Inconsistent Normals

- Many models (especially, created with modelling tools) consist of many unconnected patches
Inconsistent Normals

- Patches not necessary have consistent orientation of normals

Problems

- Real-time rendering: two-sided lighting is necessary
- Mesh processing algorithms: for example, appearance preserving simplification using vertex normals needs correctly oriented normals
Related Work

- Proximity-based and boundary-based methods
  - Hoppe et al. 1992. Surface reconstruction from unorganised points
  - Kernighan and van Wyk 1996. Extracting geometrical information from architectural drawings

- Solid-based methods
  - Thibault and Naylor 1987. Set operations on polyhedra using binary space partitioning trees
  - Teller and Hanrahan 1993. Global visibility algorithms for illumination computations
  - Murali and Funkhouser 1997. Consistent solid and boundary representations from arbitrary polygonal data
Basic Idea

- *Boundary coherence*: patches with close boundaries should be oriented consistently

or

In "solid" cases it works
Basic Idea

- **Boundary coherence**: patches with close boundaries should be oriented consistently.
- But not always...

Visibility: patches should be visible with their front-faces from most viewpoints.
Basic Idea

- **Visibility**: patches should be visible with their front-faces from most viewpoints
- Works in most cases

But in some cases it causes inconsistency
Basic Idea

- **Combined approach**
  - Maximize consistency between patches with close boundaries
  - Maximize the *front-face visibility* of all patches

Algorithm Outline

1. Detection of patches
2. Calculation of boundary coherence
3. Calculation of visibility
4. Consistent orientation of patches
Detection of Patches

1. Detect boundary and non-manifold edges
2. Divide the model into a set of manifold patches
   - Not connected with each other or
   - Connected only at vertices or non-manifold edges
3. Consistently orient polygons inside each patch

Calculation of Boundary Coherence

- Find close boundary edges of different patches
- 3D grid is used for acceleration
Calculation of Boundary Coherence

- For each pair of boundary edges calculate the local boundary coherence:

\[ c_{ij} = -\text{sgn}(s_{ij}) \cdot \sqrt{|s_{ij}|} / (1 + d_{ij}) \]

\[ s_{ij} = \vec{e}_i \cdot \vec{e}_j \]

- shortest distance between \( e_i \) and \( e_j \)

\[ d_{ij} \]

\[ \vec{e}_i \cdot \vec{e}_j \]

- Sum up all local coherences from two patches into the boundary coherence coefficient:

\[ c = \sum_{i,j} c_{ij} \]

\[ c_{12} = -\text{sgn}(\vec{e}_1 \cdot \vec{e}_2) \cdot \sqrt{\vec{e}_1 \cdot \vec{e}_2} / (1 + d_{12}) \]

\[ c_{13} = c_{12} + c_{32} + c_{45} + c_{46} \]
Calculation of Visibility

- Three methods
  - Ray shooting method
  - 5D octree method
  - GPU-based method

Calculation of Visibility

- GPU-based method
  - Distribute viewpoints uniformly around the model
  - For each viewpoint render the mesh
  - Draw each side of each patch in an unique colour
Calculation of Visibility

- **GPU-based method**
  - Read and process the frame buffer
  - For each non-black pixel increase the counter (for appropriate patch and side)
  - For each patch calculate the front- and back-face visibility coefficients:
    \[ v_f = \frac{n_f}{n_t \cdot a}, \quad v_b = \frac{n_b}{n_t \cdot a} \]

Consistent Orientation of Patches

- **Overall front-face visibility**
  \[ V_f = \frac{\sum_m v_f^m \cdot a^m}{\sum_m a^m} \]

- **Overall coherence**
  \[ C = \sum_{m,n} c^{mn} \]

- **Goal**: find orientation of all patches that maximizes both \( V_f \) and \( C \)
Consistent Orientation of Patches

- Greedy approach
  - Sort pairs of patches according to boundary coherence coefficients

- Orientation loop
  - Pop from the queue a pair of patches
  - Compare all coefficient of both patches
  - If coefficients conform
    - Connect patches into a super-patch

Result: global orientation of all patches
Results

Model with 323 patches  After applying the algorithm
Model with 78 patches  After applying the algorithm

Results

- Boundary coherence is not a sufficient criterion

Model with 71 patches  Only boundary coherence used  Boundary coherence and visibility used
Results

- Only visibility is not a sufficient criterion

Model with 137 patches

Only visibility used

Boundary coherence and visibility used

Results

- Performance rates: 3 models at 3 levels of detail

<table>
<thead>
<tr>
<th>Number of polygons</th>
<th>Number of patches</th>
<th>Patch detection time (s)</th>
<th>Coherence calculation time (s)</th>
<th>Visibility calculation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>180252</td>
<td>78</td>
<td>5.6</td>
<td>0.7</td>
<td>17.7</td>
</tr>
<tr>
<td>18019</td>
<td>80</td>
<td>1.5</td>
<td>0.6</td>
<td>6.6</td>
</tr>
<tr>
<td>1511</td>
<td>1511</td>
<td>0.4</td>
<td>0.5</td>
<td>2.5</td>
</tr>
<tr>
<td>194668</td>
<td>1508</td>
<td>5.9</td>
<td>3.4</td>
<td>6.8</td>
</tr>
<tr>
<td>64648</td>
<td>1509</td>
<td>1.8</td>
<td>2.0</td>
<td>5.8</td>
</tr>
<tr>
<td>19142</td>
<td>1511</td>
<td>0.4</td>
<td>1.1</td>
<td>2.6</td>
</tr>
<tr>
<td>300836</td>
<td>3310</td>
<td>10.5</td>
<td>9.2</td>
<td>29.1</td>
</tr>
<tr>
<td>84673</td>
<td>2040</td>
<td>2.7</td>
<td>2.8</td>
<td>9.1</td>
</tr>
<tr>
<td>14327</td>
<td>2067</td>
<td>0.4</td>
<td>0.9</td>
<td>2.5</td>
</tr>
</tbody>
</table>
Ambiguous Cases

- Hard to define the *best* orientation, even for humans

Overlapping Patches

- Could cause errors
- Special handling needed (not done yet)
Conclusion

- Consistent and sensible orientation of all normals of arbitrary polygonal models
- First method for arbitrary meshes
- Produces desirable solution for almost all practical cases
- Applicable to objects consisting of other primitives, such as NURBS

Thank you

The bot model is courtesy of Michael Beals.
Models of automotive parts are courtesy of DaimlerChrysler AG.