

Comb User Guide

version 0.02r (research)



Keenan Crane

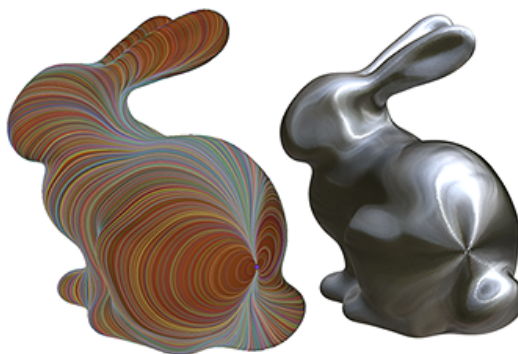
June 17, 2010

Contents

1	Introduction	3
2	Getting Started	3
3	Using Comb	3
3.1	Loading Meshes	3
3.2	Editing Interface	4
3.2.1	Navigation	4
3.2.2	Field Editing	5
3.3	Visualization	6
3.4	Exporting Data	6
4	File Formats	6
4.1	Input Formats	6
4.1.1	Wavefront OBJ	6
4.2	Output Formats	7
4.2.1	OpenEXR vector map (.exr)	7
4.2.2	Wavefront OBJ, per-face (.eobj)	7
4.2.3	Wavefront OBJ, per-vertex (.objx)	7
4.2.4	JavaView XML (.jvx)	7
5	Questions/Comments	9
6	Revision History	9
6.1	0.01r	9
6.2	0.02r	9

1 Introduction

Comb is a general-purpose direction field design application for Mac OS X. Its main function is to rapidly generate smooth direction fields on surfaces that satisfy some set of user-specified constraints – in particular, the user can interactively place a set of *singularities* that influence field behavior. Direction fields of this type can be used for a wide variety of applications, including hair modeling, anisotropic shading, texture synthesis, and remeshing. For example, the image to the right shows how a direction field produced by Comb can be used to control anisotropic shading in Luxology’s *modo*: the direction field on the left is used to produce the “brushed metal” effect on the right, where a discontinuity in shading occurs only at the specified singular point. Currently, Comb operates as a component in a graphics pipeline: it takes a mesh file as input, allows a user to interactively design a direction field, and exports this field as a texture map or one of several mesh formats. (Comb also includes rudimentary visualization tools which may be suitable for simple visualization applications.)



A major selling point of Comb is that it produces nice, smooth direction fields with *precisely* the requested singularities: no more, no less. However, there is an inherent restriction on this design process, encapsulated by the saying, “*you can’t comb the hair on a billiard ball.*” In other words, it is mathematically impossible to find a direction field on a sphere that is smooth everywhere. More generally, on any surface other than a *torus* (e.g., a donut), Comb will insist that direction fields have at least one singularity. For the same reason, Comb will also automatically update the *behavior* of singularities to respect this relationship. More information about singularity editing can be found in Section 3.2.

Comb is based on the algorithm described in Crane, Desbrun, and Schröder, “*Trivial Connections on Discrete Surfaces*” (Computer Graphics Forum/SGP 2010). Funding for this project was provided by NSF grants (CCF-0635112, CCF-0811373, CMMI-0757106, and CCF-1011944), the Center for the Mathematics of Information at Caltech, and the Institute for Advanced Study at TU München. Thanks to Felix Kälberer, Matthias Nieser, and Konrad Polthier for assistance with *QuadCover* and the JVB file format.

2 Getting Started

Currently, Comb runs only on Mac OS X version 10.6 or greater. A graphics card supporting vertex shaders is also required (pretty much any recent hardware should satisfy this requirement). On a moderately powered machine (e.g., Intel Core2 @ 2.4 GHz), the application should have no problem processing meshes on the order of 100k faces.

To install, simply drag Comb.app into your *Applications* directory. Several example meshes are provided in the Comb install directory – you may wish to try one of these first to confirm that Comb is working properly.

Developers: if you have trouble running Comb, it may be that you have an incompatible version of the Qt Frameworks installed in */Library/Frameworks*. Try moving these files temporarily to see if it resolves the issue.

