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
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
  • \( \|q_s - closest(r)\| < \text{threshold} \)
  • \( q_s \) occluded by \( q_r \) if
    • \( \angle(t_s, t_r) < 180^\circ \)

• No T-junctions
  • Stroke \( s \) in front of stroke \( r \) if
    • \( r \) completely contains \( s \)
    • \( \text{min}_s\text{height}(q_s) > \text{min}_s\text{height}(q_r) \)
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

https://en.wikipedia.org/wiki/Polygon_mesh
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: $\text{range} \left( f_j x \right) \geq \text{range} \left( s_i x \right)$
  • Deform each remaining feature:
    • $q \cdot z = q \cdot z + k \|q_p - q^s_p\| \ast \frac{\|q^-\text{eye}\|}{\|q_p^-\text{eye}\|}, q \in f_j, q_p \in f_j p$
    • $q^s_p = \text{intersection}(q_p, s)$
      
      $k = -1$ if $f_p$ is below $s$ and $k = 1$ otherwise
Step 4 – feature cost

- $E = E_{\text{dissimilarity}} + E_{\text{deformation}} + E_{\text{sampling}} + E_{\text{extension}}$
  - $E_{\text{dissimilarity}}(f) = \frac{w_1}{\text{curvelength}(f_p)} \times \int_{f_p} h_{f_p} \, dt$
  - $E_{\text{deformation}}(f) = \frac{w_2}{\text{curvelength}(f_p)} \times \int_f h_f \, dt$
  - $E_{\text{sampling}}(f) = w_3 \times \frac{\text{longestedgelength}(f)}{\max_{g \in \text{prioritylist}(s)} \text{longestedgelength}(g)}$
  - $E_{\text{extension}}(f) = w_4 \times \frac{\text{extendedcurvelength}(f)}{\text{curvelength}(f)}$

- $w_i$ are weights ($= 1.0$ in the paper)
- $H_{fp} = |f_p \cdot z - s \cdot z|
- H_f = |f \cdot z - f_p \cdot z|$
Step 5 - minimization

• **Goal:** select a feature curve for each stroke $s$

```python
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**
  • $O(sf^2), s = \text{len}(\text{strokes}), f = \text{len}(s.\text{features})$

• **Feature elimination**
Step 6 – project strokes

• Strokes: screen space -> world space
  • $\forall q_s = (x_s, z_s) \in s$
    • find $q_p = (x_p, z_p) = (x_s, z_s) = q_s$, 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
Step 7 – terrain deformation

- We have worked on features, not deformed the terrain

```python
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

```python
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