Driving Object Deformations from Internal Physical Processes

Zeki Melek[*]
Department of Computer Science
Texas A&M University

John Keyser[†]
Department of Computer Science
Texas A&M University

Figure 1: A bending match while burning.

Abstract

In this paper we present a method for deforming objects for graphics applications, based on the results of internal physical simulations. As driving examples, we describe in detail methods for simulating the bending of burning matches, and the crumpling of burning paper. In these cases, the small-scale changes in a chemical process result in large-scale deformations of the given object. We propose the use of a free form deformation to model such large-scale deformations. Changing object properties are mapped onto the edges of a proxy object, which is then modified by treating the edges as springs. This proxy object then serves as a control structure for defining the deformation of the underlying object. The results we present are fast, controllable, and visually plausible.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically Based Modeling

Keywords: physically based modeling, deformation

1 Introduction

In the push for greater realism in computer graphics applications, complex physical simulations are playing a larger and larger role. From real-life experience, users are often familiar with the physical phenomena being simulated, and different simulations have varying success at replicating the details of a physical process. Such details can make the difference between a believable virtual world that draws the user in and a jarring environment that destroys any sense of presence.

Many complex physical simulations require modeling of a number of different phenomena. Depending on the particular application, these phenomena can require different levels of accuracy in order to create an overall visually plausible result. Though an "ideal"

[*]e-mail: melekzek@tamu.edu
[†]e-mail:keyser@cs.tamu.edu

simulation might accurately simulate all details of all physical phenomena, this is usually impractical, given constraints on time, processing capability, and even underlying knowledge of the physical process. Instead, what is usually done in the computer graphics community is to simplify the physical model, use a simpler simulation, and eliminate certain secondary effects in order to achieve a plausible result in reasonable time. This is the case for all simulations, but it is particularly magnified in interactive applications.

The goal of our work is to develop a method for efficiently modeling certain secondary effects in physical simulations, thereby increasing visual plausibility of the overall simulation for only a reasonable cost in efficiency. We do this by attempting to model the large-scale effects of certain physical processes, rather than spending a disproportionate amount of computation on a minor yet potentially complex phenomenon. In particular, in this paper we propose a way of approximating larger-scale deformations of objects guided by the results of a simulated physical process.

An example of such complex physical processes is burning objects. To simulate a burning object, the combustion reaction, heat distribution, fuel consumption, and even object shape must be modeled and changed over time. The pyrolysis process, where an object releases combustible gases, causes decomposition and additional structural changes in burning objects. Although these structural effects are usually minor, some create a dominant deformation on burning objects. One such structural change is caused by microscopic contraction of fibers within a burning object. This results in effects that are quite noticeable at the macroscopic level, such as the way matches bend when burnt, and the way burning paper tends to crumple. We can simulate the combustion reaction, the object catching fire and burning, and even the decomposition of the object as it burns, but unless we model the fiber contraction, the simulated matches will not behave like actual burning matches (fig. 2).

In this paper we present a free form deformation (FFD) based method for approximating large-scale deformations due to smaller-scale physical simulations. We also show specific examples of how this approach can be applied to represent the deformation of burning objects according to the changing object properties encountered during the simulation.

The major contributions of this paper are

- We present a framework for creating deformations guided by physical simulations. This includes:

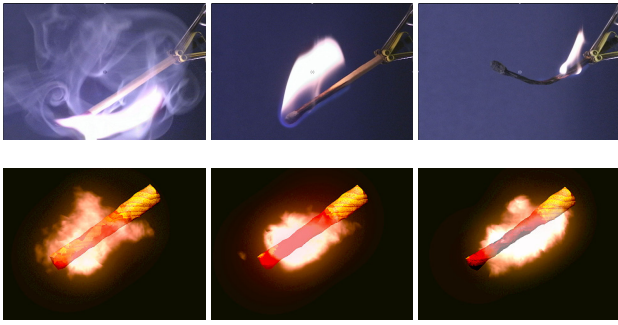 - Defining a proxy object. The deformations are simu-

Figure 2: Real vs simulated matches without bending.

lated on the proxy instead of the actual object.

  – Mapping simulation parameters to the proxy object.

  – Modifying the proxy object based on the simulation results.

  – Creating a deformation using the modified proxy object.

- We present the first models that give a physics-driven simulation of two physical phenomena, namely:

  – Bending burning matches.

  – Crumpling burning paper.

The organization of this paper is as follows. We start by discussing the background and related work (Section 2). In Section 3, we will describe our Free Form Deformation (FFD) based deformation guided by a physically based simulation. We then describe how this can be applied to two of our motivating cases, bending matches (Section 4) and crumpling paper (Section 5). We describe how this method can fit into a larger simulation framework in Section 6. Finally, we discuss drawbacks of our proposed method, and potential future work in Section 7.

2 Background

Burning Objects

Since our motivating case is deformations of burning objects, we will first look at related work in this area. We previously presented [Melek and Keyser 2003] a simple but effective model for simulation of the decomposition of burning objects using a levelset method. Losasso et al. [2006] model burning objects using remeshing, and can model fine scale decomposition structures. However, neither of these works addresses secondary deformation effects due to structural changes. Although this effect is minor for large objects, it creates a dominant deformation on small objects. Examples include the upward bending seen in burning matches and the crumpling of burning paper. The current state-of-the-art for creating such deformations involves an artist modeling the deformation manually.

