# A Painting Interface for Interactive Surface Deformations

Jason Lawrence
Princeton University
Graphics and Geometry Group
jlawrenc@cs.princeton.edu

Thomas Funkhouser
Princeton University
Graphics and Geometry Group
funk@cs.princeton.edu

## Abstract

*A long-standing challenge in geometric modeling is providing a natural, intuitive interface for making local deformations to 3D surfaces. Previous approaches have provided either interactive manipulation or physical simulation to control surface deformations. In this paper, we investigate combining these two approaches with a painting interface that gives the user direct, local control over a physical simulation. The "paint" a user applies to the model defines its instantaneous surface velocity. By interactively simulating this velocity, the user can effect surface deformations. We have found that this painting metaphor gives the user direct, local control over surface deformations for several applications: creating new models, removing noise from existing models, and adding geometric texture to an existing surface at multiple scales.*

## 1 Introduction

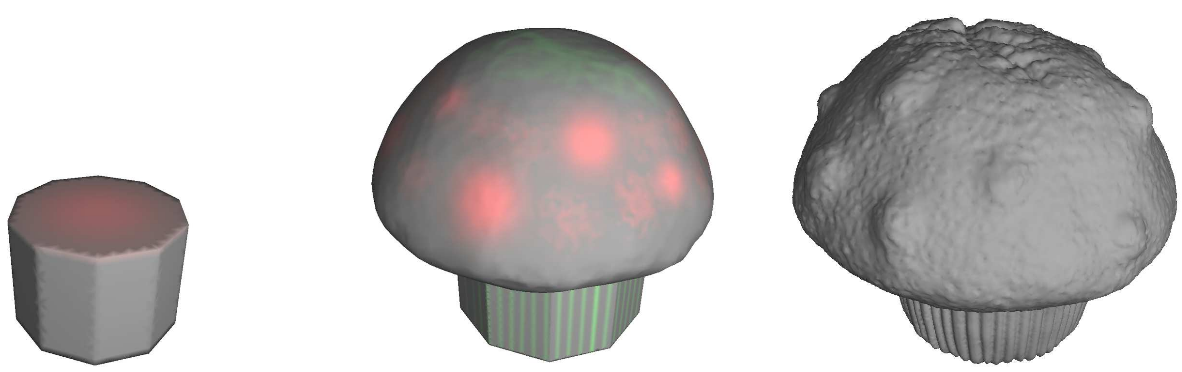
Creating intuitive geometric modeling interfaces is a fundamental problem in computer graphics. The need to manipulate complex geometric models arises in a variety of areas from making movies and video games to designing cars and buildings. The ultimate goal of any interface is to empower 3D modelers with direct control of a model's shape, freeing them from understanding the underlying machinery. However, it is inherently difficult to manipulate 3D objects with typical 2D devices such as a mouse and computer screen. This, in part, follows from the difficulty of specifying locations and directions of motion in $\Re^3$ from cursor activity restricted to the 2D image plane. Another difficulty is allowing a user to specify complex, large-scale changes to a model's surface that might require the creation/positioning of thousands of vertices. Moreover, many interfaces lack a physically intuitive connection between editing the shape of the object and the human modeler's activity.

In this paper, we describe a novel modeling interface that gives the user direct, local control over a model's surface. The key idea is a metaphor that allows the user to "paint" directly onto the model as a way of expressing surface deformations. These deformations are achieved by assigning an instantaneous velocity to the model's surface as a function of its paint and then interactively simulating the implied surface motion over a user-controlled time interval.

Figure 1(c) shows a model created using our painting interface. The surface of this blueberry muffin is quite complex and it's not clear how one would use existing modeling tools to create both its overall shape and the detailed cracking along its top. However, the muffin is the result of a physical process: as it cooked, the surface of hot dough expanded outward from the top of a cylindrical solid. In this context, the muffin's overall shape could be captured by simulating this cooking process in terms of surface velocity. This is the approach we take in our modeling interface. We begin with a simple base model and paint its surface in order to produce the desired change to its shape (Figure 1(a)). After interactively simulating this motion and obtaining the muffin's overall shape, we add more paint that will create ridges along the base of the muffin, form bulges for the blueberries, give the surface the texture of cooked bread, and produce an irregular crack along the muffin's top (Figure 1(b)). We alternate between painting and interactively simulating until we arrive at the model shown in Figure 1(c).

The main advantage of our approach is in exposing the inherently intuitive nature of physical simulation as a modeling tool through a direct painting interface. We accomplish this by establishing a metaphor that allows the user to "paint" the surface's instantaneous velocity which is then simulated to gain the desired change to the model's shape. By applying this type of paint using traditional 2D brushes to a model's 3D surface, the user can quickly and easily express complex surface deformations.

Our contributions are establishing this painting interface and developing a prototype system based on dynamic polygonal meshes and the level set model. The rest of this paper focuses on the definition of paint and the associated user interface before presenting our results.



(a) Painted Base Model (b) Painted Intermediate Model (c) Result

*Figure 1. Many natural objects, like this blueberry muffin, are easy to model with interactive physical simulation. (a) In our system the user paints a base model in order to control the instantaneous velocity of its surface where red/green paint correspond to positive/negative speed (with respect to the surface normal). (b) Next, the user interactively simulates this surface velocity until the overall shape of the muffin is achieved. Then, the user adds more paint to this intermediate model to further deform its surface. (c) After several iterations of painting and simulating, we produce the final model.*

## 2 Related Work

We envision our painting interface as being one tool within a comprehensive modeling package. Clearly, there are advantages to modeling with other techniques that our approach cannot match. However, we find our system useful for quickly and accurately adding detail at various scales, locally smoothing noisy models, and for creating certain stylized scenes.

