
[14] Mapbox. 2016. earcut: The Fastest and Smallest JavaScript Polygon Triangulation Library. https://github.com/mapbox/earcut.
[15] Per Ohlsson and Stefan Seipel. 2004. Real-time Rendering of Accumulated Snow. In *Proceedings of SIGRAD 2004*.
[16] Renato Pajarola and Enrico Gobbetti. 2007. Survey of Semi-regular Multiresolution Models for Interactive Terrain Rendering. *The Visual Computer* 23, 8 (2007).
[17] Christophe Schlick. 1994. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum* 13, 3 (1994), 233–246.
[18] Jonathan Richard Shewchuk. 1996. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering (Lecture Notes in Computer Science, Vol. 1148)*. Springer, 203–222.
[19] Natalya Tatarchuk. 2006. Artist-Directable Real-Time Rain Rendering in City Environments. In *ACM SIGGRAPH 2006 Courses*.
[20] Thatcher Ulrich. 2002. Rendering Massive Terrains using Chunked Level of Detail Control. SIGGRAPH 2002 Course Notes, Super-size it! Scaling up to Massive Virtual Worlds.
[21] Bala R. Vatti. 1992. A Generic Solution to Polygon Clipping. *Commun. ACM* 35, 7 (1992), 56–63.