Deformations related to physical processes

There has been some work on deformations related to physical processes other than burning. One example is given by O'Brien et al. [2002]; their work deal with fracturing solid objects. Yngve et al. [2000] combine shock waves together with the fracturing of solids. Carlson et al. [2002] solve the melting problem by treating the solid as a liquid with high viscosity that changes in space and time, allowing the same material to exhibit different states. Modeling internal stresses on a moving viscoelastic fluid is presented by Goktekin et

al. [2004] using a levelset method [Sethian 1999; Osher and Fedkiw 2002; Fedkiw 2002].

Clay-like deformations

Many different methods have been proposed for modeling clay-like deformations including Octrees [Bærentzen 1998], Freeform Deformations (FFDs) [Coquillart 1990], Finite Element Models (FEMs) [Wang and Kaufman 1995; Galyean and Hughes 1991], Adaptively Sampled Distance Fields (ADFs) [Bremer et al. 2002], and levelsets [Bærentzen and Christensen 2002]. Different user interfaces have been presented to control the deformations [McDonnell and Qin 2002; Bremer et al. 2002; Llamas et al. 2003]. Recently, vector fields have been used to control deformation on surfaces [von Funck et al. 2006]. Capell et al. combine finite element methodology with a multiresolution subdivision framework for interactive simulation of elastic deformable solids [Capell et al. 2002a] using trilinear subdivision [Capell et al. 2002b].

Imprints and tracks

Another group of deformation work models imprints and tracks on the ground. Methods used for doing so include bump maps [Lundin 1984; Lundin 1994], heightfields [Sumner et al. 1998], particle "blobbies" [Terzopoulos et al. 1989][Terzopoulos et al. 1991], and elastic sheets [Chanclou et al. 1996]. ADFs have been used to detect collision of soft bodies [Frisken and Perry 2001]. The contact deformation is modeled using forces defined inside the overlapping regions. Some methods also include models of soil dynamics [Li and Moshell 1993; Onoue and Nishita 2003]. Similarly, soft body deformations of colliding objects have been investigated [Gascuel et al. 1991; Dewaele and Cani 2003; Pauly et al. 2004].

Free-form deformation

Free-Form Deformation (FFD) tools are widely used in soft-object animation and shape deformation. The points on the object are represented as an affine combination of the vertices of the encapsulating shape around the object. An overview of deformable modeling can be found in [Gibson and Mirtich 1997]. Barr [1984] introduced a set of hierarchical transformations for deforming an object, including twist, bend and taper operations, using the position vector and surface normal of the undeformed object and a transformation matrix. Each level in the deformation hierarchy requires an additional matrix multiplication. Sederberg and Parry developed a technique for deforming the solid geometric models in a free form manner [Parry 1986; Sederberg and Parry 1986]. Their method could be applied to quadrics, CSG based models, parametric surface patches, or implicit objects with derivative continuity. The FFD is usually defined in terms of a tensor product trivariate Bernstein polynomial. Coefficients of the Bernstein polynomial are the control points. Coquillart presented an extension to this method, Extended FFD (EFFD) [Coquillart 1990], adding arbitrarily shaped bumps or bending the object along an arbitrarily shaped curve using B-spline control points. Chang and Rockwood [Chang and Rockwood 1994] deform objects using a Bezier curve and affine maps controlling handles.

Interpolation schemes

Affine combinations of vertices encapsulating a shape form an interpolation scheme. Vertex positions could be used as data values in these interpolation functions to form deformations. Usually, these methods are used for deformations defined by a user. Many interpolation schemes have been proposed [MacCracken and Joy 1996; Kobayashi and Ootsubo 2003]. Warren et al. [1996; 2004] extend the Washpress interpolant for convex shapes into 3D. Ju et al. [2005] use mean value coordinates [Floater 2003] for 3D non-convex shapes. Similar results were produced concurrently by Floater et al. [2005].
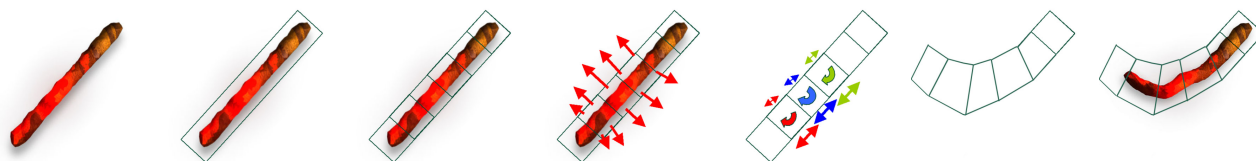
Figure 3: The proposed simulation driven deformation process. (a) initial object, (b) placing proxy, (c) subdivide along the deformation axis,(d) mapping parameters onto the proxy, (e) deformation defined on the proxy, (f,g) apply the deformation onto the initial object.

3 Simulation Guided FFD

Rather than modeling the actual chemical process fully, we propose a simplified model to mimic similar behavior for secondary deformations. Similar to the multi-representation framework proposed earlier [Melek and Keyser 2005], we propose using the change of object properties during the simulation to control the deformation. Note that we assume there is some physically based simulation determining the "major" processes acting on the object, and the effects of any "secondary" processes are either too complex to model, or too time-consuming to simulate. We therefore use a simplified deformation, guided by the primary simulation, to model the deformations created by these secondary effects, rather than modeling the effects themselves.