Some of today's most powerful modeling interfaces directly expose a mesh of vertices to the user. This mesh acts as either a representation of the surface itself or as control points for a NURBS surface [31] or subdivision surface [43]. Through direct manipulation of these vertices, a user can create smooth, analytical shapes. Adaptive editing techniques have also been investigated [12, 44]. Alternatively, moving control points can allow the user to deform the surface by deforming the space around it [32, 9, 24]. For all their advantages, however, this class of modeling interfaces has the drawback of burdening the user with the task of positioning control points, adjusting weights, and inserting knot vectors. Moving individual control points can sometimes result in unexpected changes to the surface resulting in the user having to position many control points to effect detailed changes to a model. Moreover, direct manipulation of free-form deformations [18] requires solving a minimization problem, making this technique too expensive for interactive editing of large models. Our system gives the user direct control over a physical simulation thus offering a simpler interface for expressing surface deformations. By providing adaptive refinement of the underlying mesh, our system elegantly handles deformations at any level of detail while still permitting large-scale surface changes.

Virtual sculpting interfaces are available in many modeling systems [26, 40, 30, 15, 5, 13, 42, 14]. Although sculpting offers an intuitive modeling tool for anyone accustomed to forming real objects from clay or wax, they require the precise positioning of 3D virtual tools to deform the model's surface. Consequently, making detailed changes to a model can be error-prone and time-consuming. Moreover, controlling the direction and magnitude of the deformation can often prove difficult. It is our belief that directly painting the 3D surface to define its temporal behavior before interactively simulating its motion offers a more controllable, powerful way of expressing surface deformations than applying external forces.

A third class of modeling interfaces involves physically based deformable models [37]. Generally speaking, these are surfaces that minimize an energy functional derived from a physical description of the model's properties (e.g. elasticity, plasticity, etc.). Much like virtual sculpting interfaces, interactive editing of deformable models generally involves positioning a virtual tool near the model's surface that applies external forces, which disrupt the energy functional in a controllable way [36, 38, 35, 7, 33]. As with virtual sculpting, these modeling interfaces also suffer from their difficulty at editing large, complex models at various levels of detail with 3D virtual tools. Also, the complexity of the dynamics can sometimes make the computations too expensive for interactive manipulation. Alternatively, using purely geometric approaches to deform models has been investigated [4, 6, 20, 33]. Although these techniques offer direct manipulation of the model's surface, they too lack powerful control over fine detailed alterations and fail to easily support various levels of resolution.

Our approach also relates to other work that investigates physically simulating surface velocity as a modeling technique. One such approach, [27], provides a set of surface velocities for smoothing, embossing, and globally diffusing a level set model [29]. However, the user is asked to construct a surface speed function in $\Re^3$ and can control the part of the surface to be deformed only by providing a super-ellipsoid region of influence. Although we agree that simulating surface speed is a natural way to affect shape changes, this interface lacks a physically intuitive connection between the speed function and the surface to be deformed. We address this problem by specifying the velocity function directly on the surface.

Using simulation as a modeling tool was also examined in [10] where they describe a system for procedurally authoring solid models. They, in fact, describe "painting" onto a surface as a means of directly modulating surface velocity, a source of inspiration for this work. They also explain how to generate turbulent surface layers by procedurally modulating the velocity of the current surface. However, their system does not develop the idea of exposing this painting metaphor as the user's primary modeling tool nor does it address issues of interactivity.

There has been previous work in using 2D painting metaphors to manipulate 3D models. The *Pointshop 3D* system [45] allows interactive editing of a point-based representation. The user can specify normal displacements using either a paint brush to modulate the offset distance or with virtual chisel tools. The work of [28] uses a painting metaphor to create and manipulate a layered depth image. Generlizing *Adobe Photoshop's* [2] approach to 2D image editing, their system supports editing at different layers of the scene either directly or with a *clone brushing* tool. Furthermore, [39] explores the potential of describing the surface of a height-field by directly specifying a 2D shaded image from a known viewpoint. After constructing the shaded image for known lighting conditions using traditional 2D image editing techniques, the surface can be extracted using a shape from shading vision approach. These techniques differ from our approach in that they do not use a painting interface to control a physical simulation. As a result, their changes are largely restricted to a small class of geometric deformations comprised mainly of normal offsets.

Lastly, we note the similarities between our interface and the use of displacement maps [8]. Several commercially available packages [3, 34] even allow the user to directly "paint" displacement maps onto the surface of an object. Much like these techniques, our painting interface can be used to add detail to an existing model. One difference, however, is that we adaptively refine the underlying surface mesh to guarantee the resulting geometry will match that intended by the user. With displacement maps, this procedure must be done manually. Also, our modeling interface allows arbitrary deformations to a model's surface that supersede normal offsets of the vertices and certain operations, such as surface smoothing, are more naturally expressed in terms of surface velocity.

