Frist Person Sketch-based Terrain Editing

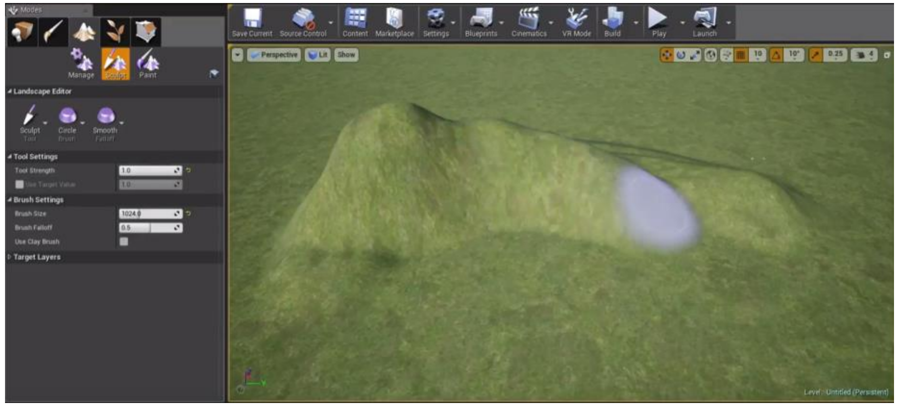
Tasse F. P.   Emilien A.   Cani M-P.   Hahmann S.   Bernhardt A.

Presentation by
Daniel Fedai Larsen

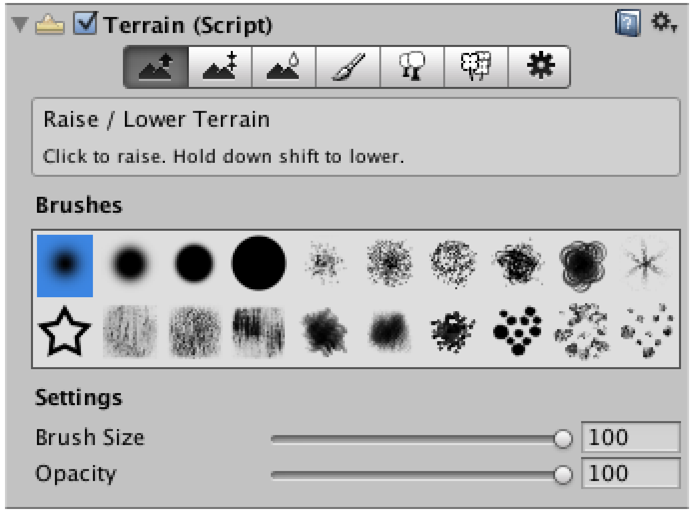
Topics

- Analyzing complex terrain (and sketches)
- Feature detection and matching
- A *minimization* problem
- Terrain deformation