The overview of our proposed method is as follows (fig. 3)

- Run the simulation for the primary physical processes affecting the initial object.

- Place a proxy object around the deforming object of interest.

- Map the simulation results from the initial object onto the proxy object.

- Determine an approximated deformation for the proxy object using the mapped properties.

- Use the deformed proxy object to control a deformation applied to the original object.

In the remainder of this section, we will describe the elements in the most general terms. Later sections (section 4, 5) will show how this can be specialized to very particular situations.

3.1 Placing a Proxy Object

The basic idea is to define a simplified object that can later be used to deform the object of interest. We refer to this simplified shape as a "proxy object." Given a particular object or region of interest, there are many ways of defining a proxy object. Options include a bounding box, the convex hull of the object, a simplified skeleton of the object, or a user-specified simple shape. This proxy object simplifies the calculations required for deformation, such that we can define the deformation using the proxy and then apply the same deformation to the high resolution object.

In general, we will consider the proxy object to be a polygonal boundary representation that encapsulates the object of interest. However, it is certainly possible to use other proxy objects. For example, a tricubic spline function would be useful for some applications where deformations of the internal structure must be modeled. In other applications, a medial or skeleton-based representation might be useful. There is no particular restriction on the proxy object that can be used, other than that it must be usable in the later stages of the process; the later need to map properties onto the proxy and define a deformation from the proxy might influence the particular choice of a proxy object. Obviously, in most cases the proxy object should be significantly simpler in some sense (e.g. fewer polygons, lower genus, or more convenient representation) than the original object, since the idea is to simplify the simulation.

Note that while this is not a strict restriction, we will make an assumption that we have a single proxy object of genus zero. In reality, even if we start with a genus zero object, solids might undergo topological changes during complex physical simulations. We usually do not want to have to model such topological changes explicitly in the proxy, as it would defeat the simplicity we are seeking in the proxy. So, we assume that a genus zero proxy will be sufficient to represent any deformations and that any changes in the deformation due to topological changes in the underlying object will have only a minor effect. Clearly this is not always the case; if a single deformation cannot be used to model the underlying deformation, or if the proxy would need to change in genus, another method might be more appropriate.

3.2 Mapping Simulation Results

Given a proxy object and a simulation on the underlying object, the second step involves mapping the "interesting" simulation properties from the original object onto the proxy. The exact nature of this mapping will vary depending on exactly what proxy object was chosen. Note that we are not mapping the results of the deformation we are trying to simulate, rather we are mapping properties that can be used to later define the deformation. For instance in the examples we provide below related to burning objects, we map internal heat distribution and rate of pyrolysis onto the proxy faces. For a different simulation, we might map another parameter, e.g. air pressure, onto a surface.

If we assume that we have an encapsulating convex proxy object, the mapping process can be as simple as a cylindrical or spherical map. If a skeletal proxy is used, the mapping process can be a simple retraction to the skeleton. Note that this mapping onto the proxy is not necessarily one-to-one; several points of data from the underlying simulation can map to the same point on the proxy object, or none of the underlying points on the object might map to a particular point on the proxy. Also note that parameters could map to different parts of the proxy object. For example, for an encapsulating polygonal proxy, we could conceivably map parameters to the faces, or to the edges, or to the vertices. The part that we want to map to should be determined by the way the proxy object itself will be deformed, as described in Section 3.3.

3.3 Defining Deformation of the Proxy

The critical part of our proposed method is defining the approximated deformation on the simplified proxy object. However, it is difficult to give details for this process as this portion will be very specific to the particular simulation. The main point is that we take the parameters that have been mapped onto the proxy object, and use these to define a deformation of the proxy object itself.

A few examples of the ways that the mapped properties (the "values") might define a deformation on the proxy object include:

- The values can be used as constants for a spring system along edges of the proxy. The proxy object can then be simulated to equilibrium.

- The values can be used to define a local transformation (e.g. scaling, translation) of the proxy mesh. The deformed proxy is formed from the superposition of these local transformations.

- Weighted averages of the values can be used to determine parameters of the proxy. For example, a medial proxy representation could have the radius information set by the mapped parameters.

- Values can be used to apply weighting to particular points. For example, for a tricubic rational spline proxy, some values could adjust the positions of control points, and others the weights.

Generally, a "good" deformation will have these properties:

- *Similarity:* The deformation will have the same effect on the proxy that one would want it to have on the underlying object. That is, the proxy object should "behave" in the same way (though at a coarser level) as you would want the underlying object to behave in response to those simulation parameters.

- *Simplicity:* It will be significantly cheaper to compute than the simulation on the original object would have been. Note that you could potentially use the *exact* same deformation as would have been used on the original object, but save significant time by simulating over a much simpler object.

- *Control:* The user should be able to control the way the deformation behaves. Generally, this deformation is an abstract model for some more complex underlying process. Because of this level of abstraction, it is likely that a user will need to set some parameters manually (e.g. the ratio between a particular mapped property and the spring constant it defines on the mesh). In addition, user control of *parameters* allows greater artistic control in some cases, while still ensuring that the deformation is defined by the physical simulation.

We will discuss sample deformations on two different cases in detail in the following sections.

3.4 Applying the Deformation

Finally, we apply the deformation defined on the proxy to the encapsulated object. The proxy will define the deformation that will be used to warp the local coordinate system of the actual object. The specific way this is accomplished will depend on the way the proxy was defined.