3 Using Comb

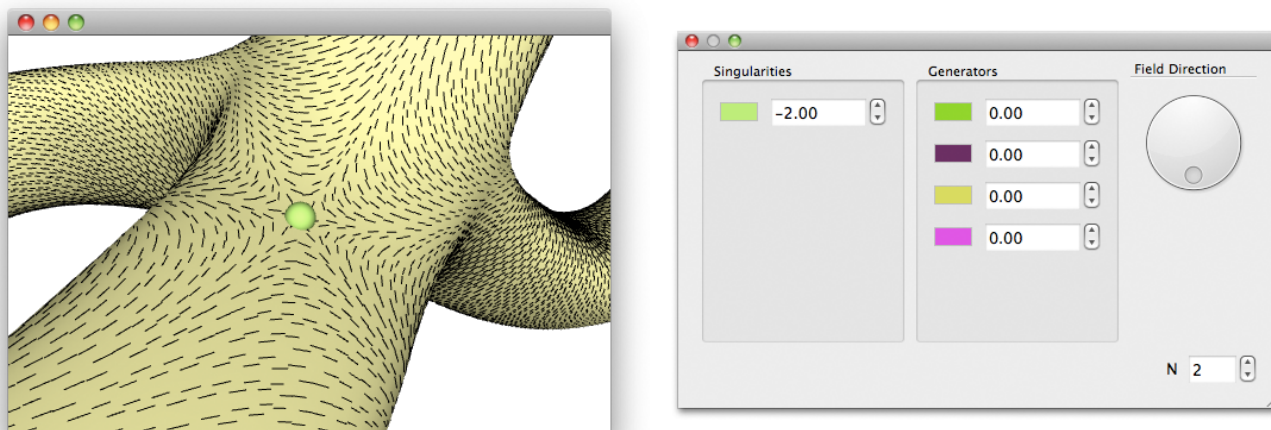
3.1 Loading Meshes

Upon starting Comb, you will be presented with a **Load Mesh** dialog. From here you can select a mesh in Wavefront OBJ format (this format is described in Section 4). The following restrictions apply to meshes supported by Comb:

- All faces must be triangles.
- The mesh must be manifold (possibly with boundary) – in other words, the set of faces touching any vertex must constitute a single triangle fan.
- The mesh must be orientable.
- The mesh must consist of a single connected component.

Once a mesh file has been specified Comb will proceed to initialize the solver and visualization routines. For large meshes there may be a slight delay before the user interface shows up – please be patient. Currently you must restart the application to load a different mesh (*File*→*Open* does nothing).

3.2 Editing Interface



The user interface (pictured above) consists of two main windows: the **Edit Window** (left), where the mesh is displayed, and the **Tool Palette** (right), which is used to specify constraint values.

3.2.1 Navigation

The viewpoint can be manipulated in the following ways:

- *Rotate camera*: click and drag in Edit Window
- *Reset rotation*: double click in Edit Window
- *Translate camera*: option-click and drag in Edit Window
- *Zoom in/out*: scroll wheel or two-finger drag (on systems with a multitouch input device)

(*Note*: If you cannot locate your mesh in the Edit Window, you may need to center and rescale the mesh data so that it fits roughly within a box between -1 and 1 around the origin.)

3.2.2 Field Editing

The primary method of controlling field behavior is via point constraints, a.k.a. *singularities*. Each singularity controls the behavior of the flow into or out of a specified point. To add a new singularity, simply shift-click near the target vertex. A colored ball will show up, indicating that the singularity has been created. Shift-clicking again on an active singularity will remove it. Note that on some meshes the final singularity cannot be removed – this situation is a result of the “hairy ball theorem” mentioned in the introduction.

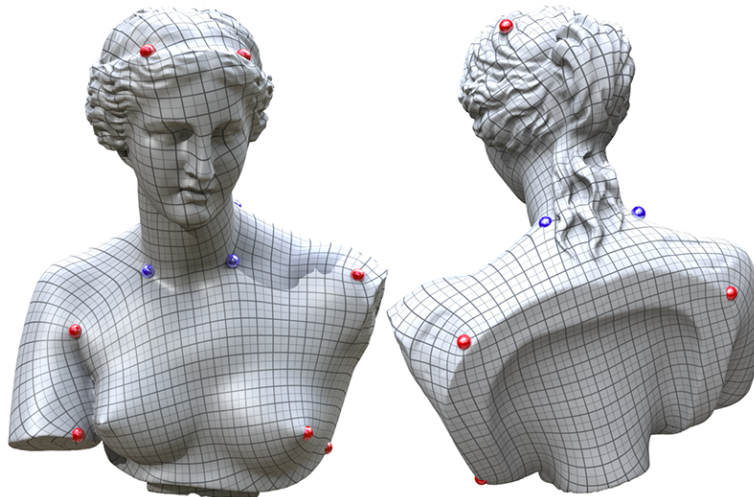
A newly created singularity may not immediately change the field behavior; alternatively, it may change the field in an undesirable way. The Tool Palette can be used to adjust the behavior of the field near a singularity. In particular, for each singularity there is a corresponding entry in the **Singularities** box; the associated number or *index* controls the appearance of the singularity. (The best way to get an intuition for the meaning of these numbers is to try it out!) Note that for certain sets of indices it is mathematically impossible to construct a corresponding direction field. For this reason, Comb will automatically modify the index of the singularity touched *least* recently to satisfy this constraint.