# 3 Basic Approach

Our approach gives the user direct interactive control over surface deformations. We control the instantaneous velocity of the surface and, consequently, the resulting change in its shape, using a metaphor that allows the user to paint the surface in order to define its motion. The user then interactively simulates this surface motion until the desired change is met.

In our system, the modeling process consists of repeating the following steps:

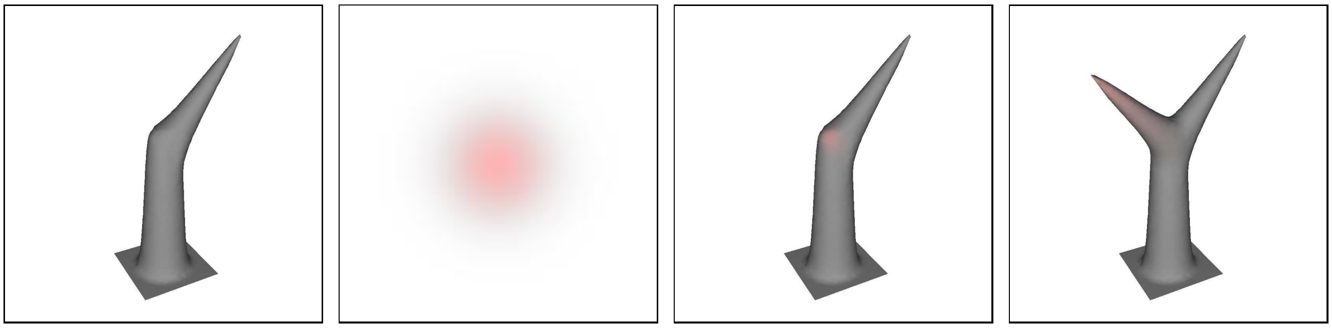
- Beginning with a base model (Figure 2(a)), select the paint and brush that will give the desired deformation (Figure 2(b)).
- Apply the paint to the model using the selected brush and a direct painting interface (Figure 2(c)).
- Simulate the motion of the surface until the desired effect is realized (Figure 2(d)).

The advantages of this approach are five-fold. First, surface velocity is naturally a good way for humans to "think" about surface deformations, as changing the shape of a model easily relates to moving its surface at different relative speeds. Second, our system provides local control over surface deformations: only the painted area of the surface moves. Third, our interface allows the user to define surface deformations as a 2D function on an existing surface, which is easier to understand and to use than ones requiring the user to move points through 3D space. Fourth, our system displays the simulated motion of the surface at interactive rates, which allows the user to "see" what is taking place to the model's shape and stop when the desired change is met. Finally, many useful modeling operations have natural specifications in the context of surface velocity. For example, smoothing/sharpening a model's surface is easily framed in terms of curvature-dependent surface velocity, whereas accomplishing this same effect by manipulating control points would be far more difficult.

# 4 Research Challenges

In realizing an implementation of a system based on this painting metaphor, several challenges must be met:

- **Defining Paint:** The first challenge is defining the relationship between paint and surface velocity. We



(a) Original Model (b) Paint (c) Painted Model (d) Result After Simulation

*Figure 2. (a) In our system the user begins with a base model. In this example, it is the extended trunk of a stylized tree for a cemetery scene. (b) Next, the user selects the paint that will give the surface a desired type of deformation and a brush to modulate its intensity. Here the user has selected a paint-brush pair whose resulting motion will form a spiky protrusion. The paint creates positive speed of the surface (with respect to the surface normal) and is visualized in bright red. (c) Next, the user directly paints the surface of the model where they desire the simulated motion to occur. (d) Lastly, the user interactively simulates the motion of the surface until the final result is achieved. In this case, the user simulated the surface's motion until the model included a branch of the desired length. Notice how the paint has moved with the surface during the physical simulation.*

would like it to support *any* deformation of a surface while making the more useful deformations the easiest to express.

- **Applying Paint:** We wish to maximize the ease the user experiences in applying paint. This implies creating an interface that supports very generic distributions of paint over the surface.
- **Evolving the Surface:** To support interactive deformations of the surface, we must evolve the surface efficiently, while maintaining a stable representation of the model.
- **Surface Representation:** We need a dynamic representation of the model. This should allow the simulation to execute quickly, maintain a high quality representation of the model during the simulation, and support adaptive levels of detail.

The remaining sections describe how we address these challenges.

## 5 Defining Paint

We are interested in a simple model of paint to control the instantaneous velocity of freely moving surfaces. We describe surface velocity at some point along the model's surface, $\mathbf{x} \in \Re^3$, with surface normal, $\mathbf{n}$, as the linear combination of three terms:

$$\mathbf{v}(\mathbf{x}) = \mathbf{v}_{prop}(\mathbf{x}) + \mathbf{v}_{adv}(\mathbf{x}) + \mathbf{v}_{curv}(\mathbf{x}), \quad (1)$$

where each term is defined as follows:

- **Propagating velocity** causes the surface to move at a constant speed in the direction of its surface normal:

$$\mathbf{v}_{prop}(\mathbf{x}) = \alpha\mathbf{n} \quad (2)$$

When simulated, this type of surface velocity results in blobby, organic deformations. As an example, the overall shape of the muffin's top was created by simulating propagating velocity whose speed smoothly varied over the top of a cylindrical solid (Figure 1(a,b)).