For basic encapsulating polygon proxies, the simplest approach is to use the proxy object as a free form deformation (FFD) lattice around the deforming object. We can decompose the cells into tetrahedra [Ning and Bloomenthal 1993] and apply piecewise linear interpolation using barycentric coordinates inside the tetrahedra. This approach can be made arbitrarily more complex. If more continuity is desired, a nonlinear interpolation method could be used. Or, one could easily use mean value coordinates [Ju et al. 2005] or similar interpolation schemes to define a deformation from a given polygonal proxy.

Other types of proxies would afford different deformations. For example, tricubic spline proxies could directly define an FFD. Skele-

tal or medial proxies could be used as the basis for a distance-field based deformation of space.

3.5 Evaluation

This procedure outlined above allows us to define a deformation behavior similar to the desired physical process. Note again we are not simulating the actual physical process, but rather we use a computationally cheap approximation of the same phenomenon to give visually "plausible" results. The key issue of our proposed method is that by using a set of related object and simulation properties we are able to approximate our desired behavior as a simple deformation.

4 Bending Matches

We describe here how the deformation described above can be used to model the bending of burning matches.

Figure 4: Before and after simulation driven bend deformation.

Matches are made out of fibrous material oriented along the length of the match. During burning, the fibers lose water and other chemicals. Though there is still debate about the precise mechanism, this loss of material causes the fibers to contract at a microscopic level. Due to the shape of a flame, the upper part of a match receives more of the heat generated from the combustion reaction that forms the flame. Thus, it is hotter on the upper side of the match compared to the bottom; this means that the fibers on the top contract more than the ones at the bottom. This imbalance accumulated at the microscopic level forces the match to bend upwards at the macroscopic level. This is exactly the type of situation our proposed method was designed for: a simulation-driven deformation, where modeling the true process (fiber deformation at a microscopic level) is impractical, but the overall behavior is significant.

We begin by selecting a proxy object for the match. Due to the shape of the match, we use a simple bounding box aligned along the match axis as a proxy. This bounding box is subdivided into a number of individual segments along the bending axis (the length of the match), creating a 1x1xN FFD lattice surrounding the match.

During the simulation of burning and decomposition, the rate of pyrolysis $\frac{d}{dt}P$ and internal heat T of the object are mapped onto the faces of the proxy object. Each face of the proxy will store the average of the pyrolysis rate and heat values that map to it. Since the proxy object we used in the previous step is rectangular, we can just use a cylindrical mapping from the burning match onto the lattice. Note that this simplification does not take changing topology during

the decomposition process into account, but this should introduce only a minimal amount of error.

Before we define the deformation, we need to clarify one issue. We can define a simple cylindrical mapping at the start, but what will happen as we bend the match? Since the match will also decompose, the faces will change and we cannot fix the mapping. One thing to note here is that although the match is bending in world space, it is still undeformed in its own local space, and so is the proxy. Hence we can define the mapping from the object to the proxy in the unbent local object space.

At this point, we have our proxy object and some simulation values (pyrolysis rate and heat) stored at the faces, and must use that information to deform the proxy object. To achieve the bending behavior we are seeking, we consider opposing faces of the proxy object. The proxy faces are contracted or expanded according to the difference on facing faces. This individual contraction/expansion of the individual proxy cells defines a smooth deformation along the proxy object. This expansion/contraction of the opposing faces also causes rotation of the rest of the lattice.

$$D_T^{ij} = \Delta T_i - \Delta T_j \quad (1)$$

$$D_P^{ij} = \frac{d}{dt}P_i - \frac{d}{dt}P_j \quad (2)$$

$$D^{ij} = \begin{cases} \alpha D_T^{ij} + \beta D_P^{ij}, & P_i, P_j \geq P_{thresh} \ \& \ T_i, T_j \geq T_{thresh} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where i and j are opposing faces. T_i and P_i are the temperature and pyrolysis values mapped to vertex i. D^{ij} is the rotation amount. α and β control how much the heat and rate of pyrolysis affect the final deformation respectively; these are the parameters that allow an element of user control to the deformation. T_{thresh} and P_{thresh} are thresholds for starting the deformation. Note that the axis around which the rotation occurs is perpendicular to both the $\vec{ij}$ axis and the axis along the match length.
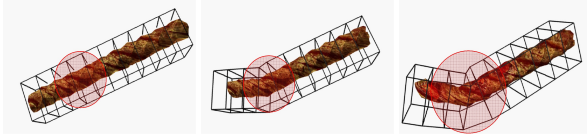

Figure 5: Curling match using simulation guided FFD.

On a rectangular proxy, we have two sets of opposing faces, hence two sets of deformation values for every cell (D_{ij},D_{kl}) for two axes orthogonal to the match length. We use those deformation values by rotating the rest of the cells centered around the "active" cell. Here, we can either apply rotation to both sides, or fix one side (if the match is anchored or held at one end) and apply the rotation to one side only (fig. 5).
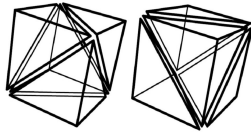

Figure 6: Two alternating configurations of tetrahedral elements.

Finally, we use our deformed proxy to deform the match itself. We subdivide the proxy lattice cells into tetrahedral elements [Ning and Bloomenthal 1993] (fig. 6) and apply piecewise linear interpolation using the barycentric coordinates inside the tetrahedra to deform the space around the burning match. Although this approach is only C^0 continuous, we have not observed any artifacts even with a fairly small number of segments ($N = 20$). This then creates the overall deformation that we were seeking.

