EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2007)
M. van de Panne, E. Saund (Editors)

Free-form Sketch

Haixiong Wang and Lee Markosian

University of Michigan

Abstract

We describe a sketch-based system for modeling 3D shapes based on a new multiresolution shape representation we call a layered mesh. *Like subdivision surfaces, layered meshes provide a multiresolution hierarchy of meshes. A key difference is that a layered mesh lets you edit the shape and structure of the mesh at any level of the hierarchy, through the notion of shape primitives organized in a dependency network. The simplest primitives are points and curves, which can be used to define several kinds of parameteric surface. Surfaces can be inflated or smoothly joined to produce a broad range of shapes. An important feature of the system is the ability to refine shapes and add detail by* oversketching *primitives, either directly or via the curves that define them. While our user interface is still in development, our initial results show the potential for this approach.*

1. Introduction

The problem of how to model 3D shapes has remained one of the more interesting and challenging in computer graphics. Although existing methods are extremely powerful – nearly anything can be modeled quite convincingly – most applications are geared toward expert users; the tools tend to be hard to learn and slow and painstaking to use. One reason is that these methods were originally developed for computer-aided design, where precise mathematical control is an important requirement. Many other applications, like stylized animation, 3D games, and interactive 3D illustrations have different requirements. For them, it is more important that the user can quickly and intuitively specify *approximate* shapes.

Existing commercial and research systems have addressed this need using various strategies, but all impose significant limitations on the types of shapes that can be modeled (e.g., SKETCH [ZHH96], Teddy [IMT99] and Smoothteddy [IH03], SMARTPAPER [SC04], FiberMesh [NISA07]), and SketchUp [Ske07].

We describe a new prototype system, Free-Form Sketch, that lets the user create organic, free-form shapes through a sketching interface. The system is based on an incremental approach: simpler shapes help define more complex ones. The simplest shapes are points and curves – both are easy to specify (even in 3D) with a sketching interface. Points and curves in turn help define several kinds of surfaces, including those that interpolate a set of boundary curves, generalized cylinders, and surfaces of revolution. Surfaces can be “inflated” and smoothed to yield a rich set of shapes, which can be smoothly joined with other shapes at arbitrary scales to produce more complex surfaces still.

Underlying the system is a new shape representation that we call a *layered mesh*. Like subdivision surfaces [Zor06], layered meshes define a multi-resolution mesh hierarchy. That is, each mesh generates a “child” mesh, which by default is the result of applying one round of approximating subdivision to the mesh. We use Loop subdivision in areas of triangles, Catmull-Clark in regions of quads, and a hybrid method in mixed regions, similar to Stam and Loop [SL03]. Like Hybrid Meshes [GKSS02], layered meshes support topological changes between levels of the hierarchy through the addition of new structures.

Layered meshes are distinguished by two key features: (1) regions within the mesh (at any level of the hierarchy) may be procedurally controlled, e.g. to fit a shape such as a parameterized or implicit surface; and (2) each mesh in the hierarchy may have non-manifold surface topology (specifically, some edges may have more than two adjacent faces), but the mesh can always be decomposed into 2-manifold surface regions (where every edge has one or two adjacent faces) that are smoothly joined through an extra layer of surface called “skin.” (See Figure 1.)

Each separate procedurally controlled mesh region is

Copyright © 2007 by the Association for Computing Machinery, Inc.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions Dept, ACM Inc., fax +1 (212) 869-0481 or e-mail permissions@acm.org.
Sketch-Based Interfaces and Modeling 2007, Riverside, CA, August 02-03, 2007.
© 2007 ACM 978-1-59593-913-5/07/0008 $5.00