In addition to singularities, meshes with *holes* (e.g., the double torus or car meshes included with Comb) also have a special set of loops called *generators*. Like singularities, manipulating the indices of generators modifies the behavior of the field around these loops. Generator indices can be specified in the **Generators** box of the Tool Palette.

There are two controls that affect the overall appearance of the field: the **Field Direction** dial and the **N** or *symmetry* control. The Field Direction dial rotates each vector by a fixed angle, and does not interfere with singularities’ placement or index. The N-control allows the user to change the type of field being designed. Although any N is allowed, the following values are often most useful:

- $N=1$: direction field – a vector field where every vector has unit length.
- $N=2$: line field – a direction field with no distinction between “forward” and backward.
- $N=4$: cross field – two orthogonal lines directions are specified at each point.

Different types of fields also permit singularities with different “shapes.” Again, the best way to to understand these parameters is via experimentation. The image below demonstrates an application of *cross fields*: field directions are used to produce a quad mesh from an input triangle mesh (this particular mesh was generated by feeding the output of Comb as a JVX file into *QuadCover*). In this case, singularities correspond to vertices where the quad mesh has irregular valence – this kind of control can be difficult to achieve when building quad meshes by hand.



(Aphrodite model courtesy of Jotero GbR.)

3.3 Visualization

Several basic visualization facilities are provided by Comb and are accessible from the **View** menu:

- *Show wireframe* (command-W) - toggle wireframe overlay
- *Draw generators* (command-G) - toggle visualization of generator loops
- *Draw tree-cotree* (command-T) - toggle visualization of tree-cotree decomposition (for debugging purposes)
- *Accumulate curves* (command-R) - visualize direction field as smooth curves. These curves will accumulate until the menu is toggled again.

3.4 Exporting Data

Direction fields can be exported (along with the corresponding mesh data) to several different file formats, described in Section 4. To export a direction field, simply go to *File*→*Export...* An **Export Field** save dialog will pop up, allowing you to specify a file format from the drop-down menu – navigate to the target path and hit **Save**.

4 File Formats

The file formats currently supported by Comb are described below. These formats have been chosen for compatibility with several existing geometry processing pipelines. All files are encoded as 8-bit ASCII text files.

4.1 Input Formats

4.1.1 Wavefront OBJ

The most basic Wavefront OBJ file consists of *vertex data* and *face data*. Vertex data is specified by lines of the form

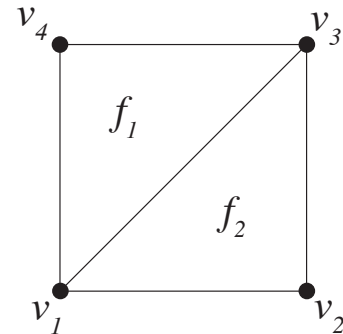
v [x] [y] [z]

where [x], [y], and [z] are x , y and z vertex coordinates specified in floating point. Face data is specified by lines of the form

f [i1] [i2] ... [in]

where [i1], [i2], etc., are 1-based indices into the vertex list. For instance, the following file specifies the mesh shown below:

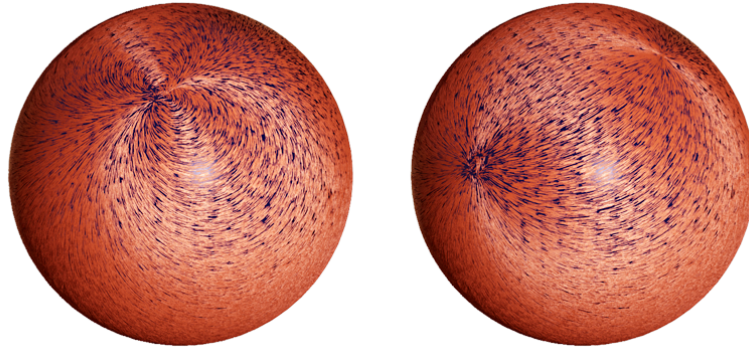
```
v 0 0 0
v 1 0 0
v 1 1 0
v 0 1 0
f 1 3 4
f 1 2 3
```



Note that Comb supports only meshes where all faces are triangles. All other attributes (e.g., vertex normals, texture coordinates, etc.) are ignored by Comb.