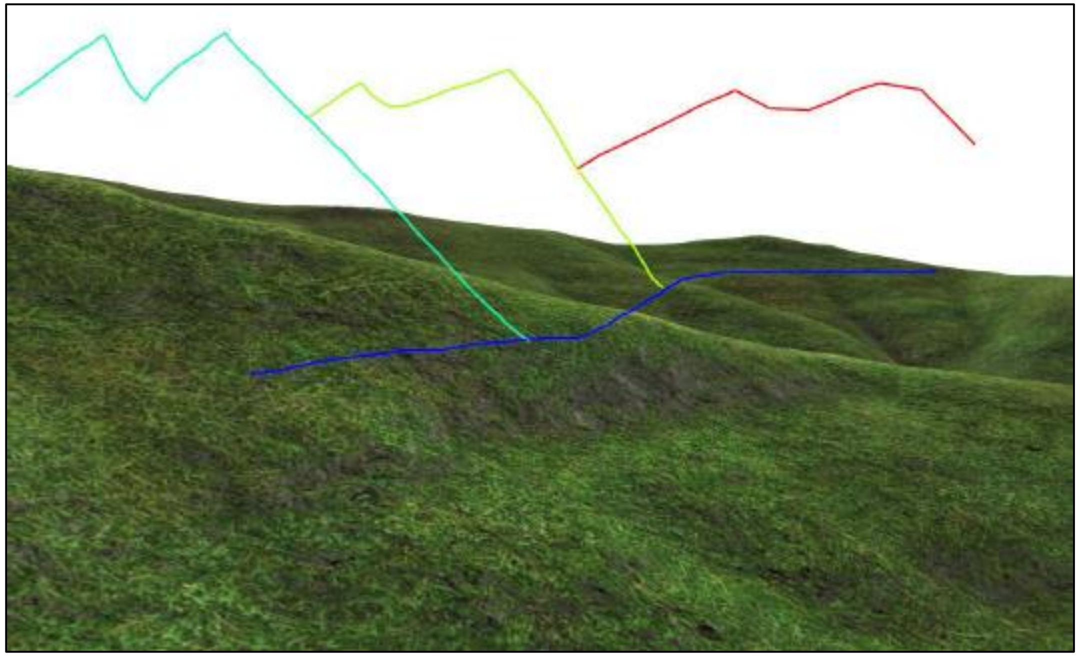
The Goal



https://www.youtube.com/watch?time_continue=2&v=HjZBhUsKa3s&ab_channel=UE4Docs

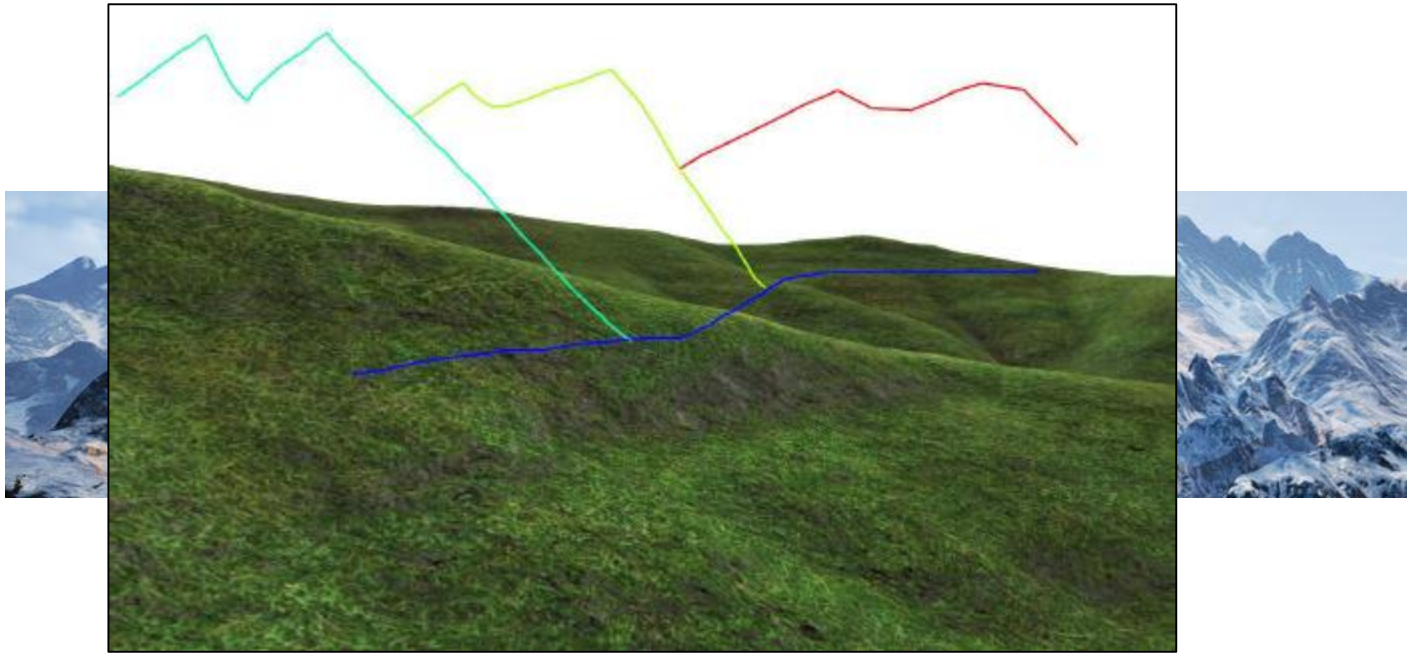


https://docs.unity3d.com/Manual/terrain-UsingTerrains.html



What are we working with?