delivered by EUROGRAPHICS DIGITAL LIBRARY
www.eg.org diglib.eg.org

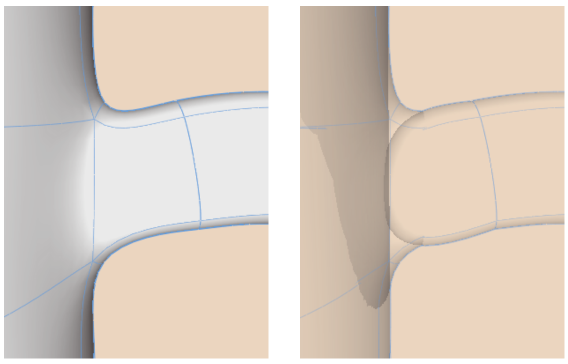

Figure 1: *Left: two surfaces blended together (normal view). Right: interior view showing original surfaces, plus the layer of "skin" that joins them.*

called a "primitive." The simplest primitives are points and curves (corresponding to individual mesh vertices and chains of mesh edges, respectively). Various types of surface primitives are described in Section 3.3. Primitives form a dependency network, where a newly added primitive can depend on (and derive its shape from) existing ones.

Layered meshes thus let the user smoothly attach finer structures to existing coarse mesh structures. The finer structures are "carried along" when the coarser shape is changed by the user. We describe a prototype system based on this shape representation that supports sketch-based modeling of a range of free-form shapes. Some aspects of the system are still in development (e.g., the user interface, and the set of available operations), but even our preliminary results show the potential for layered meshes to model a rich set of organic, free-form shapes that would be difficult to create with previous methods.

2. Related Work

Free-Form Sketch is largely inspired by SKETCH [ZHH96] and Teddy [IMT99], which also pair a gestural user interface with a set of basic primitive shapes that can be combined to yield more complex shapes. Both systems impose significant restrictions on the kinds of shapes that can be created – mainly boxy shapes in SKETCH and simple rounded shapes with little geometric detail in Teddy – and both support only limited ability to oversketch and refine the shape once it has been created.

Like Teddy and earlier work by van Overveld [vW96], Free-Form Sketch uses *inflation* to build smooth shapes from curves or simpler surfaces ("polygons" in van Overveld's approach). Bloomenthal and Wyvill [BW90] take a similar approach, defining implicit surfaces from skeletal components and offset distances. In our system, the user can directly oversketch the shape (via silhouettes and feature curves) or the skeleton points, curves, or surfaces that define it.

A new modeling system, FiberMesh [NISA07], extends Teddy to let the user oversketch feature curves that define the shape. The system can handle both sharp and smooth shape features, and provides better control over the final shape. The approach is based on functional optimization and is essentially different from ours, which uses a multi-scale and hierarchical shape representation.

ShapeShop [SWSJ05] is a sketch-based modeling system based on BlobTrees (implicit surfaces combined via CSG operators) as the underlying surface representation. Due to the representation, arbitrary topology models can be robustly created and edited. Topology edits are harder to make in Free-Form Sketch, as the user edits the mesh structure explicitly. In return, the resulting meshes are highly regular and have useful parameterizations.

Another approach to sketch-based modeling is to interpret finished sketches all at once, rather than incrementally, as they are drawn. Shesh and Chen's SMARTPAPER [SC04] is one example of such a system. This approach generally places greater restrictions on the kinds of shapes that can be sketched, in part because of the inherent ambiguity of sketches, especially when it comes to free-form shapes.

Subdivision surfaces [Zor06] can compactly represent piecewise smooth shapes of arbitrary topology. While a great deal has been written about them, most research has investigated subdivision rules and properties of the the resulting limit surface, assuming the control mesh is given. Our system lets the user interactively build control meshes with regular connectivity and good triangle or quad aspect ratios – both of which help improve the quality of the resulting subdivision surfaces.

3. System Description

Free-Form Sketch is still under development, particularly the user interface. Our goal is to implement a pen-based UI that is self-disclosing and easy to learn, but also fast and effective for practiced users. Our basic approach is to classify the user's input strokes as ordinary strokes or special commands, e.g. to select an object or activate a "widget." We use basic gesture interpretation algorithms similar to those used in many gesture-based systems (e.g., MathPad^2 [LZ04]).