- **Advective velocity** causes the surface to move at a constant speed in a constant direction:

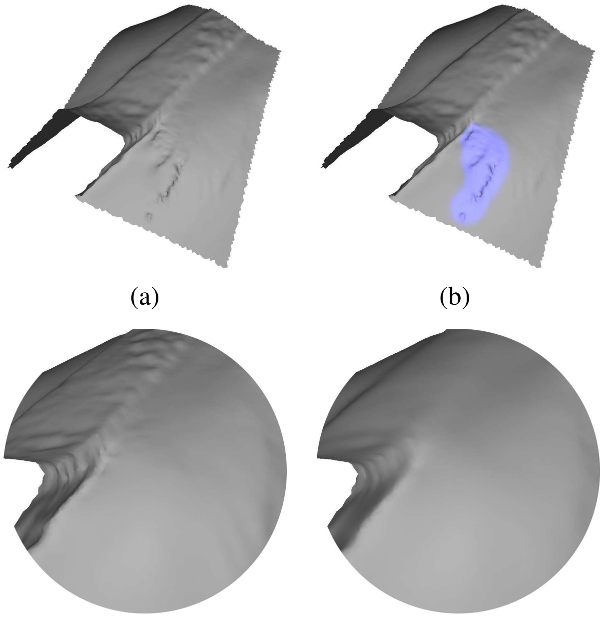
$$\mathbf{v}_{adv}(\mathbf{x}) = \beta\mathbf{p} \quad (3)$$

This type of motion gives rise to discontinuous geometry. Traditional sculpting tools that displace vertices a certain distance along some axis are, in fact, simulating purely advective motion of a surface over a single, fixed time step. Another example of simulating this type of velocity is the spiky tree limb created in Figure 2.

- **Curvature-dependent velocity** causes the surface to move at a speed proportional to its mean curvature, $\kappa$, in the direction of its surface normal:

$$\mathbf{v}_{curv}(\mathbf{x}) = \gamma\kappa\mathbf{n} \quad (4)$$

This type of surface velocity can be used to smooth irregular meshes [11, 27]. In Figure 3 we use curvature-dependent velocity to locally smooth a noisy scan from the Digital Michelangelo Project [21].



(a) (b)

(c) (d)

*Figure 3. (a) Starting with a noisy scan of the right leg of Michaelangel's statue of David, (b) the user applies curvature-dependent paint to the corrupted area. (c) After physically simulating the implied motion, only the noisy area is repaired. (d) An alternative approach, such as global diffusion would remove both the noise and the detail in David's sling.*

The total velocity of a point on the model's surface is then:

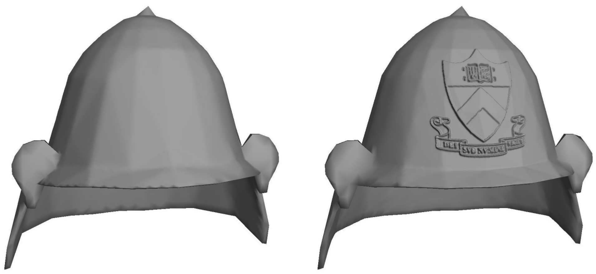
$$\mathbf{v}(\mathbf{x}) = \alpha\mathbf{n} + \beta\mathbf{p} + \gamma\kappa\mathbf{n} \tag{5}$$

Every point on the model's surface can contain an element of paint, consisting of the parameters: $\alpha$, $\beta$, $\gamma$, and $\mathbf{p}$ (called the "pigment vector") that fully describe its instantaneous velocity. The velocity is assumed to be zero over un-painted regions of the surface. Defining a surface deformation consists of mixing propagating, advective and curvature-dependent paints and applying this mixture directly to the model's surface before interactively simulating its motion.

## 6 Applying Paint

The success of our modeling interface clearly depends on the user's ability at controlling the distribution of paint along a model's surface. After the work of [16], we directly paint the object by projecting a 2D image into the scene from the current center of projection (Hanrahan and Haeberli call these "screen-space brushes").

In our system, the user first selects the paint that will give rise to the desired type of surface deformation (i.e. the user adjusts $\alpha$, $\beta$, and $\gamma$) and then chooses a 2D image to use as the brush. Next, the user directly paints the model by drawing in the image plane with the selected brush.



(a) Before (b) After

*Figure 4. Simulating advective motion where the direction of the velocity is the existing surface normal can create embossing effects. In this example, the user adds the Princeton emblem to a kabuto by painting positive advective paint onto its crown.*

Additionally, we give the user direct control over the paint's pigment vector, $\mathbf{p}$. In our current implementation, the direction of advective velocity (if any) can be taken as the surface normal where the paint was applied to the model, the viewing vector for the current camera position, or as an arbitrary global vector. These options help easily express directed motion of the surface. One application of defining the pigment vector at each point to be the current surface normal is embossing (Figure 4).

We find the process of creating 2D brush images and painting directly onto a 3D surface to be a powerful and easy way of expressing surface deformations. Although not explored in our system, other techniques that advance the idea of direct texture painting could easily be incorporated [19]. However, unlike texture painting, we are interested in giving each vertex a physical property as opposed to decorating the object by interpolating images across faces of the mesh. Therefore, considerations about texture space warping are not directly related to our approach. Instead, we pay particular attention to the mesh's vertices because we directly sample the projection of the 2D image at these points. Accurately sampling this projection often requires adaptive refinement of the mesh, where adding a sufficient number of vertices to sample this projection guarantees that the resulting geometric detail will match the deformation implied by the model's paint.