- Terrain
  - Mesh
    - Height map
    - Texture(s)
    - 3D (X, Y, Z)
  - Ridges (static)
  - Silhouettes (non-static)
- Sketch
  - A set of 2D strokes



https://forum.unity.com/threads/terrain-textures-looking-awful-in-one-project.166909/
https://pixabay.com/en/ridge-silhouette-mountain-nature-918505/
https://en.wikipedia.org/wiki/Heightmap

Requirements

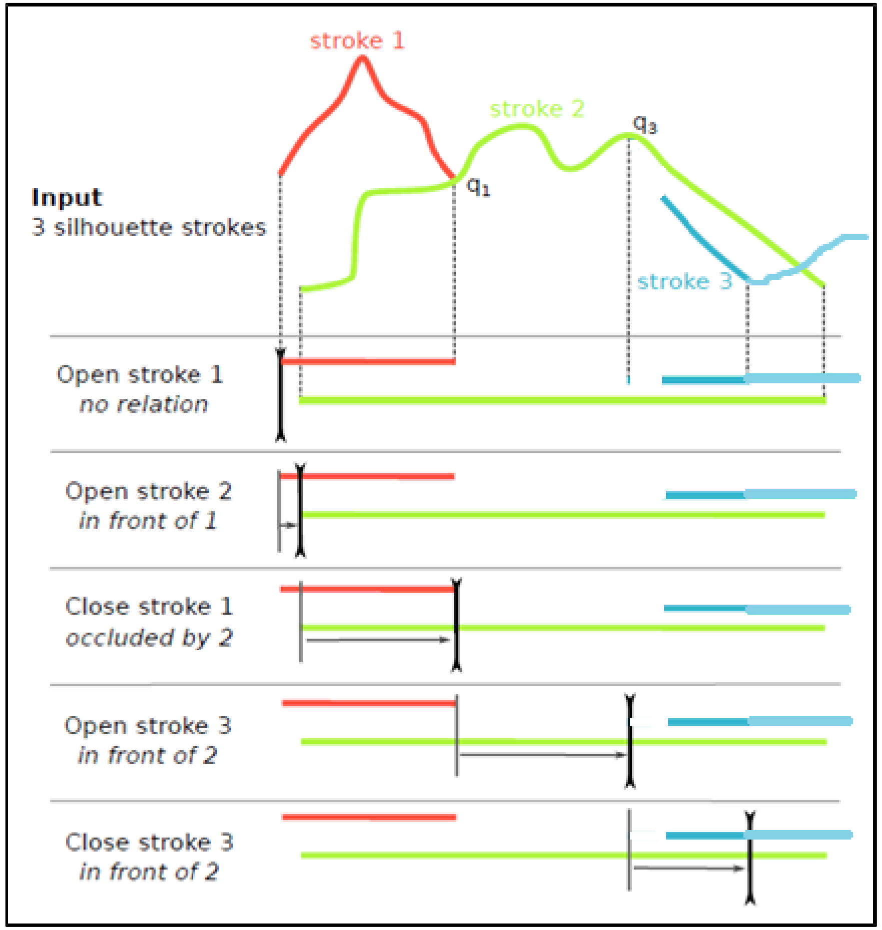
1) Every sketched stroke should be a terrain silhouette, given the current perspective view from the first-person camera viewpoint

2) Each of these terrain silhouettes should be visible, i.e. not hidden by any other part of the terrain

3) The deformed terrain should not have artifacts nor contain unrealistic deformations from any other viewpoint

Step 1 – order sketched strokes by depth

- T-junctions
  - $\|q_s - closest(r)\| < threshold$
  - q_s *occluded by* q_r *if*
    - $\angle(t_s, t_r) < 180°$
- No T-junctions
  - Stroke *s* in front of stroke *r* if
    - *r* completely contains *s*
    - $\text{min\_height}(q_s) > \text{min\_height}(q_r)$



Step 2 – feature detection

Given a polygon mesh:

- Silhouettes
  - All visible edges shared by a front face and a back face in the current perspective view -> link neighboring silhouette curves
- Ridges (PPA algorithm Chang[3])
  - Connect likely (magic) ridge points
  - Polygon-breaking on cyclic components by lowest height value until acyclic

