

Multi-Representation Interaction For Physically Based Modeling *

Zeki Melek
Department of Computer Science
Texas A&M University
College Station, Texas, USA
melekzek@neo.tamu.edu

John Keyser
Department of Computer Science
Texas A&M University
College Station, Texas, USA
keyser@cs.tamu.edu

ABSTRACT

For simulations involving complex objects, a number of different properties must be represented. An example of this is in modeling an object undergoing combustion—heat amounts, fuel consumption, and even object shape must be modeled and changed over time. Ideally we would put everything into a unified representation, but this is sometimes not possible/feasible due to measurement limitations or the suitability of a specific representation. In this paper we define a multi-representation framework for dealing with multiple properties and their interactions within an object. This model is especially useful in physically based modeling, where the time variation of some properties affect other properties including geometry or topology. As a motivating example case, we present a method for modeling decomposition of a burning object.

Categories and Subject Descriptors

I.3.5 [Computer Graphics]: Physically Based Modeling

General Terms

multi-representation

Keywords

physically based modeling, simulation framework

1. INTRODUCTION

1.1 Overview

For objects participating in complex physically-based simulations, we may need to keep track of several different object properties. In some simulations these properties can change, and changes in one property may affect some other property of the object. For example, temperature in different parts of an object may affect the conductivity of parts of an object.

*put copyright information here

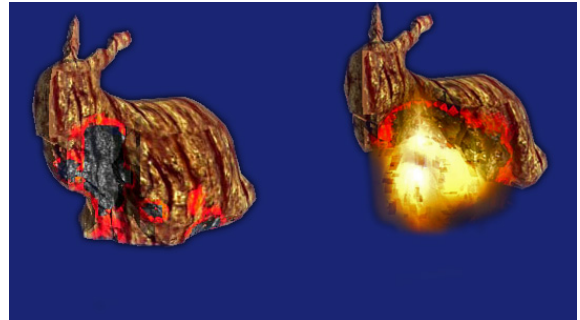


Figure 1: Burning Bunnies (OpenGL output from the interactive simulation)

A particular motivating example for us is trying to model the decomposition of an object as it undergoes combustion. Heat is stored (and varies) throughout the object, reactions release fuel from a solid to a gaseous state, and the geometry and topology of the object change over time. No prior methods have captured these properties within a simple simulation framework, as our method does.

It often seems that an ideal solution would be to store all such object properties in a single representation. For example, an object might be broken up into hexahedral finite elements, and all property information stored at the vertices or cell interiors of this one representation. Traditionally, material properties are modeled procedurally [17] or volumetrically. While such a single-representation modeling approach may be effective for many applications, it also has various drawbacks:

- Different properties might not need to be stored at the same scale. For example, in a particular application, we might need to know density information at a finer resolution than temperature information.
- Different representations might yield computational efficiencies. For example, distance field implicit representations can be very fast for collision detection [6], while applying impact forces is easier with a boundary representation such as a polygonal surface model.
- There may be limitations on the measurement capabilities for certain properties. For example, a real-world



Figure 2: Burning log

object might have a high-resolution surface scan as well as a lower-resolution volumetric measurement.

- Different properties might have a natural representation in a different type of representation, e.g. a procedural definition versus a volumetric grid.
- We might want different properties to diverge significantly. For example, we might want two grid representations to change over time for different properties, or define one property over only a small portion of an overall object.

It is usually possible to convert from one representation to another. However, within a simulation framework, it is not usually feasible to do so at every time step, and such conversions often tend to lose accuracy through repeated conversion. An alternative approach is to keep multiple representations of the same object and use the appropriate one in different parts of the simulation. Examples of this are keeping both a level set and boundary representation of an object during collision simulations [6], or maintaining a set of isosurfaces for a volumetric model. In these approaches, however, the different representations are actually referring to the same object and do not contain additional information about the object. Furthermore, such approaches usually assume a “ground truth” to which all other properties can be referenced, and any modification in one representation must be propagated to all of the others.

In this paper we define a model for supporting interactions between different properties of an object kept in different representations, as well as functions based on this interaction. Note the difference to the previous methods: we are proposing using different representations to define different properties, every representation is only a partial definition of the object, and all representations together define the object in the simulation. This model is especially useful in physically based modeling, where the time variation of some properties affect other properties including geometry or topology. While we assume the general idea is not novel, and similar approaches can likely be found in various implementations, we believe this is the first formal attempt to put this framework into a more generalized structure.

