Università della Svizzera italiana

Faculty of Informatics

Barycentric Data Visualization for Triangle Meshes

Student: Costanza Volpini **Advisor:** Prof. Kai Hormann **Assistant:** Jan Svoboda

Abstract

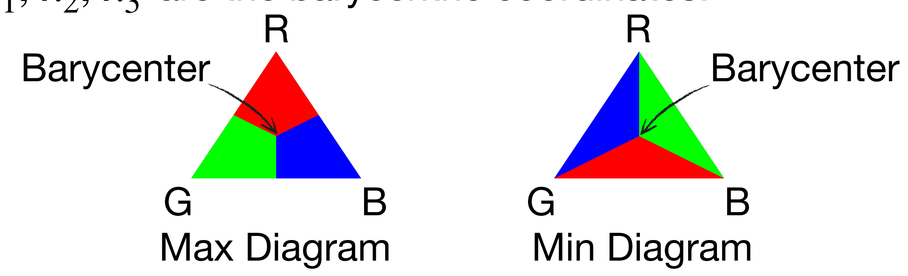
The usual way to visualize data for a triangle mesh is to associate the data with the vertices and then use linear interpolation over the mesh triangles. While this is the obvious way to go for data given at the mesh vertices, it is less natural for data given at the edges or triangles, since it requires to first aggregate the data neighboring each vertex, thus introducing an additional averaging step. In this project we want to explore alternative data visualization techniques, using the power of barycentric coordinates and GPU programming.

Barycentric Coordinates

The position of any point in a triangle $[p_1, p_2, p_3]$ can be expressed using a linear combination of barycentric coordinates:

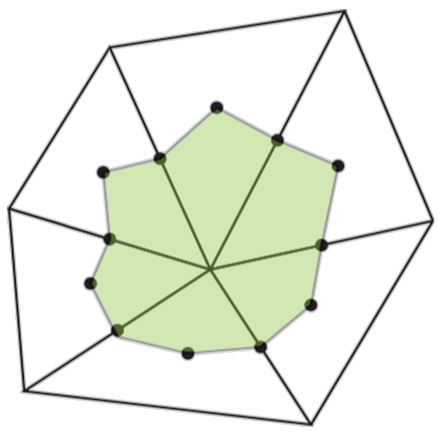
$$p = \lambda_1 p_1 + \lambda_2 p_2 + \lambda_3 p_3$$

where $\lambda_1, \lambda_2, \lambda_3$ are the barycentric coordinates.



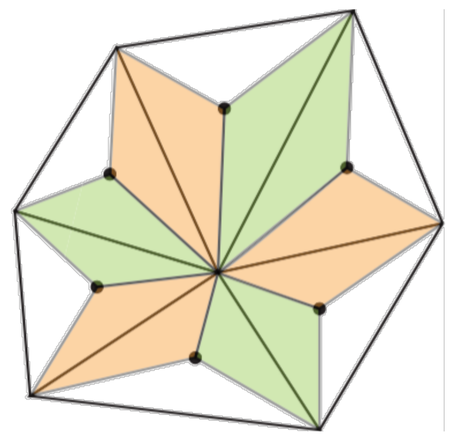
Vertex Area

For each point in a triangle, we can easily determine its closest vertex (Max Diagram), which we use as a cue for coloring all pixels in these regions. Thus visualizing data given at the vertices of the mesh in a piecewise constant, not necessarily continuous way, resembling the classical triangle flat shading.



Edge Area

To visualize data given at the edges of a mesh, we use the Min Diagram to determine the closest edge for each pixel and derive the color of the pixel from the data of that edge. This results in rhombus-shaped regions with constant color around each edge.



Shading

Triangle flat shading: compute the color at the barycentre using the triangle normal and then use it for all pixels.

Triangle Gouraud shading: compute the color at each vertex using the vertex normal, these colors are then linearly interpolated $c = \mu_1 c_1 + \mu_2 c_2 + \mu_3 c_3$.

Vertex flat shading: compute the color at each vertex and then use the max diagram algorithm to select the resulting color.

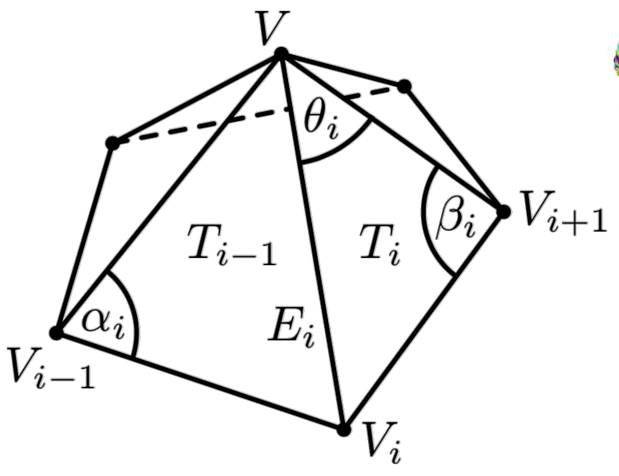
Triangle flat shading | Triangle Gouraud shading | Vertex flat shading