Widgets are 2D or 3D user-interface elements that help mediate various editing operations. A widget represents a temporary editing mode in which the user sees an on-screen representation of the points, curves, or surface regions that can be edited. (See for example the red and blue curves defining the cross-section and profile of a surface of revolution in Figure 4.)

As explained in Section 1, shapes in Free-Form Sketch can be decomposed into *primitives*. Each primitive represents a basic part of the shape that can typically be edited in some way. Widgets are thus provided for each type of

© Association for Computing Machinery, Inc., 2007.

primitive and associated editing operation. In the following sections we describe the various types of primitives currently supported in our system.

3.1. Points

A point is the simplest primitive. It can be a lone point in 3D, but more often it represents an endpoint of a curve. As moving a point in screen space does not uniquely define a movement in 3D, we set certain constraints on the points. Every point is associated with an arbitrary plane that it resides on. Its movement is constrained to be either parallel to the plane or along the plane normal, depending on the initial direction of the motion.

When a point is moved, all entities that depend on the point are updated as necessary. For example, when the endpoint of a curve is moved, the curve is scaled and rotated accordingly.

A point can be embedded in a curve or a surface, which means the movement of the point is constrained so it stays on the curve or surface. Also, it is possible that a point is the centroid of a triangular face, so it cannot be directly manipulated by the user. However, if the face changes, the position of the point will be updated.

3.2. Curves

Creating and editing curves in 3D is one of the most important tasks for a modeling system. In our system, a curve is represented as a parameterized polyline that is approximated by a chain of mesh edges and vertices (down to a prescribed level of subdivision). We support two ways to create curves: drawing on a user-specified surface (usually a plane), or creating a non-planar 3D curve that corresponds to a prescribed (planar) shadow curve, using the method of Cohen *et al.* [CMZ*99].

In either case, once the 3D shape of the curve is given as a polyline, mesh vertices are created to sample the curve with a prescribed spacing, and mesh edges are generated to connect vertices into a chain. Each curve may generate a "child" curve in the subdivision hierarchy, up to a maximum resolution, after which ordinary smooth subdivision rules take over. (We use subdivision rules that reproduce uniform cubic B-splines [HDD*94]). Other primitives use a similar policy. This way, we can fit a given reference shape (e.g.a polyline drawn the user) without fitting the "noise."

Control points of the curve except the two endpoints are invisible from the user and thus cannot be manipulated directly. Instead we support curve editing by "oversketching." When a curve is selected and a stroke is drawn near the curve, the system determines the region of the curve affected by the stroke and rebuilds that region using the rules described by Baudel [Bau94].

As with points, once a curve is oversketched, primitives that depend on it must be updated. For example, if the user oversketches a boundary curve of a surface, vertex positions within the surface may need to be recomputed. We discuss different types of surfaces in the following sections; each has its own way of computing vertex positions.

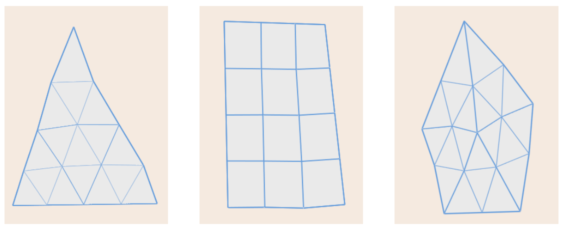
Figure 2: *Tessellation of triangular, quadrilateral and pentagonal panels by adding interior vertices.*

3.3. Surface Primitives

In Free-Form Sketch, a surface primitive controls some region of the mesh (including both mesh connectivity and vertex positions). There are various types of surface primitives, with corresponding ways to edit them. Nealen *et al.* [NSACO05] implement surface oversketching by solving a linear system of equations to achieve coarse edits while preserving fine geometric detail in meshes loaded from file. In our system, we can use information collected during construction of the control mesh to support oversketching.

3.3.1. Panels

A *panel* is a surface defined by its boundary curves (see Figure 2). A panel can be used to model a desired shape directly, or it can serve as a skeleton to generate an inflated shape (see Figure 8). When the user draws a closed sequence of boundary curves, the system instantiates a panel to fill the interior region. As a fall-back, we can triangulate the surface using a Delaunay triangulation, using just the mesh vertices around the boundary. However, such triangulations typically have bad aspect ratios, and so perform poorly for subdivision.