1.2 Motivating Problem

The work presented in this paper focuses specifically on the problem of modeling the decomposition of a burning object,

such as a piece of wood, in an interactive simulation. This problem has been the main motivation behind our work. There are several challenging aspects in modeling such an object, and we know of no prior work addressing these issues. Among the key properties we have sought to model are:

- The object is a store of fuel for a combustion reaction (modeled separately). We must be able to maintain a measure of this fuel, and control its release.
- The object will have differing temperatures at different points in the model. This temperature is both affected by external factors, and must affect the release of fuel from the solid.
- As the object burns, its geometry and topology can change significantly. As the object burns up, it must leave some ash residue behind.
- The object must be able to interact with the external world as a solid, e.g. interacting via collisions, or affecting air flow.
- For interactive rendering, a surface model is needed.
- While our current implementation does not implement this, the object will be subject to internal stresses and strains as it changes shape.

Trying to provide all of this in a single representations is not feasible. Instead, we define a multi-representation model that allows us to model these properties and their interactions, and that can be run within an interactive simulation.

1.3 Main Results

There are two main results we present in this paper.

First, we present the first known method for modeling the decomposition of a burning object within an interactive simulation. We take into account a variety of physical properties, including topological changes, ash production, heat distribution, and fuel release.

Second, we generalize the multi-representation approach used for our burning object model to describe a framework for representing other models within physically-based simulations. This generalization makes a separation between object properties and their representations, and organizes the

simulation structure around the properties. We describe the nature of such a representation, and the methods needed to supply interactions between the multiple representations.

1.4 Overview of Paper

In section 2 we describe some related and background work. In section 3 we describe our general multi-representation framework. Section 4 describes our approach to modeling a burning object. Section 5 concludes with a discussion of how this approach can be applied to other problems and the key features and drawbacks to the work we have presented.

2. BACKGROUND

Although the proposed framework is/might be used in implementations, this is the first attempt to formalize it in a generalized form. Similar uses could be found inside the implementation details in various papers.

While recent fire models [9, 12, 5] are promising, none of them yet model decomposition of the burning solids and solids are rather treated only as a fuel source. Some recent visualization research has focused on tracking the motion of the flame front, yet it assumes the underlying geometry is fixed [16, 19, 1, 20]. There has not been any previous model to address decomposition of solids under combustion process. Some recent work on modeling flames include Nguyen et al.[12], defining the flame front as a moving boundary between two fluids and using a level set method to capture very complex motion of the fire. Yet they only model burning of solid objects as an on/off mode, and do not address the decomposition at all. Fire safety related research generally deals with smoke behavior and heat distribution, such as [8, 9] and do not model solid burning. The approach of Wei et al. [21] is similar to ours in defining solids as a volumetric implicit field, yet their fuel consumption is simple, and they do not model deformation under decomposition.

Although not directly related to decomposing objects, there is some research on modeling deformation of solids under physically based simulations. Carlson et al.[2] solve the melting problem by treating the solid as a liquid with high viscosity. Dorsey et al.[3] deal with the effects of erosion and mineral dissolution.

3. MULTI-REPRESENTATION

A multi-representation framework basically puts each different property into whatever representation(s) best suits it. Note that this does not require that we put all properties into its own space; two or more properties could make use of the same representation, as with multiple property values being defined over the same grid. Within a simulation, the properties are accessed and modified, and some of these changes can affect the representation(s) used for that property. Changes to a representation shared with other properties thus propagate certain global effects. This framework allows us to separate the properties from their representations, and separate the main simulation modification and interaction among the properties. Doing this requires shifting the simulation design from the representation to the property itself.