4.2 Output Formats

4.2.1 OpenEXR vector map (.exr)



Comb has preliminary support for exporting floating-point texture maps which can be used to control hair direction in Luxology *modo 401*. The RGB components of the exported image represent the XYZ coordinates of the unit tangent vector on the surface relative to local texture directions. (Note that the input mesh must already include texture coordinates.) This feature is under development and may not produce exactly the results you expect.

To use the exported texture in *modo*, add a "Fur Material" layer and set all of the following parameters under the "Fur Shape" tab to zero percent: *Growth Jitter*, *Position Jitter*, *Direction Jitter*, *Size Jitter*, *Flex*, *Root Bend*, *Curls*. Next, add an image map layer (specifying the EXR file exported from Comb) and set the effect to "Fur Vector." The image above shows the resulting effect of two different direction fields on a sphere.

4.2.2 Wavefront OBJ, per-face (.eobj)

This format exports direction fields as a single unit-length vector per face. It is identical to the Wavefront OBJ format, except for additional lines of the form

```
# attr f [i] [x] [y] [z]
```

Here [i] is a 1-based index into the face list, and [x], [y], and [z] are the *x*, *y* and *z* coordinates of the unit vector on the corresponding face. The initial # character denotes a "comment" line, meaning that this additional data should not interfere with standard Wavefront OBJ readers.

4.2.3 Wavefront OBJ, per-vertex (.objx)

This format also extends the Wavefront OBJ format, but exports direction fields as a single unit vector per *vertex*. In particular, for each vertex there is an additional line of the form

```
vf [x] [y] [z]
```

where [x], [y], and [z] are the *x*, *y* and *z* coordinates of the unit vector associated with the corresponding vertices – vertices and per-vertex field directions are given in the same order.

4.2.4 JavaView XML (.jvx)

This format is an XML-based format used by JavaView (<http://www.javaview.de>). It is particularly useful for quad meshing applications where direction fields may be smooth only up to local rotations by multiples of $\pi/2$. The primary fields in this format

are:

- **points** - vertex coordinates as x, y, z triples
- **faces** - triangle indices as $i1, i2, i3$ triples (0-based indexing)
- **vectors** - per-face direction vectors as x, y, z triples

In the standard JVX format, a direction field on the mesh depicted above is encoded by the following file:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE jvx-model SYSTEM "http://www.javaview.de/rsrsrc/jvx.dtd">
<jvx-model>
  <geometries>
    <geometry name="some name">
      <pointSet dim="3">
        <points num="4">
          <p>0 0 0</p>
          <p>1 0 0</p>
          <p>1 1 0</p>
          <p>0 1 0</p>
        </points>
      </pointSet>
      <faceSet>
        <faces num="2">
          <f>0 2 3</f>
          <f>0 1 2</f>
        </faces>
      </faceSet>
      <vectorField name="First field" base="element">
        <vectors num="2">
          <v>1.0 0.0 0.0</v>
          <v>1.0 0.0 0.0</v>
        </vectors>
      </vectorField>
      <vectorField name="Second field" base="element">
        <vectors num="2">
          <v>0.0 1.0 0.0</v>
          <v>0.0 1.0 0.0</v>
        </vectors>
      </vectorField>
    </geometry>
  </geometries>
</jvx-model>
```

Note that there are actually *two* direction fields specified here, pointing in orthogonal directions (this type of input is useful for quad meshing applications). For cross fields, Comb also exports *matching* data of the form

```
<matchingSet>
  <matchings num="[E]">
    <m>[i] [j] [n]</m>
    ...
  </matchings>
</matchingSet>
```

Here $[E]$ is the number of non-boundary edges in the mesh, $[i]$ and $[j]$ are the indices of two faces that share an edge, and $[n]$ is the *matching number*. More specifically, a matching number of n indicates that the frame in triangle i needs to be rotated counterclockwise by an angle of $n\pi/2$ to get it to agree with the frame in triangle j . The matchings exported by Comb are exact for all edges, even near singularities.

5 Questions/Comments

Comb was developed as part of an academic research project and could still use a lot of work! We would love to hear about any suggestions you have for further development (for instance, real-world applications that require direction field design, useful file formats), as well as any problems you encounter while using the software. Please contact Keenan Crane at keenan@cs.caltech.edu.

6 Revision History

6.1 0.01r

June 17, 2010. First release.

6.2 0.02r

June 25, 2010. Indices of boundary loops are now properly managed by the GUI.