In practice, therefore, we check for several cases that have a simple structure allowing us to do better. For example, when there are 4 boundary curves, and every other one has the same number of edges, we fill the interior with a grid of quads. When the number of boundary curves is 3, and they have the same number of edges, we can triangulate by cutting the surface in 3 directions (see Figure 2). The system handles several other cases like these. The system has some limited support for changing the existing mesh connectivity. E.g., a curve can be re-tessellated to contain more edges. The goal of this and specialized panel tessellation rules is to create triangles and quads with good aspect ratios, and facilitate subdivision.

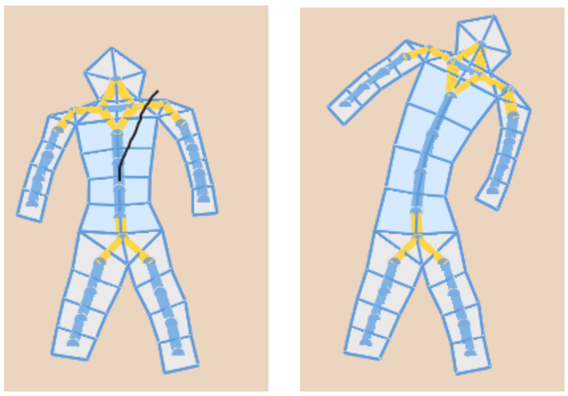

Figure 3: *Left: a paper-doll consisting of 10 panels; thick blue lines/dots denote the spines of panels, and the yellow lines connects them together; the user is oversketching a spine. Right: the result of the oversketch.*

The tessellation process provides us with useful information. For a Delaunay triangulated panel, we can find its spine by the "chordal axis" technique used in Teddy [IMT99]. For a panel that is triangulated using a specific rule, the spine is defined by connecting the center of "meaningful" cells in the panel, which are sets of faces determined by the rule used. (See Figure 3 for examples.) Note that panels can be connected together. In our system, the spine is a curve that the user can oversketch. Afterward, a simple recursive procedure ensures that all the neighboring panels are updated accordingly.

Finally, since a panel is defined by its boundary, its shape should change as the boundary curves are oversketched. To compute new locations for interior vertices of the panel, we perform an iterative relaxation process on each interior vertex. At each iteration, each vertex is moved toward the centroid of its one-ring neighbors. The resulting surfaces act like "soap films," approximating minimal surfaces that interpolate the boundary curves.

3.3.2. Parameterized surfaces

Free-Form Sketch also supports parameterized surfaces, which assign vertex positions based on a mapping from a 2D domain to 3D space. One example is a surface of revolution (see Figure 4). In this case, the profile curve and two cross-section curves together determine the mapping. Since those are the curves that define the surface, the user can oversketch them to adjust the shape of the surface of revolution. When that happens, vertex positions are recomputed based on their assigned 2D coordinates.

3.3.3. Tubes

A generalized cylinder in our system is called a *tube*. It consists of a roughly planar base shape extruded along a skeleton curve. For each edge in the skeleton curve, we create a cross section similar to the base shape, centered at the midpoint of the edge and perpendicular to it. Each vertex on the cross section has a fixed local coordinate with respect to the coordinate system determined by the edge (using the edge midpoint as origin and edge direction as one axis). Neighboring cross sections are then connected together. An example of a tube is shown in Figure 5.