As mentioned before, we can use one representation space

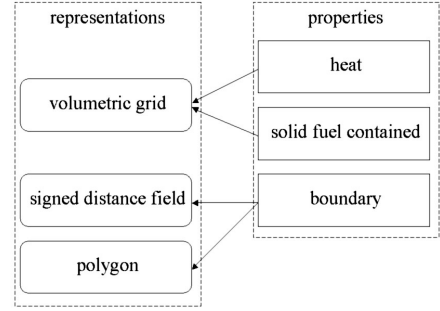


Figure 3: Separation of properties from the representation

for many properties. We can also use multiple representations for the same property as long as we ensure that all representations are kept in synch. The motivation for use multiple representations for the same object is that for different processes in the simulation, a different representation may be more suitable. When the cost of choosing a single representation is larger than the cost of having multiple and synching them, we use multiple representations. An example of such a dual representation in use is in Guendelman et al.’s work [6], where they used a dual representation (one implicit and one boundary) for a rigid object.

Also note that in some cases we might want to manipulate the representation itself. Simulation level-of-detail (SLOD) applications, where a range of simulation accuracies can be obtained, are one such example. If we have only a single representation for all the properties, using a finer scale representation desired by one property would create a large and unnecessary overhead for the other properties. Likewise, a coarse scale favored for efficiency by some parts of the simulation may lose accuracy needed in other parts.

We organize the simulation structure around the properties and their manipulations. This abstraction from the representation itself enables us to choose suitable representations for any property based on the property, manipulation function, and/or cost of the manipulation. Properties can share the the same representation, use an individual representation, or span multiple representations. In figure 3 we have two extreme cases. One representation (volume) containing two properties (heat and solid fuel) and one property with two representations (implicit and polygonal).

3.1 Framework

We consider our proposed framework a bit closer. Since our focus is on organizing simulations around object properties, we structure the simulation accordingly. Instead of using a monolithic simulation time step, we propose a two step simulation by grouping property manipulation functions (PMF) into two groups. We also can group the properties themselves based on which PMFs manipulate them.

3.1.1 Property Manipulation Functions

- *External* PMFs account for actions taking place outside/around the object during the simulation and which

affect the object. Some examples of these are external heat, collisions, wind forces, magnetic fields, etc. Note that external PMFs do not necessarily change the object—they can model how the object affects others.

- *Internal PMFs* account for changes of some properties within the object itself; these can be (but do not have to be) secondary effects of external PMFs. Internal PMFs can be defined as chains of functions where changes in the values of one property of the object triggers changes in other properties (and are thus called *property interaction functions*) or other values of the same property. Examples of internal PMFs include heat propagation within a solid, force propagation through an object, and changes in object composition due to chemical or physical reaction.

In the first of the two steps, the simulation body accesses some object properties and modifies them using external PMFs. In the second simulation step, we apply internal PMFs, in some predefined order. Here, one should note that in most simulations we only need to define a subset of possible interactions among only a few subsets of properties; it is not necessary to define every possible property interaction).

3.1.2 Property Classes

We now classify object properties themselves into two groups based on what group of manipulation functions effect them.

- *Public* Public properties are changed **directly** from external forces/effects during the simulation. They are accessed using external PMFs.
- *Private* Private properties are affected **only** from the changes of other properties. They are manipulated using internal PMFs.

Note that although we used OOP naming conventions, they apply only to how the property is manipulated. Private properties are **not** necessarily internal structures; they can be read from outside, but not manipulated directly. Public properties could be changed by private properties as well.

Grouping the properties into two categories does not mean that their representations need to be separate. This grouping of properties and functions is intended to organize the simulation. Properties from different groups can share the same representation, but be changed in different passes of the simulation.

3.2 Requirements

When using a multi-representation, some properties should be ensured, in order to ensure that the object, which is stored as a collection of properties spread over multiple representations, is always a consistent and valid object.

- *Local vs Global Changes:* The effect of local changes in some property on the others is modeled as an internal PMF. External local effects are modeled as external

PMFs. On the other hand, global changes affecting the object as a whole need to be addressed separately, such that all representations stay in registration with each other to ensure the validity of the object. The most obvious example of this is rigid body motion. If a simulation will allow rigid body motion, then all representations for the object must refer to a modified local coordinate system.