## 7 Evolving the Surface

Once the instantaneous surface velocity has been painted onto the model, the user interactively simulates the implied motion. We gain interactivity by explicitly integrating these simple equations of motion and updating the painted vertices at each time step. Maintaining a stable mesh is achieved through a heuristic that scales the speed of the surface by the average length of the painted edges. This provides fine control (small displacements) for highly-detailed areas and coarse control (large displacements) for less detailed portions of the model. By scaling the speed of the surface, we effectively normalize the values of the paint parameters ($\alpha$, $\beta$, and $\gamma$) to lie between -1.0 and 1.0 and the time step to lie between 0.0 and 1.0 for any level of detail. We also use the normalized mean curvature [11] that permits explicit integration, although our current system does not compute curvature at boundary points.

In our current implementation, the user can set the time step to any value between 0.0 and 1.0 and advance the simulation with either the keyboard or mouse. This provides interactive control, allowing the user to "see" the surface move and stop once the desired deformation is achieved. Additionally, we offer the facility to "undo" a modeling operation by saving copies of the model before the user deforms its surface and tracking the user's painting operations.

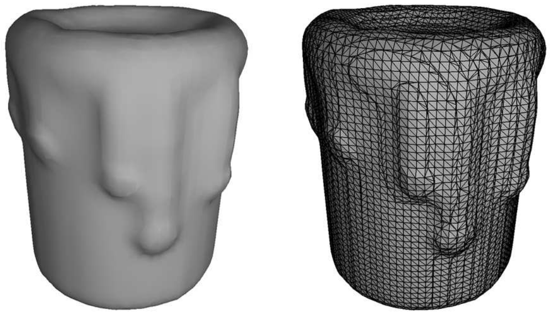
## 8 Surface Representation

We wish to accurately and efficiently represent a dynamic surface that moves at interactive rates. We have implemented our painting interface with two types of surface representations: level sets and polygonal meshes. In the end, we found dynamic polygonal meshes more suitable for our interactive application.

### 8.1 Level Sets

Level set methods [29] describe a dynamic surface as the zero level set of a time-dependent function. The main advantages of this approach are its elegant handling of topological changes to the surface and its ability to uniformly sample a moving surface. To incorporate level sets in our modeling system, we need an implicit representation of the surface paint. We accomplish this by defining a "paint volume" such that paint on the surface at some point **x** corresponds to the value at the same location in the paint volume. By warping this paint volume after each time step according to the changes in the implicit representation of the model, we cause the paint to move with the surface. Furthermore, we must maintain a paint volume with at least twice the resolution of the model itself in order to prevent "bleeding" of the paint from one side of the model to another.

Although our level set prototype system robustly handles arbitrary topology changes, its main drawbacks are that it cannot support both high resolution and interactive updates (Figure 5). Narrow band level set techniques [1] ensure that their simulation time is proportional to the number of voxels that intersect a model's surface and we update the narrow band only along the painted regions. Nevertheless, when creating models of varying complexity, a level set approach must sample the signed distance field at its finest resolution. Furthermore, taking an approach where only the painted region is converted into a level set representation does not work either as the surface deformations can cover large distances and a majority of the model is often painted. At each time step, we must also extract and display the zero level set, [23], which further decreases the update rate of the simulation. While multiresolution level set methods might address these issues, using them in an interactive modeling tool is still a future topic of research.



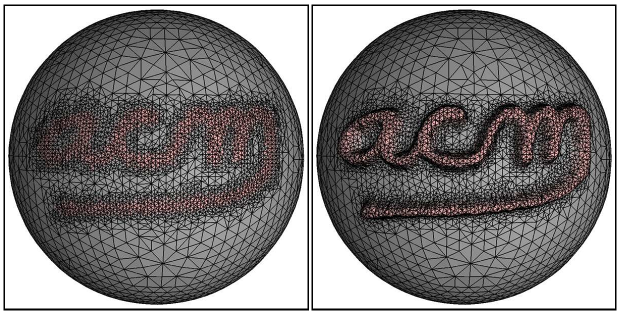
(a) (b)

*Figure 5. (a) A melting candle created with a prototype system of our modeling interface using a level set representation with a volume grid of 100x100x200 voxels. (b) The lack of adaptive resolution in the level set model prohibits the user from editing the model at various levels of detail. Moreover, this simple model took approximately 45 minutes to create due, in part, to the slow simulation speeds permitted by level sets.*

### 8.2 Polygonal Mesh

Alternatively, the moving surface can be represented as a polygonal mesh where the vertices are free to move in space. Because updating polygonal meshes is an inexpensive operation, they can easily support real-time simulation of the surface's velocity. The paint can also be stored at each vertex, avoiding the need for an implicit paint volume as required for a level set approach. Also, commonly available graphics hardware renders polygonal meshes much more efficiently than volumetric representations.

With polygonal meshes we can support adaptive resolution of the dynamic model. This is necessary while the user paints the model's surface because the distribution of vertices in the brush's 3D projection must accurately sample its 2D image. Ignoring this situation can result in aliasing of the paint along the model's surface creating undesirable affects on the resulting deformation. The reason for this is that high frequency components in the brush's image convey quickly varying relative speeds that will create high frequency geoemtric components along the model's surface. To prevent undersampling the brush image, we locally refine the mesh until the location of vertices along the model's surface matches the resolution of the image projection (Figure 6(a)). This refinement guarantees that the mesh's complexity matches the geometric detail in the resulting surface deformation (Figure 6(b)).