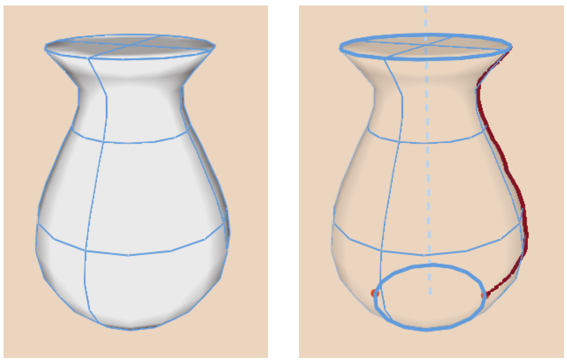

Figure 4: *Left: a subdivided surface of revolution. Thin blue lines show the structure of the control mesh. Right: a "widget" lets the user overksetch the thick blue cross-section curves or the red profile curve to modify the shape.*

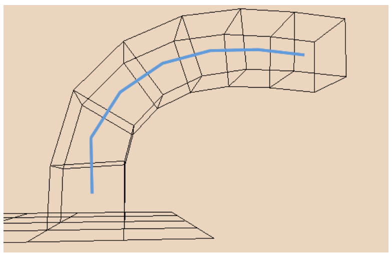

Figure 5: *The structure of a tube.*

Tubes are similar to the extrusion primitive used in Teddy [IMT99]. Because of the underlying dependency network, our system offers more control over primitives than Teddy does. The surface of the tube depends on its skeleton, so once created, the tube can be further modified by oversketching the skeleton curve. Since vertices on the tube are controlled by local coordinates with respect to edges of the skeleton curve, their locations are naturally updated after the shape of the curve changes (see Figure 6).

Our system also supports sketch-based editing of the silhouette or cross section of a tube. Similar to the silhouette oversketching method of Nealen *et al.* [NSACO05], we select a silhouette or cross section, and compute 3D offsets for vertices within the area of influence based on the oversketch. For a structure as simple as a tube, we do not need to solve a linear system of equations for this computation. Instead,

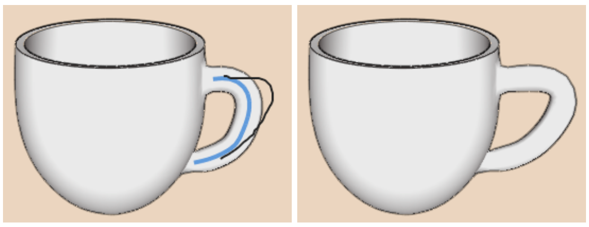

Figure 6: *Left: a cup in a toon shading style; user is oversketching the skeleton of the handle, which is a tube. Right: the result of the oversketch.*

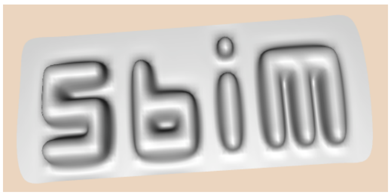

Figure 7: *An example of inflation with uniform offset.*

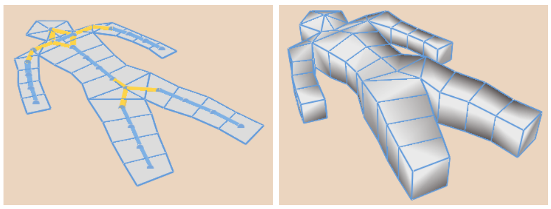

Figure 8: *Inflation of a paper doll.*

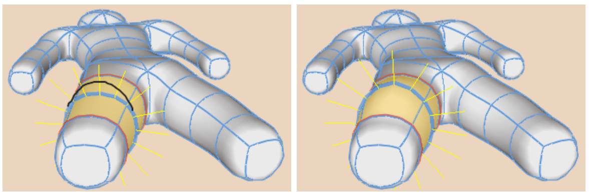

Figure 9: *Left: thick blue curve is the selected cross section; normals of vertices are shown to guide the oversketch; orange denotes the area of influence. Right: result of the oversketch.*

we follow the simple rule that the further away that a vertex is from the silhouette or cross section, the smaller offset it receives. Figure 11 demonstrates this editing operation: we transformed the rounded cap of a tube into a sharp end by editing the cross section at one end of the tube.

In most cases, we can treat a regular tube as the inflation of its skeleton curve. It is also intuitive to think of a sphere as the inflation of a point, which is trivial to implement. Next, we describe the inflation of a surface.