- *Interpolation:* Since we heavily depend on data exchange between separate representations, all properties (and thus all representations) should support an interpolation interrogation, to get the value of any property at any point inside or on the boundary of the object.
- *Validity:* If we allow changes in the representations themselves (e.g. a change in a boundary mesh), we need to ensure that the overall model remains valid. As will be seen in our example in section 4, it can be useful to allow multiple representations to diverge, but we must do so very carefully, and ensure that we have limited any use of the object in such a way that this divergence is not a problem.
- *Synchronized Behavior:* When one property shares more than one representation, both representations are referring to the same thing, and thus we must ensure that the two representations are synchronized throughout the simulation.
- *Function Ordering:* The simulation accesses and changes properties using external PMFs, and the changes affect other properties through internal PMFs. For time step simulations an execution order inside the time step should be defined in order to prevent cyclic calls. If cyclic calls are desired, the ending conditions or oscillating states should be thoroughly checked to ensure simulation stability.

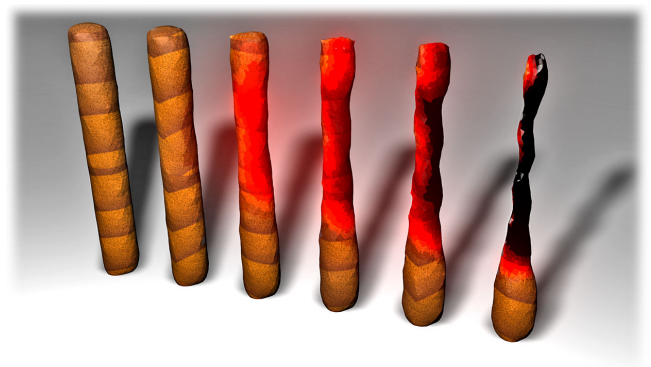


Figure 4: Burning match

4. MODELING DECOMPOSITION

We have applied our multi-representation framework in order to simulate the decomposition of objects under combustion. This decomposition is a complex process, involving many different interacting properties, and so is ideal for our framework. The simulation of object decomposition under

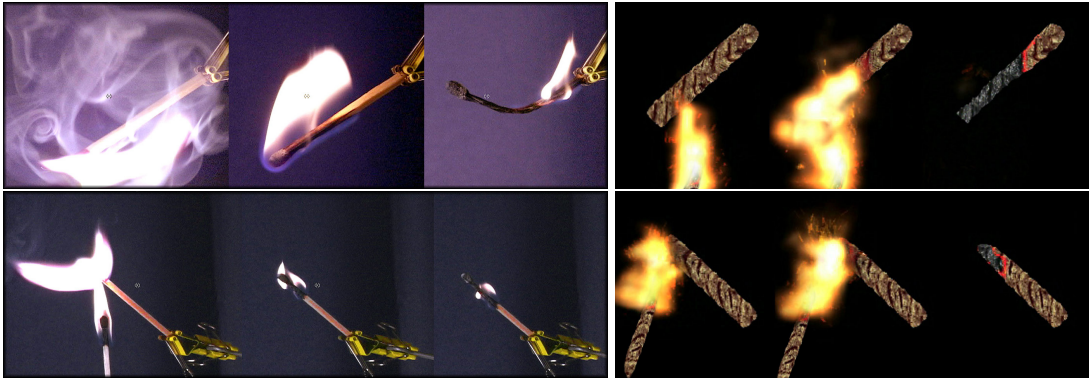


Figure 5: Real vs simulated matches in different orientations

fire has not been addressed previously, and our method captures many of the needed aspects of that process. We present here a detailed description of how we implement decomposition within the multi-representation framework, including experimental results.

A separate simulation [10] models the flame production and generates heat from a combustion process, which is used to trigger pyrolysis of the solid object (For a recent, similar approach to modeling chemical reactions refer to [7], and for the fluid solver refer to [18, 4]). The solid must participate within the simulation, both affecting the flame simulator (by supplying fuel, and affecting air flow), and being affected by the simulation (heat triggering the release of fuel, and the environment providing external forces). Even with a very simple flame model on a coarse grid, we can achieve a plausible decomposition of the burning object [11].

4.1 Properties and Representations

Every solid to be burned has three main properties, object boundary (surface), heat distribution inside the object, and solid fuel contained. For a more complex model additional properties could be added to model such things as variations of density, variation in heat conductivity, and internal stresses. We store three main properties in three separate representations:

The first representation is an implicit form, a finite element signed distance field. It represents the object boundary at zero crossing, which will be used to model the surface of the decomposing object. The resolution depends on how detailed we want our initial/intermediate object boundaries to be. We will generate intermediate decomposition states using this representation. Visualization of the decomposed object is done by generating another representation, a polygonal representation of the object, temporarily.