(a) (b)

*Figure 6. (a) We subdivide the mesh as a function of the painting direction and location. This refinement guarantees that vertices along the model's surface accurately sample the brush bitmap. (b) This results in a mesh whose complexity matches the geometric detail of the deformation.*

We also refine the mesh during the simulation. In order to maintain an even sampling of vertices along the model's surface we perform edge splits, edge collapses, and edge swaps [41, 25, 17]. These operations maintain roughly constant edge lengths while maximizing the minimum interior angles of the faces.

The main drawback of using polygonal meshes is their difficulty in handling events like topological changes and self-intersections. Although our current implementation does not explicitly handle either case, techniques do exist to prevent self-intersections, [25], and could be added to our modeling system. For our purposes, however, we found the fast simulation rate that meshes permit more important than their difficulty in handling these degeneracies.

The advantage polygonal meshes provide over level sets are gained through their adaptivity and by allowing an explicit representation of the surface paint. At each time step far more voxels must be updated as compared to the corresponding number of vertices in an adaptive polygonal mesh representing the same surface. Moreover, a polygonal mesh representation avoids the implicit paint volume required with level sets because the paint can be stored explicitly at each vertex. Lastly, commonly available graphics hardware favors efficient rendering of polygonal meshes as opposed to volumetric representations.

## 9 Results

We have found our painting interface useful for creating certain stylized scenes, adding detail to existing models at various scales, and locally smoothing noisy models. The following results were created with our system:

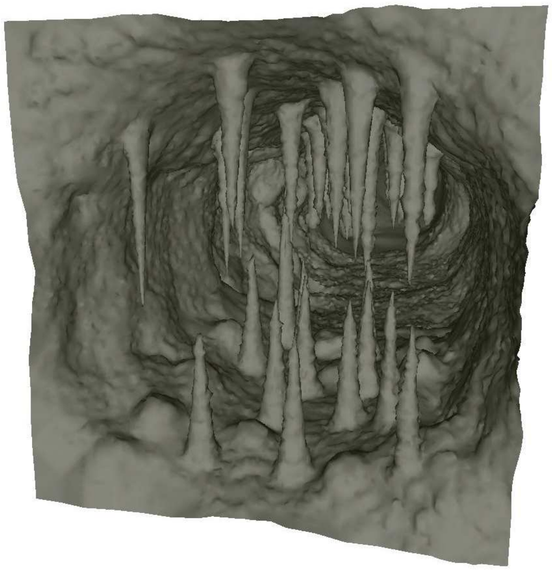


*Figure 7. Starting with a plane, this entire model was created with our painting interface.*

- **Entrance to a Cave:** (Figure 7) Starting with a single plane, we grew the overall shape of the cave's entrance using advective paint before growing the spiky stalactites and stalagmites. Lastly, we added the bumpy texture using a noisy brush bitmap and a combination of advective and propagating paint. This example highlights the usefulness of painting on many parts of the model at one time and then interactively simulating its motion. Sculpting techniques would require individually moving the model's surface for each spike. This highly detailed scene would also pose problems for a level-set approach that uses constant-sized voxels to represent the model.
- **Stylized Cemetery:** (Figure 8) This stylized cemetery scene was created quickly with our modeling system. Starting with a base plane, we first created the tree trunk using positive advective paint and then filled in its crown of branches using brush-dependent refinement and more advective paint. The gravestones were

grown off the base plane using a smoothly varying brush bitmap of a rectangle that decreased in intensity at its ends. The engravings on the gravestones show how brush-dependent refinement can create highly detailed geometry on an initially simple object. Other examples of multi-resolution deformations include the grassy texture of the ground and the knot in the trunk of the tree.

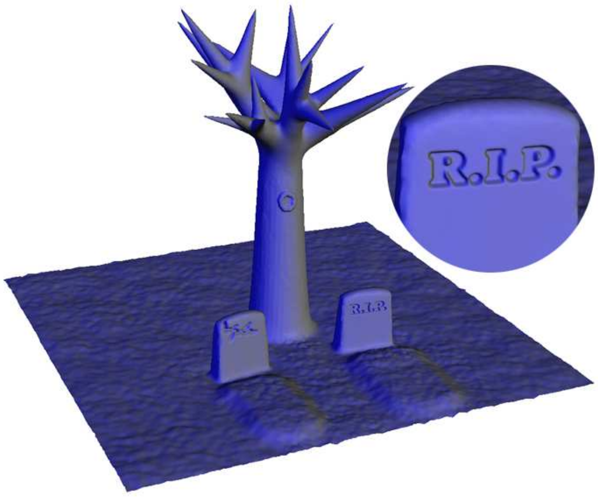



*Figure 8. This stylized cemetery was created by "painting" surface deformations onto a model.*