3.3.4. Inflated surfaces

Free-Form Sketch supports two kinds of inflation for surfaces. The first uses a uniform offset distance. It takes an already existing surface (or subset of the surface), creates a copy of the surface with vertices offset along their normal direction, and stitches the two surfaces together by creating quads that connect corresponding boundary edges. Figure 7 shows an example of such extruded surfaces. The offset distance can be adjusted by the user.

The second kind of inflation is non-uniform two-sided extrusion. It takes an existing "skeleton" surface composed only of panels, from which the system computes corresponding skeleton curves (see Figure 3). The inflation process is similar to uniform inflation, except that the offset distance depends on the distance to the nearest skeleton curve (similar to the inflation policy used in Teddy [IMT99]). Figure 8 shows an example of such inflation. After the inflated surface is created, the original skeleton surface disappears, but can be re-displayed to be oversketched. Any edits to its shape are propagated to the inflated shape.

The user can switch between viewing the inflated surface and the skeleton surface for editing purposes. Since the skeleton surface is composed of panels, its spines and boundary curves can be oversketched as described in Section 3.3.1. As with the tube, vertices of the inflated surface are controlled by local coordinates with respect to points on the skeleton surface, so whenever the skeleton changes, its inflated surface is updated accordingly. A surface that is inflated from a quad panel can be treated as a tube (e.g., the torso, legs and arms in Figure 8), thus the user can also apply sketch-based editing to its silhouettes or cross sections (Figure 9).

3.4. Combined Shapes

An important feature of our layered mesh representation is that it lets the user smoothly combine primitives to create more complex composite shapes. As an example, Figure 10 shows a cup model, constructed according to the following steps. First, the user creates a surface of revolution, then "knocks out" the top panel. Next, the user inflates the resulting surface to give the cup thickness. Finally, a handle (represented by a tube) is attached to the side of the cup. As shown in the figure, instead of staying put, the handle adjusts appropriately when the body of the cup changes shape.

In the final step mentioned above, our current implementation requires that the two areas to be joined have the same mesh structure. For example, both are quad faces in Figure 5. The smooth join is accomplished using an additional "skin" surface that covers the 1-ring neighborhood of the joined areas (Figure 1). While our skin primitive is based on the algorithm of Markosian *et al.* [MCCH99], our implementation uses a static triangulation. A more dynamic approach would

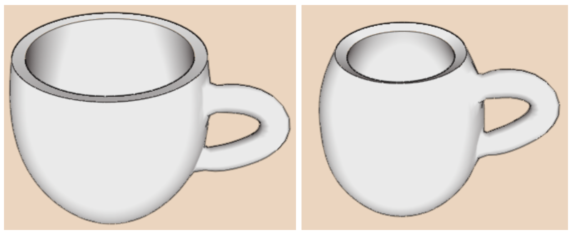

Figure 10: *The handle of the cup changes along after the profile curve of the body (surface of revolution) is oversketched.*

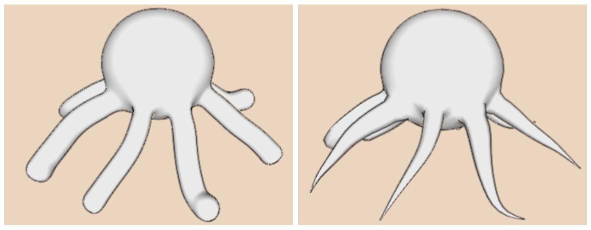

Figure 11: *Transforming the round tentacles of an octopus to have sharp ends; this octopus model is composed of a sphere and 8 tubes.*

be needed to handle the case that the two areas to be joined do not have the same mesh structure.

Figure 11 shows another example of combined shapes. The octopus consists of a sphere inflated from a point, and 8 tubes attached to the sphere.