The second representation is a volumetric representation, which is the amount of solid fuel stored in the cells. This representation usually does not require as fine a grid resolution as for the distance field. The resolution of the boundary and fuel representations can be adjusted for simulation behavior (either representation) and visual quality (distance field representation). Although the two representations are equivalent at the start of the simulation, being generated from the same initial boundary representation, we inten-

tionally allow them to diverge during the simulation. The different behavior of the two representations during the decomposition simulation allows us to consume solid fuel but leave additional material behind in the form of ash residue.

The third representation is similar to the previous one—a volumetric representation is used to model the heat distribution inside the object, and used to trigger the pyrolysis reaction. Pyrolysis is the process by which a solid emits combustible gases. Once a solid cell reaches the pyrolysis temperature T_{sp} , a pyrolysis process is applied at every simulation time step. This temperature can be set low for volatile solids, arbitrarily high for nonvolatile solids, or vary through the solid to model mixed solids. During the pyrolysis step, some of the solid fuel is converted into fuel gas as a function of the temperature. The boundary representation is updated based on the change in solid fuel amount. Here, note that since the last two representations are similar, they can be put into one representation tracking two properties at the same time. Figure 3 shows an example of this change.

4.2 Simulation Structure

The simulation structure is as follows.

- First, the heat exchange to/from the outside is modeled by an external PMF on the heat property.
- Second, internal heat diffusion is modeled as an internal PMF on the heat property itself.
- The new heat distribution is used to trigger pyrolysis from the volume representation of the solid fuel amount property and the released fuel gas is output to the fire simulation. The change in the solid fuel amount is modeled as an internal PMF.
- The release of fuel from the solid also has a secondary effect, in modifying the boundary property representation. This is how the decomposition of the object is achieved.
- For visualization, a temporary polygonal representation is created; this can be thought of as an internal PMF on the boundary property.
- The items must be able to undergo rigid body motion (a global property) within the simulation. Coupled

with this, we need to perform collision detection on the objects.

- As objects burn, their topology can change, eventually causing them to split into two or more pieces. We must detect these changes and modify the object topology.

4.2.1 Heat Transfer

We model heat transfer within the simulation framework in three stages: heat transfer in the air, heat transfer between the air and the solid, and heat conduction within solids. This three-stage heat transfer model enables us to treat solids with varying thermal properties.

For the heat transfer between the solid and the air, we first find each solid-air boundary voxel in the distance field representation. We initialize the heat in the neighboring unfilled cells by from the heat info from the air. For each of boundary cell, we exchange heat to/from the adjacent unfilled “air” voxels, using the heat differential between the filled cell and the unfilled neighbors. The change of heat in the boundary air cells is interpolated back to the heat information in the air.

4.2.2 Heat Diffusion

Heat transfer inside solids is modeled as a diffusion process using implicit integration. In this way, the heat transferred from outside the object as above spreads into the interior of the object being burned.

$$\frac{d}{dt}T = k\nabla^2T \quad (1)$$

, where k is a constant based on density, thermal conductivity, and specific heat of the material. For most objects being burned, this heat diffusion is quite slow (k is small), and constant through the object. For nonuniform material, we can incorporate variations of thermal conductivity in our simulation by adding one more property, *thermalconductivity*, as a volumetric grid.

4.2.3 Pyrolysis

Once the cell reaches the pyrolysis temperature, a pyrolysis process is applied at every simulation time step. This temperature can be set low for volatile solids, arbitrarily high for nonvolatile solids, or vary throughout the solid (if we add another property) to model mixed solids. Note that this information could also be generated from a texture map to allow variations across the object. During the pyrolysis step, some of the solid fuel is converted into fuel gas.

When $T > T_{pyrolysis}$

$$\frac{\partial}{\partial t}d_g = r_s \frac{\partial}{\partial t}V \quad (2)$$

where V is the solid fuel amount, d_g is the density of fuel gas, and r_s is the conversion rate from solid fuel to fuel gas. We have an in-depth analysis of the amount of solid fuel

converted in the next section. Note that the decomposition model is integrated into our interactive flame simulation framework, enabling the user to tweak the parameters for the desired simulation behavior easily.