- **Melting Candle:** (Figure 9) This candle was created by first growing its overall shape from a base plane using advective paint. Next, we added an indentation to its top using negative advective paint. To give the model a melting look, we added the wax dripping down its sides by simulating a mixture of propagating and advective paint modulated with a small brush bitmap the user manually moved along the side of the candle. The pool of wax around the candle's base was created in the same way. Lastly, we added a wick to the candle's top using a small circular brush. This example highlights the main advantage of polygonal meshes over level sets. The melting candle created with the level sets system (Figure 5) does not have nearly the resolution of this candle because interactively simulating its motion would not be feasible for larger volumetric representations.
- **Modifying a Terrain Scene:** (Figure 10) In this example we used a mixture of advective and propagating paint to add a deep ravine to an existing terrain scene. This highlights another useful feature of our painting interface: easily expressing large-scale deformations. With other modeling techniques, this deformation would have to be formed incrementally, requring the user to move along the location of the ravine, manually deforming the surface appropriately. With our interface, however, the user was able to paint the location of the deformation all at once before simulating its motion. Therefore, expressing a large-scale deformation that involves re-positioning hundreds of vertices spread across the entire model was as simple as expressing a small, local surface deformation.

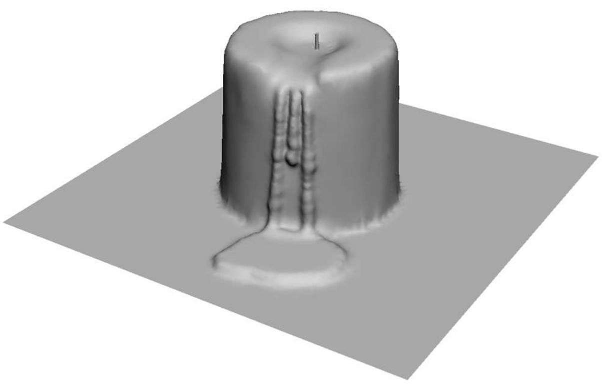


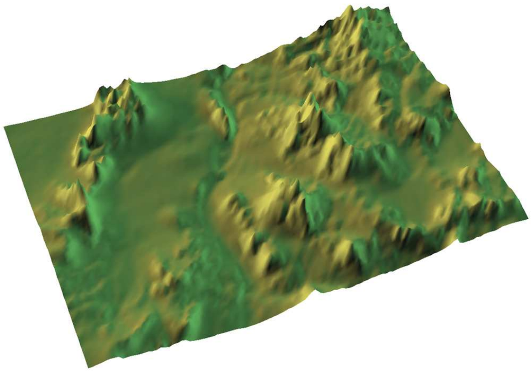
*Figure 9. Compared to the melting candle modeled with a level set approach (Figure 5), this example demonstrates the fine resolution that meshes can provide while maintaining interactive simulation rates.*

- **Locally Smoothing a Noisy Model:** (Figure 11) This example demonstrates the usefulness of providing local surface deformations. Starting with a model of a human male foot acquired from the Visible Human project [22], the user has painted the undesirable wire running along the top of the foot with curvature-dependent paint. The user then simulates this motion, smoothing away this unwanted artifact. Other, more global, diffusion techniques would have removed the detail in the rest of the foot and toes, demonstrating the usefulness in providing local control over a smoothing process.

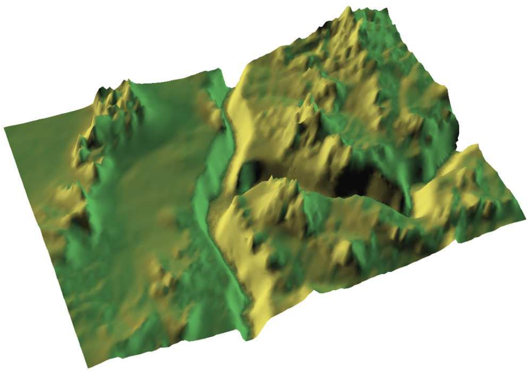
## 10 Conclusion

In this paper, we have described a new interface for interactive 3D modeling in which a user specifies deformations by painting directly onto the model's surface. This approach combines direct manipulation with physical simulation in an interactive modeling tool. Our implementation includes several tools for applying paint to surfaces and algorithms for evolving a mesh data structure and a level set model as a surface deforms.

Although our modeling interface does not outperform existing techniques in all scenarios, it does provide unique



(a) Original



(b) Modified

*Figure 10. (a) The user starts with a terrain scene. (b) Using a mixture of negative advective and propagating velocity, they create a deep ravine that cuts across the terrain.*

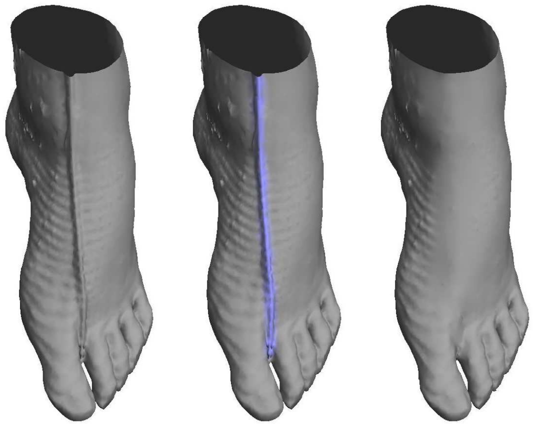
control over an interactive simulation useful for creating new models, removing noise from existing models, and adding texture to an existing surface at multiple resolutions.