Note in all of this the importance of the choice of proxy object. The particular proxy object we chose to use plays a big role in what mapping can be used, how the proxy object itself can be deformed, and how easy it is to define a deformation from the proxy. While a different proxy object could be used, it might require significant modification of the approach, particularly for determining how the mapped values are used to define the deformation.

5 Crumpling Paper

We describe here a second possible application, crumpling of burning paper, using a different type of proxy object. Like matches, paper is also made out of fibrous material. When paper is burned, the contraction of the fibers can cause the paper to crumple (the specific nature of crumpling depends on the fiber structure). A simple single axis lattice proxy object like that presented in the previous section would not be applicable here, because of the planar structure of a sheet of paper.
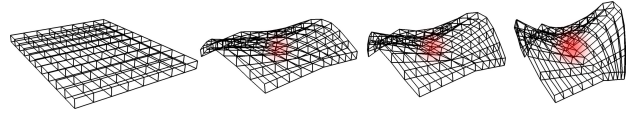

Figure 7: Deformation of the center cell on the "2D lattice". Note that the deformation falls off as we move farther from the deformation axis.

First, we choose a proxy object suitable for representing the deformation of the planar paper. We use an enclosing bounding box, subdivided along two axes to form a NxMx1 lattice around the object. We will call this lattice a "2D lattice". One should notice that this approach is suitable for a burning sheet of paper, but not for burning folded origami artwork.

Similar to the case of bending burning matches, the rate of pyrolysis and heat are mapped onto the faces of the proxy, with each face storing an average. In this case, however, we use an orthogonal mapping from the object to the proxy. The face normal of the object determines whether the properties of that face are mapped to the top or bottom of the proxy.

The proxy object is then deformed. Similar to the case for matches, we will use rotations defined by the difference in values on opposing faces. For the match bending case, a simple rotational scheme worked fine using the deformation D_{ij}, but we need to use a more complex scheme to deform our 2D lattice.

We will deform the proxy lattice by combining the deformation effects computed at each cell of the lattice. For a given cell of the lattice, i, we set an amount of deformation, D_i, by the same process as for D_{ij} in Equation 3. There are several possible ways to deform the proxy cell from this value. We choose a simple approach that works as follows:

- We choose an axis for the cell. The cell will cause the lattice to tend to "fold" about this axis, $\vec{u}$. Generally, we choose $\vec{u}$ to be one of the two lattice axes, passing through the center of the cell, though any axis through the center would be valid. We have not observed any difference in choosing one of the
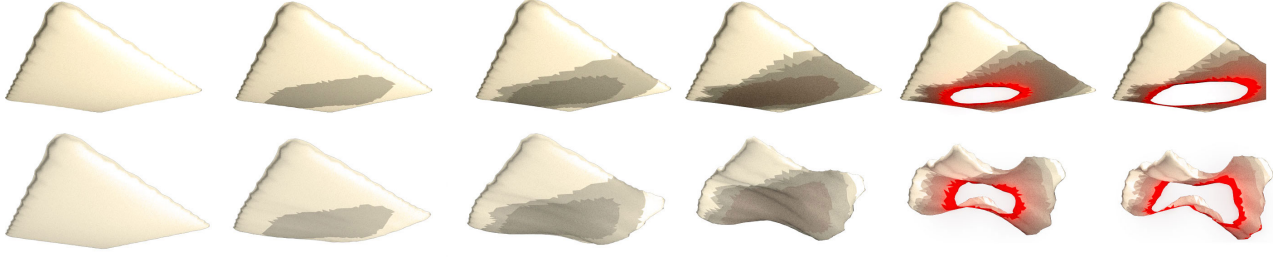
Figure 8: Crumpling paper during burning. (top) decomposition alone, (bottom) decomposition and crumpling. Flames are not shown.

two lattice axes randomly versus alternating the deformation axis between neighboring cells.

- We then define how the deformation in that cell, D_i, will affect other cells in the lattice. That is, we model the *result* of the deformation of that cell on the nearby cells. This will be defined as a rotation about the axis chosen above. The amount should be highest near the axis and fall off as one moves away from the axis. We choose to use a cosine function to describe this weighting (Eq. 5), though a different function (e.g. gaussian) could be used instead. This effect corresponds to the rotation around the deforming cell in the match case, except only cells near the axis of the deforming cell are modified.

- The deformation defined above will tend to cause stretching in the lattice as we move along the deformation axis away from the cell. To minimize this effect, we add a deformation that pulls the corners of the cells closer to the axis as we move along the axis away from the original cell (Eq. 6).

- At this point, we have defined a deformation function independently for each active cell. We then need to combine the contribution of these deformations within all the cells of the mesh. So, each cell of the lattice will end up in a new position defined by the deformation amounts given by the other cells (Eq. 4). Note again that after summing these deformations, we immediately have the final position of that cell - i.e. the result of the deformation.

To summarize, then, the new position of a vertex, V_j, in the lattice is determined as follows:

$$V'_j = V_j + \sum_i D_i * dist_u(i,j) * (D^1_{ij} + D^2_{ij}) \qquad (4)$$

$$D^1_{ij} = \alpha * cos(dist_v(i,j)) \qquad (5)$$

$$D^2_{ij} = \beta * dist_v(i,j) \qquad (6)$$

where $dist_u(i,j)$ is the distance *along* the deformation axis of cell i to vertex V_j, and $dist_v(i,j)$ is the distance of cell V_j *orthogonal* to the deformation axis of cell i. α and β are parameters that allow user control of the deformation - i.e. controlling the amount of deformation created by these effects.