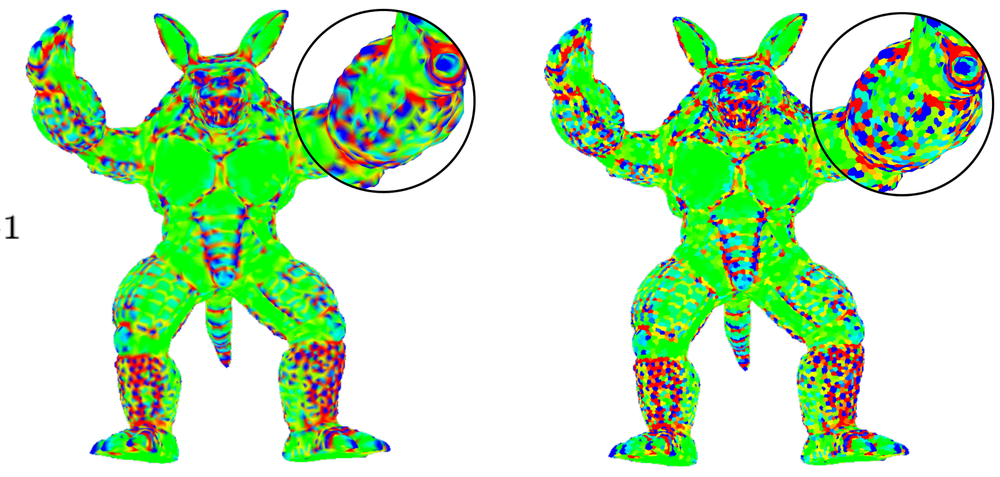
Gaussian Curvature

Angle defect: $K(V) = (2\pi - \sum_{i=1}^{n} \theta_i) / \mathcal{A}_{Mixed}$

Gouraud Gaussian curvature: compute curvature per vertex, convert to color, and linearly interpolate.

Constant Gaussian curvature: constant color around each vertex using the max diagram algorithm.





Gouraud Gaussian curvature | Constant Gaussian curvature

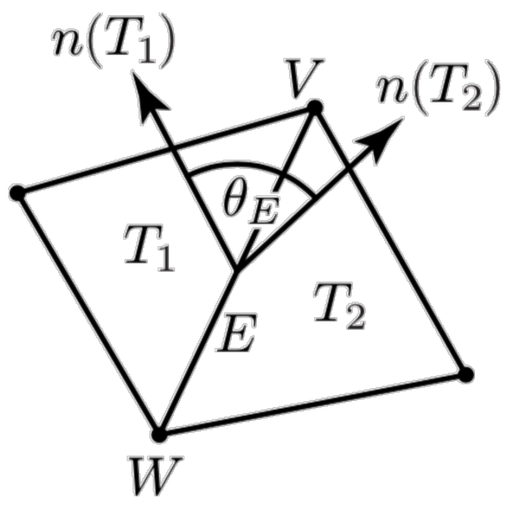
Mean Curvature

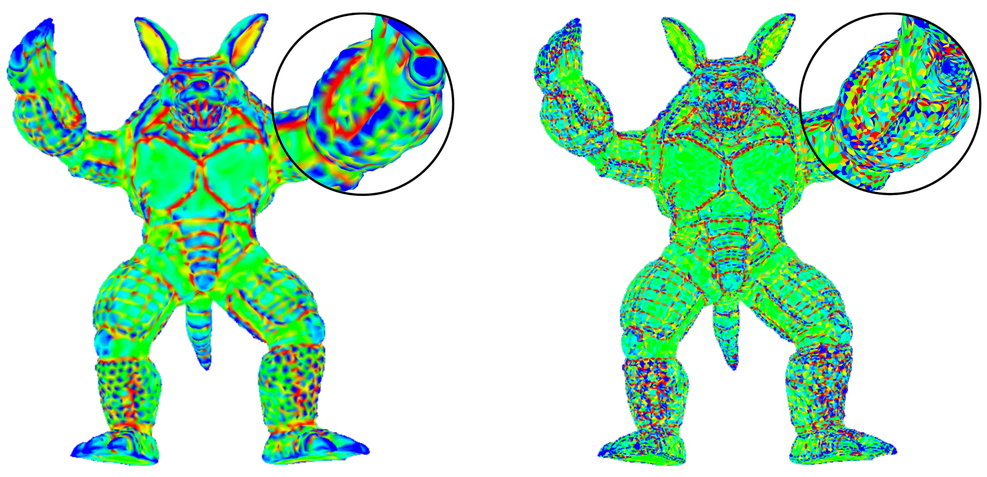
Gouraud mean curvature: linearly interpolate colors from curvature per vertex:

$$H(V) = \frac{1}{2\mathcal{A}_{Mixed}} \sum_{i=1}^{n} (\cot \alpha_i + \cot \beta_i)(V - V_i)$$

Constant mean curvature: curvature per edge: $H(E) = \frac{1}{2\mathcal{A}_{Barycentre}} \sum_{i=1}^{n} ||E_i|| (\theta_E / 2)$

and constant color around each edge using the min diagram algorithm.





Gouraud mean curvature | Constant mean curvature