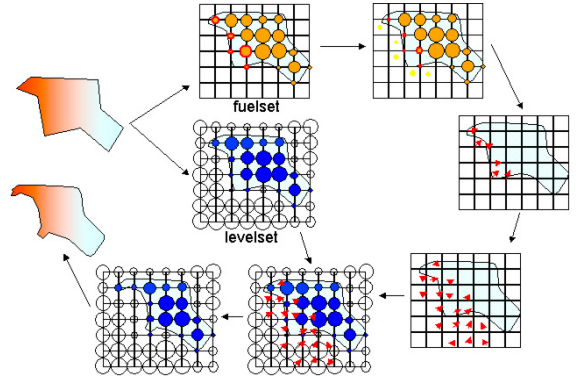


Figure 6: Decomposition steps

4.2.4 Decomposition

The decomposition of the burning solid is modeled as a moving boundary in the distance field representation of the solid. Level set methods, which we use to model decomposition are a simple yet powerful approach for computing moving interfaces [15, 14]. The decomposing solid is defined implicitly as

$$\phi(\vec{x}, t) = 0 \quad (3)$$

We define the decomposition of the object with two components: d the amount of decomposition, and \vec{D} the direction of decomposition.

$$\frac{\partial}{\partial t}\phi = d(\vec{x})\vec{D}(x) \quad (4)$$

Defining d , the amount of decomposition, is easy. We define it proportional to the amount of solid fuel released as fuel gas in a time step.

$$d(\vec{x}) = k_1 \frac{\partial}{\partial t}V(\vec{x}) \quad (5)$$

$$0 \leq k_1 \leq 1 \quad (6)$$

where V is the solid fuel representation and k_1 is a constant controlling the strength of decomposition. Note that k_1 represents the physical quantity of the ratio of residue (nonflammable material) vs the solid fuel (flammable material) and thus controls the amount of ash left.

Defining \vec{D} , the direction, is dependent on how we define the pyrolysis process. We have implemented two versions, one

with a constant pyrolysis process and one with time varying pyrolysis. They are covered in the next section.

4.2.5 Polygonalization for Visualization

The distance field representation is polygonalized by an iso-surface generation using tetrahedral decomposition of the boundary cells [13]. This method generates a polygon soup, and additional pointers are used to keep track of which simulation cell the generated triangles belong to, and merge the corresponding ones. After each timestep, previous visualization polygons are discarded and new ones are created. This working scheme simplifies the implementation, yet the lack of frame-by-frame tracking prevents consistent texture coordinate inheritance during the interactive simulation. Instead, we used projection methods to map texture coordinates from the implicit form onto the visualization polygons.

4.2.6 Rigid Body Motion

Collisions are determined by comparing objects pairwise. For one object, we use a set of particles placed at the vertices of the visualization polygons. After a global-to-local transformation of the vectors, interpolation on the distance field grid of the other object directly gives us the approximate distance to the boundary, letting us know whether the objects have collided (and if so, where) [6]. Note that the particles live only for one frame/timestep of the simulation, since the visualization polygon is discarded and recreated in the next timestep, keeping only the position, velocity, and angular velocity of the solid piece from frame to frame. Standard collision response approaches then model response of the pieces. Note that rigid body motion is a global change affecting all representations of the object.

4.2.7 Disconnected Pieces

Pieces becoming disconnected in the burning process should be detected. The polygons created from the implicit representation are used to detect such separations. Here, note that we need to have a complete solid object representation (including topological connections), compared to the faster polygon-soup algorithms. Once the separate pieces are detected within the given solid, we split the volume and distance fields accordingly, creating two or more separate solid objects. This structure allows us to simulate the motion of individual pieces, while each still burns and decomposes inside its local computational domain. Finding the changed center of mass and estimating the moment of inertia is straightforward once each object has been separated.

4.3 Defining Decomposition Direction

As stated earlier, we have implemented two versions, one with a constant pyrolysis process and one with time varying pyrolysis.