The collection of individual deformations results in a deformation that mimics the crumpling action seen in burning paper. Note that for simplicity we omitted any explicit tests to avoid self-intersections; self-intersections were never seen as a problem in our examples.

Because of the additive deformations, we observe excessive stretching behavior around the corners (fig. 9). This is an expected result,
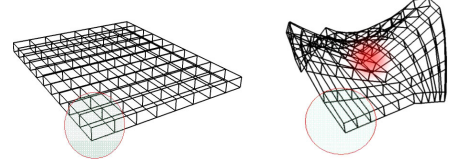

Figure 9: Excessive stretching at the corners.

since we are not using a volume preserving scheme. While controlling α and β can help reduce this, they are not sufficient. To avoid the artifacts that would result from such a stretching, a relaxation step is applied to pull back those over-stretched areas to create a more plausible deformation. This is done by setting a spring mass system with the rest lengths defined from the initial state, and iteratively allowing the system to come to a rest state. In practice, we have observed that just a couple of iterations of relaxation are sufficient enough to avoid noticeable artifacts from the stretching, but it can create oscillatory artifacts. We suggest to use relaxation only to prevent excessive stretching, but not as a global volume preserving scheme. Note that such a cloth-like solver could have been used as an alternative (but slower) method for deforming the proxy lattice.

Now that we have deformed the lattice mesh, the final deformation is then applied to the paper itself. Again, we can easily tetrahedralize the lattice, and use this to apply a linear FFD. A more complex deformation could be used but did not appear necessary in our tests.

Note again the significance of our choice of proxy shape. Besides requiring a different mapping from the previous example, the deformation process applied was also significantly more complex.

6 Integration

Generally, the proposed method should be fairly easy to integrate into an existing simulation system. The mapping of simulation data onto the proxy can occur at almost any point in the simulation. Care must be taken as to when to actually apply the deformation of the object, but there will usually be several points in a simulation process at which it is "safe" to adjust geometric data. To show how this can be done, we describe how the deformations we described earlier will fit into a general simulation of burning objects.

Our prior simulation framework consists of two main modules, along with a synchronization and data exchange interface. The two main components are a fire/flame simulation and a solid object simulation; each of these is composed of several sub-modules dealing with different phenomena.

A detailed overview of the fire simulation framework is beyond the scope of this paper. The fire module is responsible for air motion, gas distribution, and heat generation. It models the combustion pro-
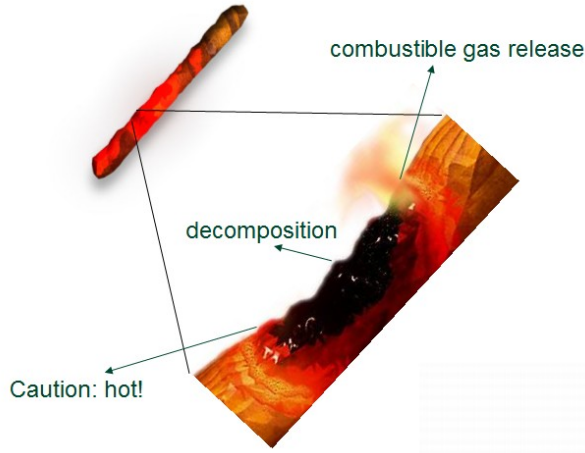
Figure 10: Decomposition process.

cess, which generates heat and drives the air motion. The solid objects module consists of heat transfer, object decomposition, and a rigid body solver.
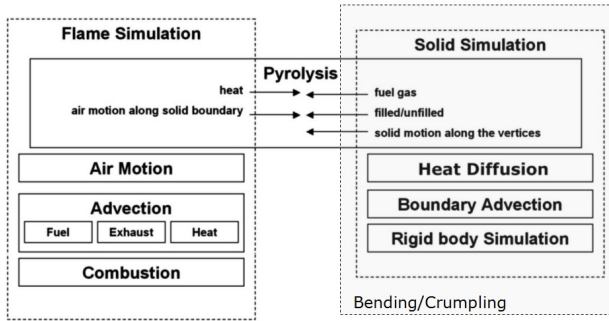

Figure 11: Extending the model to include bending/crumpling.

The two simulations are coupled together by: pyrolysis (transferring fuel from the solid representation to the fluid representation), heat interpolation (transferring heat information from the fluid simulator to the solid representation), and external forces (changes in air motion due to object movement and the reverse).

To incorporate the deformation into this framework, we add a "wrapper" function to the solid simulation. Solid objects are simulated within their own local space in the simulation framework described above. The FFD model is placed between the flame simulation and solid decomposition processes. The burning solid does not know that it is deformed, except during the synchronization with the flame module at the pyrolysis stage, when the simulator must follow an extra layer of indirection to map locations between the two modules. No special handling or extra code is required in any of the modules themselves. The proposed process is simple, easy to integrate to our existing simulation framework, and could also be used to model similar behavior from other secondary phenomena. Compared to the rest of the simulation, the deformation adds only a minor overhead. Most of the overhead does not come from the deformation process itself, rather the space deformation adds a fixed amount of overhead during the synchronization between flame and solid modules.