Step 3 – stroke-feature matching

- For a stroke s_i
  - Project all terrain features $f_j : j \in 1 \ldots n$ onto the sketching plane (2D) $f_p : p \in 1 \ldots n$
  - Keep feature on condition: $range\left(f_{j_x}\right) >= range(s_{i_x})$
  - Deform each remaining feature:
    - $q.z = q.z + k\left\|q_p - q_p^s\right\| * \frac{\|q - eye\|}{\|q_p - eye\|}, \mathrm{q} \in f_j, q_p \in f_{j_p}$
    - $q_p^s = intersection(q_p, s)$

$k = -1$ if f_p is below s and $k = 1$ otherwise

Step 4 – feature cost

- $E = E_{dissimilarity} + E_{deformation} + E_{sampling} + E_{extension}$
    - $E_{dissimiliary}(f) = \frac{w1}{curvelength(f_p)} * \int_{f_p} h_{f\,p}\, dt$
    - $E_{deformation}(f) = \frac{w2}{curvelength(f_p)} * \int_{f} h_f\, dt$
    - $E_{sampling}(f) = w3 * \frac{longestedgelength(f)}{max_{g \in prioritylist(s)} longestedgelength(g)}$
    - $E_{extension}(f) = w4 * \frac{extendedcurvelength(f)}{curvelength(f)}$
- $w_i\ are\ weights (= 1.0\ in\ the\ paper)$
- $H_{f\,p} = |f_p.z\ - s.z|$
- $H_f = |f.z - f_p.z|$

Step 5 - minimization

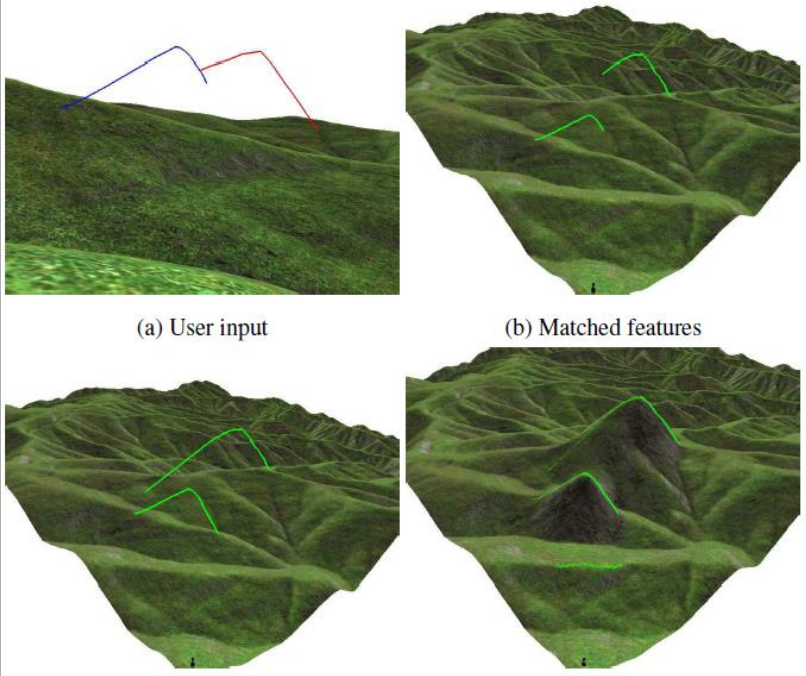
- Goal: select a feature curve for each stroke *s*

```
for s in strokes:
    feature = init_max_cost_feature()
    for f_i in s.features:
        cont = True
        for f_j in s.features(j > i):
            if f_i >= f_j: #f_j occludes f_i
                cont = False
                break
        if cont and E(f_i) < feature.energy:
            feature = f_i
```

- NP-hard problem
  - $O(sf^2), s = len(strokes), f = len(s.features)$
- Feature elimination

Step 6 – project strokes

- Strokes: screen space -> world space
  - $\forall q_s = (x_s, z_s) \in s$
    - $find\ q_p = \left(x_p, z_p\right) = (x_s, z_s) = q_s,$
      $where\ q_p\ is\ the\ projection\ of\ a\ feature\ point\ q_f$
  - Undetermined points follow tangent
- Stroke endpoints can be un-natural
  - Extend features along endpoint tangents until an intersection occurs



(a) User input (b) Matched features (c) Extend the matched features (d) Resulting terrain