4. Discussion and future work

We have described a system for sketching free-form shapes based on two key strategies: (1) we define complex shapes via inflation from simpler (lower-dimensional) shapes that are easy to sketch; and (2) we use layered meshes to smoothly join primitives together to create more complex composite shapes. Both strategies require thinking through the structure: the user must mentally decompose the intended object into basic forms that can be modeled with primitives provided in our system. While this may be difficult for novice users, it can also be a powerful and intuitive approach to modeling (Figures 6 and 11).

Looking ahead, we are interested in streamlining the user interface, supporting additional operations (e.g., better controls for editing mesh connectivity), and providing more support for procedural modeling. For example, after the first octopus tentacle is created, it may be preferable to add the remaining tentacles via "copy and paste," instead of creating each manually.

References

[Bau94] BAUDEL T.: A mark-based interaction paradigm for free-hand drawing. In *Proc. SIGGRAPH 94* (1994), pp. 109–116.

[BW90] BLOOMENTHAL J., WYVILL B.: Interactive techniques for implicit modeling. *1990 Symposium on Interactive 3D Graphics 24*, 2 (1990), 109–116.

[CMZ*99] COHEN J. M., MARKOSIAN L., ZELEZNIK R. C., HUGHES J. F., BARZEL R.: An interface for sketching 3D curves. *1999 Symp. on Interactive 3D Graphics* (1999), 17–22.

[GKSS02] GUSKOV I., KHODAKOVSKY A., SCHRÖDER P., SWELDENS W.: Hybrid meshes: multiresolution using regular and irregular refinement. In *SCG '02: Proc. of the 18th Annual Symposium on Computational Geometry* (2002), pp. 264–272.

[HDD*94] HOPPE H., DEROSE T., DUCHAMP T., HALSTEAD M., JIN H., MCDONALD J., SCHWEITZER J., STUETZLE W.: Piecewise smooth surface reconstruction. In *Proceedings of SIGGRAPH 94* (1994), pp. 295–302.

[IH03] IGARASHI T., HUGHES J. F.: Smooth meshes for sketch-based freeform modeling. In *2003 ACM Symposium on Interactive 3D Graphics* (2003), pp. 139–142.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3D freeform design. In *Proceedings of SIGGRAPH 99* (1999), pp. 409–416.

[LZ04] LAVIOLA J. J., ZELEZNIK R. C.: Mathpad^2: a system for the creation and exploration of mathematical sketches. *ACM Transactions on Graphics 23*, 3 (2004), 432–440.

[MCCH99] MARKOSIAN L., COHEN J. M., CRULLI T., HUGHES J. F.: Skin: A constructive approach to modeling free-form shapes. In *Proc. SIGGRAPH 99* (1999), pp. 393–400.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: Designing freeform surfaces with 3D curves. *ACM Transactions on Graphics 26*, 3 (2007).

[NSACO05] NEALEN A., SORKINE O., ALEXA M., COHEN-OR D.: A sketch-based interface for detail-preserving mesh editing. *ACM Trans. on Graphics 24*, 3 (2005), 1142–1147.

[SC04] SHESH A., CHEN B.: SMARTPAPER: An interactive and user friendly sketching system. *Comput. Graph. Forum 23*, 3 (2004), 301–310.

[Ske07] SketchUp. `http://sketchup.com/`, 2007.

[SL03] STAM J., LOOP C.: Quad/triangle subdivision. *Computer Graphics Forum 22*, 1 (2003), 79–85.

[SWSJ05] SCHMIDT R., WYVILL B., SOUSA M. C., JORGE J. A.: Shapeshop: Sketch-based solid modeling with blobtrees. In *SBIM 2005* (2005).

[vW96] VAN OVERVELD C. W. A. M., WYVILL B.: Polygon inflation for animated models: a method for the extrusion of arbitrary polygon meshes. *The Journal of Visualization and Computer Animation 8*, 1 (1996), 3–16.

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: SKETCH: An interface for sketching 3D scenes. In *Proceedings of SIGGRAPH 96* (1996), pp. 163–170.

[Zor06] ZORIN D.: Modeling with multiresolution subdivision surfaces. *SIGGRAPH 2006 Course Notes* (2006).

© Association for Computing Machinery, Inc., 2007.