As discussed in the earlier section, the deformation framework does not handle self intersections. However, it is important to note that

the deformed object now exists in a different world position. Thus, any simulation data can be correctly placed into or interpolated from world coordinates; this introduces a bidirectional coupling between the transformation and the underlying simulation. For example, if a match bends into a U shape, the ends of the match will be near each other in world space, and thus heat that is transferred via a world-space (as opposed to local object space) calculation, as in our flame simulator, will easily allow heat transfer from one end of the match to the other.

By using the FFD approach presented in this paper, we can thus deform complex models without the additional simulation cost associated with modeling the actual physical process of the secondary deformation.

7 Conclusion

In this paper we present a simple deformation framework to approximate some physical process, thus giving increased realism to a simulation using only a limited computational overhead. Instead of modeling a rather complex and detailed process to create such secondary behavior, we propose the use of a free form deformation to model large-scale deformations. Changing object properties from the physical simulation modeling the "main event" are mapped onto the edges of a proxy object, which is then modified by treating the edges as springs. This proxy object then serves as a control structure for defining the deformation of the underlying object. We note here that this approach fits well in the multi-representation object modeling paradigm [Melek and Keyser 2005], where the proxy object can be seen as just another representation of the original object.

We have demonstrated that this approach can be used to create deformations that currently must be modeled by hand in simulation systems. We present results on bending burning matches as well as crumpling of burning paper that demonstrate the first such physically-based graphical simulation of these phenomena.

7.1 Limitations and Drawbacks

Though our model presented here is simple, there are some drawbacks that accompany this simplicity:

- We ignored topological changes during the mapping of physical properties onto the proxy object.

- Material compression during the bending is ignored.

- Self collisions are not handled.

- The proposed model is suitable for simple deformations and might be of more limited use for complex deformations.

- By definition, we are not simulating the "real" physics.

7.2 Advantages

Although we do not simulate the actual physical process, we can approximate some of the deformations easily using our proposed model. Our method thus also has several advantages:

- This framework has a minor computation overhead that is easily integrated into existing systems.

- The user has some control over the desired behavior. Though the deformation is still physically-based, we have not eliminated the opportunity for user input.

- The method is "based" on simulation data, and thus is driven by a physical process, but does not actually simulate the "real"

physics behind the process, which might be overkill for many purposes.

- Plausible results are easy to construct for the cases presented.

7.3 Future Work

There are still many avenues open to further work. Though we have presented two possible applications, one can easily imagine several other options for applications of this framework. These other applications will, of course, require different proxy objects, mapping of parameters, deformation descriptions, and possibly even applied deformations. Two examples are briefly mentioned here.

One such alternative deformation is the bending of heated metal objects. We can simulate the heat diffusion inside the object, and then map this onto the simplified proxy encapsulating the melting object. We can now define torques on the proxy cells by using the density of the object, and find high stress regions on the proxy to apply a bend deformation. Similarly, high stress regions could be used to fracture the object. Here, the proposed model would not perform fracturing, rather it will point to potential fracture areas. A number of more complex bending operations can also be imagined.

Another interesting possible use of this model is geared towards information visualization. When we deform the objects using physical simulation as presented in this paper, we deform them in the global space, and all the interactions with other objects are affected by this deformation. For scientific or information visualization purposes on the other hand, we can deform the object to highlight which part of the object has "interesting" data. Using the same framework described here, we can define a deformation-based visualization. We start by mapping the information on/in the object onto the proxy. Here, we can use only one kind of data, or a combination of data values we are interested in. We can grow parts of the object with "interesting" data and shrink the rest. This deformation would not affect the actual object or its interactions with the others, but it could help us to visualize and investigate the data and detect features in it more easily.

Acknowledgments

References

BÆRENTZEN, J. A., AND CHRISTENSEN, N. J. 2002. Volume sculpting using the level-set method. In *SMI '02: Proceedings of the Shape Modeling International 2002 (SMI'02)*, IEEE Computer Society, 175.

BÆRENTZEN, J. A. 1998. Octree-based volume sculpting. In *Proc. IEEE Visualization 98*, IEEE Computer Society Press, 9–12.

BARR, A. H. 1984. Global and local deformations of solid primitives. In *Proceedings of Siggraph'84*, 21–30.

BREMER, P. T., PORUMBESCU, S., KUESTER, F., HAMANN, B., JOY, K. I., AND MA, K.-L. 2002. Virtual clay modeling using adaptive distance fields. In *Proceedings of the 2002 International Conference on Imaging Science, Systems, and Technology (CISST 2002)*, Computer Science Research, Education, and Applications Press, H. R. Arambnia and et al. Eds., vol. 1.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIć, Z. 2002. A multiresolution framework for dynamic deformations. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, 41–47.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. Interactive skeleton-driven dynamic deformations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 586–593.

CARLSON, M., MUCHA, P. J., VANHORN, B., AND TURK, G. 2002. Melting and flowing. *ACM SIGGRAPH Symposium on Computer Animation*.

CHANCLOU, B., LUCIANI, A., AND HABIBI, A. 1996. Physical models of loose soils dynamically marked by a moving object. In *Computer Animation '96*, 27–35.

CHANG, Y., AND ROCKWOOD, A. P. 1994. A generalized de casteljau approach to 3d free-form deformation. In *Proceedings of Siggraph'94*, 257–260.