Topics for future work include:

- Utilizing graphics hardware to accelerate the simulation of dynamic polygonal meshes.
- Investigating adaptive level set methods.
- Investigating the potential of describing geometry as simulated surface velocity for compression applications.
- Using paint to transfer geometric texture from one part of a model to another.
- Developing a constructive theory of geometry for how paint texture relates to resulting geometric detail.
- Extending the current interface to support time-dependent pigment vectors.

## References


[1] D. Adalsteinsson and J. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*,

(a) (b) (c)

*Figure 11. (a) A model of a human foot has an undesirable wire running along its top. (b) The user applies curvature-dependent paint (visualized in bright blue) only to this area. (c) Next, the user interactively simulates the implied motion to remove this unwanted artifact.*


pages 269–277, 1995.

[2] Adobe. Photoshop, 2003.

[3] Alias—Wavefront. Maya, 2000-2003.

[4] A. H. Barr. Global and local deformations of solid primitives. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 21–30, 1984.

[5] J. R. Bill and S. K. Lodha. Computer sculpting of polygonal models using virtual tools. Technical Report UCSC-CRL-94-27, 1994.

[6] P. Borrel and A. Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics (TOG)*, 13(2):137–155, 1994.

[7] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 257–266. ACM Press, 1991.

[8] R. L. Cook. Shade trees. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 223–231. ACM, 1984.

[9] S. Coquillart. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196. ACM, ACM Press, 1990.

[10] B. Cutler, J. Dorsey, L. McMillan, M. Mller, and R. Jagnow. A procedural approach to authoring solid models. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 302–311. ACM, ACM Press, 2002.

[11] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324. ACM, ACM Press/Addison-Wesley Publishing Co., 1999.

[12] D. R. Forsey and R. H. Bartels. Hierarchical b-spline refinement. In *Proceedings of the 15th annual conference on*

*Computer graphics and interactive techniques*, pages 205–212. ACM Press, 1988.

[13] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 249–254. ACM Press/Addison-Wesley Publishing Co., 2000.

[14] J. Gain. Virtual sculpting: An investigation of directly manipulated free-form deformation in a virtual environment, 1996.

[15] T. A. Galyean and J. F. Hughes. Sculpting: an interactive volumetric modeling technique. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 267–274. ACM Press, 1991.

[16] P. Hanrahan and P. Haeberli. Direct wysiwyg painting and texturing on 3d shapes. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 215–223. ACM, ACM Press, 1990.

[17] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26. ACM Press, 1993.

[18] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 177–184. ACM, ACM Press, 1992.

[19] T. Igarashi and D. Cosgrove. Adaptive unwrapping for interactive texture painting. In *2001 ACM Symposium on Interactive 3D Graphics*, pages 209–216. ACM, March 2001.

[20] F. Lazarus, S. Coquillart, and P. Jancene. Axial deformations: an intuitive deformation technique. *Computer-Aided Design*, pages 607–613, August 1994.

[21] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144. ACM Press/Addison-Wesley Publishing Co., 2000.

[22] W. E. Lorensen. Marching through the visible man. In *IEEE Visualization*, pages 368–373, 1995.

[23] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM Press, 1987.

[24] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 181–188. ACM Press, 1996.

[25] L. Markosian, J. M. Cohen, T. Crulli, and J. Hughes. Skin: a constructive approach to modeling free-form shapes. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 393–400. ACM Press/Addison-Wesley Publishing Co., 1999.

[26] S. Mizuno, M. Okada, and J. ichiro Toriwaki. Virtual sculpting and virtual woodcut printing. *The Visual Computer*, pages 39–51, 1998.

[27] K. Museth, D. E. Breen, R. T. Whitaker, and A. H. Barr. Level set surface editing operators. In *Proceedings of SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, pages 330–338. ACM, ACM Press / ACM SIGGRAPH, 2002.

[28] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 433–442. ACM Press, 2001.

[29] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[30] R. N. Perry and S. F. Frisken. Kizamu: a system for sculpting digital characters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 47–56. ACM Press, 2001.

[31] L. Piegl. On nurbs: a survey. *Computer Graphics and Applications*, 11:55–71, 1991.

[32] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160. ACM, ACM Press, 1986.

[33] K. Singh and E. Fiume. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 405–414. ACM Press, 1998.

[34] Softimage. Xsi, 2000-2003.

[35] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26(2):185–194, 1992.

[36] D. Terzopoulos and K. Fleischer. Deformable models. *Visual Computer*, pages 306–311, 1988.

[37] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214. ACM Press, 1987.

[38] D. Terzopoulos and H. Qin. Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Transactions on Graphics (TOG)*, 13(2):103–136, 1994.

[39] C. van Overveld. Painting gradients: Free-form surface design using shading patterns. In W. A. Davis and R. Bartels, editors, *Graphics Interface*, pages 151–158. Canadian Human-Computer Communications Society, 1996.

[40] S. W. Wang and A. E. Kaufman. Volume sculpting. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 151–ff. ACM Press, 1995.

[41] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 247–256. ACM Press, 1994.

[42] J. P. Y. Wong, R. W. H. Lau, and L. Ma. Virtual 3d sculpting. *Journal of Visualization and Computer Animation*, 11(3):155–166, 2000.

[43] D. Zorin and P. Schröder. Eds. course notes: Subdivision for modeling and animation, 1998.

[44] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 259–268. ACM Press/Addison-Wesley Publishing Co., 1997.

[45] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: an interactive system for point-based surface editing. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 322–329. ACM Press, 2002.