Step 7 – terrain deformation

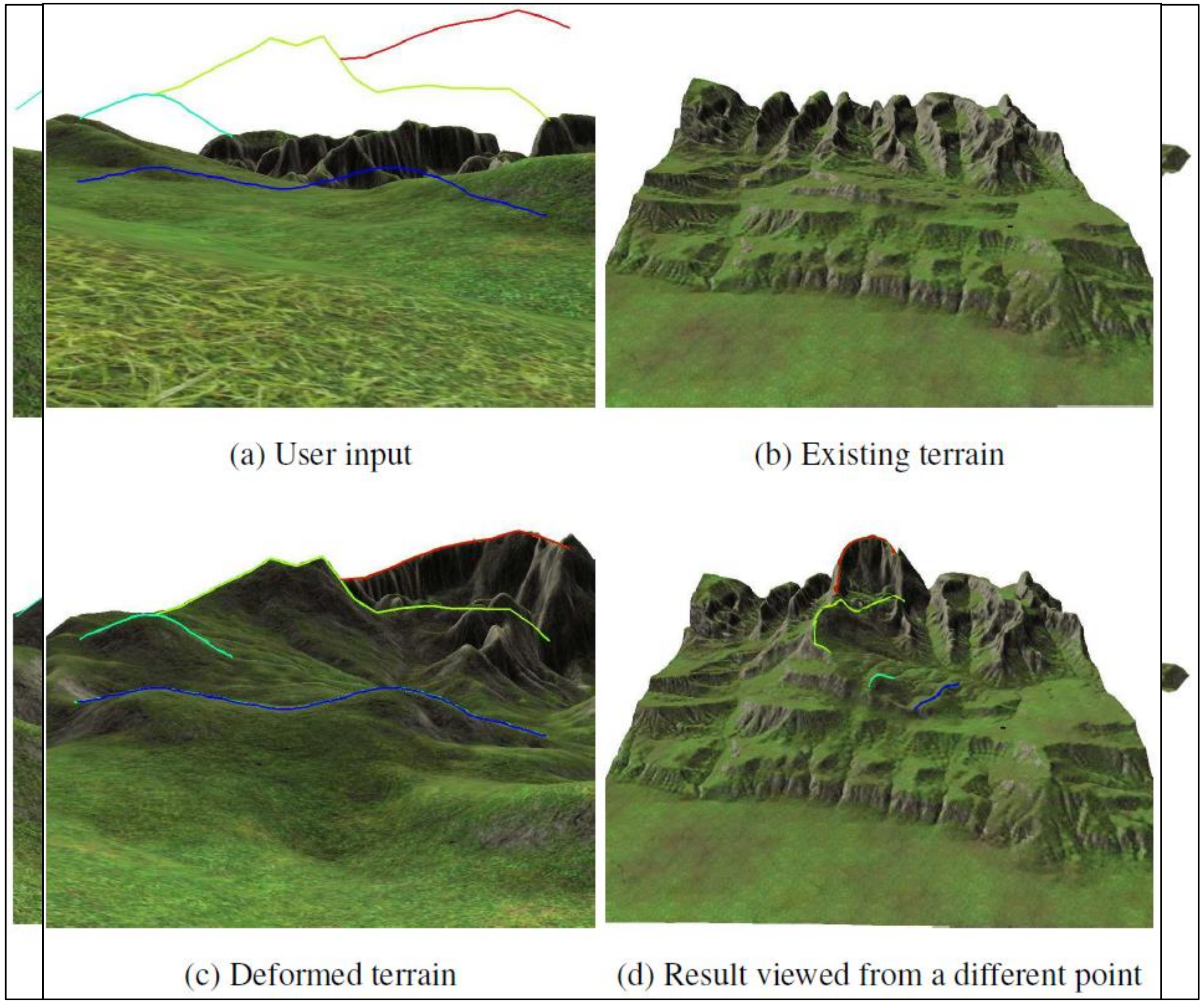
- We have worked on features, not deformed the terrain

```
for point on stroke:
    delta = point.height - height_map(p.x, p.y)
    displacement_map(p.x, p.y) = delta

new_height_map = height_map + displacement_map
```

- Solve user-defined silhouettes being hidden by lowering terrain
  - Rerun

```
for point on stroke:
    delta = point.height - height_map(p.x, p.y)
    displacement_map(p.x, p.y) = delta

new_height_map = height_map + displacement_map
```

Voilà


(a) User input
(b) Existing terrain
(c) Deformed terrain
(d) Result viewed from a different point