COQUILLART, S. 1990. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, ACM Press, 187–196.

DEWAELE, G., AND CANI, M.-P. 2003. Interactive global and local deformations for virtual clay. *Pacific Graphics* (october).

FEDKIW, R. 2002. Simulating natural phenomena for computer graphics. *Geometric Level Sets in Imaging, Vision and Graphics, edited by S. Osher and N. Paragios*.

FLOATER, M. S., KOS, G., AND REIMERS, M. 2005. Mean value coordinates in 3d. *Computer Aided Geometric Design 22*, 623–631.

FLOATER, M. S. 2003. Mean value coordinates. *Computer Aided Geometric Design 20*, 1, 19–27.

FRISKEN, S. F., AND PERRY, R. N. 2001. A computationally efficient framework for modeling soft body impact. *Technical Report 2001-TR2001-11*.

GALYEAN, T. A., AND HUGHES, J. F. 1991. Sculpting: an interactive volumetric modeling technique. *SIGGRAPH Comput. Graph. 25*, 4, 267–274.

GASCUEL, M.-P., VERROUST, A., AND PUECH, C. 1991. Animation and collisions between complex deformable bodies. *Graphics Interface '91* (june), 263–270.

GIBSON, S., AND MIRTICH, B. 1997. A survey of deformable modeling in computer graphics. Tech. Rep. TR-97-19, Mitsubishi Electric Research Lab., Cambridge, MA, November.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. In *Proceedings of ACM SIGGRAPH 2004*, ACM Press.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3, 561–566.

KOBAYASHI, K. G., AND OOTSUBO, K. 2003. t-ffd: free-form deformation by using triangular mesh. *SM'03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, 226–234.

LI, X., AND MOSHELL, J. M. 1993. Modeling soil: Realtime dynamic models for soil slippage and manipulation. In *SIGGRAPH '93 Conference Proceedings*, ACM SIGGRAPH, 361–368.

LLAMAS, I., KIM, B., GARGUS, J., ROSSIGNAC, J., AND SHAW, C. D. 2003. Twister: a space-warp operator for the two-handed editing of 3d shapes. *ACM Trans. Graph. 22*, 3, 663–668.

LOSASSO, F., IRVING, G., GUENDELMAN, E., AND FEDKIW, R. 2006. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics 12*, 3, 343–352.

LUNDIN, D. 1984. Motion simulation. Nicograph '84.

LUNDIN, D. 1994. Works' ant. In *Special Issue: Fifteen Years of Computer Graphics 1979-1994*, ACM SIGGRAPH.

MACCRACKEN, R., AND JOY, K. I. 1996. Free-form deformations with lattices of arbitrary topology. *SIGGRAPH'96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 181–188.

MCDONNELL, K. T., AND QIN, H. 2002. Virtual clay: Haptics-based deformable solids of arbitrary topology. In *AMDO '02: Proceedings of the Second International Workshop on Articulated Motion and Deformable Objects*, Springer-Verlag, 1–20.

MELEK, Z., AND KEYSER, J. 2003. Interactive simulation of burning objects. *Proc. of Pacific Graphics '03*, 462–466.

MELEK, Z., AND KEYSER, J. 2005. Multi-representation interaction for physically based modeling. ACM Symposium on Solid and Physical Modeling.

NING, P., AND BLOOMENTHAL, J. 1993. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, November 33–41.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. *Proc. of ACM SIGGRAPH '02*, 291–294.

ONOUE, K., AND NISHITA, T. 2003. Virtual sandbox. In *Proc. of Pacific Graphics '03*, 252–259.

OSHER, S., AND FEDKIW, R. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag.

PARRY, S. R. 1986. Free-form deformations in a constructive solid geometry modeling system. In *PhD Thesis*, Brigham Young University.

PAULY, M., PAI, D. K., AND GUIBAS, L. J. 2004. Quasi–rigid objects in contact. *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 109–119.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Proceedings of Siggraph'86*, 151–159.

SETHIAN, J. A. 1999. *Level Set Methods and Fast Marching Methods*. Cambridge University Press.

SUMNER, R. W., O'BRIEN, J. F., AND HODGINS, J. K. 1998. Animating sand, mud, and snow. In *The Proceedings of Graphics Interface '98*, 125–132.

TERZOPOULOS, D., PLATT, J., AND FLEISCHER, K. 1989. Heating and melting deformable models (from goop to glop). *Proc. Graphics Interface '89* (June), 219–226.

TERZOPOULOS, D., PLATT, J., AND FLEISCHER, K. 1991. Heating and melting deformable models. *The Journal of Visualization and Computer Animation 2*, 2, 68–73.

VON FUNCK, W., THEISEL, H., AND SEIDEL, H.-P. 2006. Vector field based shape deformations. *ACM Trans. Graph. 25*, 3, 1118–1125.

WANG, S. W., AND KAUFMAN, A. E. 1995. Volume sculpting. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, ACM Press, 151–156.

WARREN, J., SCHAEFER, S., HIRANI, A., AND DESBRUN, M. 2004. Barycentric coordinates for convex sets. Tech. rep., Rice University.

WARREN, J. 1996. Barycentric coordinates for convex polytopes. *Advances in Computational Mathematics*, 6, 97–108.

YNGVE, G. D., O'BRIEN, J. F., AND HODGINS, J. K. 2000. Animating explosions. *Proc. of ACM SIGGRAPH '00*, 29–36.