4.3.1 Constant Rate Pyrolysis

Our first implementation is based on a constant fuel gas release model. During simulation, any object cell passing the self-pyrolysis threshold converts a fixed amount of solid fuel into gaseous fuel until it consumes all of its solid fuel. It is a simple on/off switch, continuing until it runs out of fuel (or until the temperature drops low enough, which is unlikely in realistic simulations).



Figure 7: On the left is constant pyrolysis decomposition and on the right is time-varying pyrolysis decomposition.

$$\frac{\partial}{\partial t} V(\vec{x}) = dt \begin{cases} k_2, & T(\vec{x}) \geq T_{sp} \text{ and } V(\vec{x}) \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

$$-1 \leq k_2 \leq 0 \quad (8)$$

where $T(\vec{x})$ is the temperature at \vec{x} . k_2 controls how fast fuel gas is released. Together with the self-pyrolysis temperature threshold (T_{sp}), it controls how volatile the solid will be. Since the fuel gas release is constant from burning cells, tracking the flame spread direction is not possible. For the direction, then, we used for our decomposition direction the normal from the implicit model, which is a “sure” way to ensure the boundary continually moves inward.

$$\vec{D}(\vec{x}) = -\vec{N}(\vec{x}) = -\frac{\nabla\phi(\vec{x})}{|\nabla\phi(\vec{x})|} \quad (9)$$

Putting equations 5 and 9 together, the decomposition is defined as

$$\frac{\partial\phi}{\partial t} = -k_1 \frac{\partial}{\partial t} V(\vec{x}) \frac{\nabla\phi(\vec{x})}{|\nabla\phi(\vec{x})|} \quad (10)$$

4.3.2 Time-varying Pyrolysis

Instead of having a constant release rate, we can define the amount of fuel released as being proportional to the solid fuel left. There are then multiple ways of defining the pyrolysis, one of them being

$$\frac{\partial}{\partial t} V(\vec{x}, t) = dt \begin{cases} k_3 \left(1 - \frac{V(\vec{x}, t)}{V(\vec{x}, t=0)}\right), & T(\vec{x}) \geq T_{sp} \\ & \text{and } V(\vec{x}, t) \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

We choose a monotonically decreasing function, since it will give us the benefit of determining the direction of the moving

flame front. Cells newly entering into the pyrolysis stage will have a higher output than the cells at a more advanced in the pyrolysis. By examining these differences, we can define the direction of flame propagation.

$$\vec{D}(\vec{x}) = \frac{\nabla \frac{\partial}{\partial t} V(\vec{x})}{|\nabla \frac{\partial}{\partial t} V(\vec{x})|} \quad (12)$$

Putting equations 5 and 12 together, the decomposition is defined as

$$\frac{\partial \phi}{\partial t} = -k_1 \frac{\partial}{\partial t} V(\vec{x}) \frac{\nabla \frac{\partial}{\partial t} V(\vec{x})}{|\nabla \frac{\partial}{\partial t} V(\vec{x})|} \quad (13)$$

Note that equation 13 is valid for a wide range of pyrolysis definitions. We can adjust the release rate over time to be any monotonically decreasing function, and the equation 13 holds.

4.3.3 Comparison

As seen in figure 7 both methods of decomposition give similar results. This is to be expected, since the amount of decomposition applied is the same in both cases. Even in the closeup view in figure 8 the results are almost the same. The only differences are that the direction of decomposition changes slightly, and that the normal possibly gives a nicer approximation. Note that the color change rendered is based on how long a given cell has been in pyrolysis.

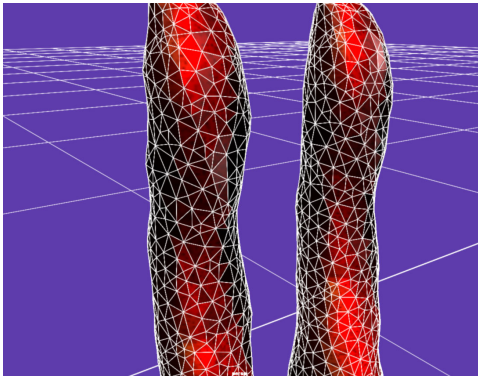


Figure 8: Closeup of decompositions. Constant-rate pyrolysis is at left, time-varying at right.

5. CONCLUSION

We have proposed a framework for organizing a complex physically based simulation, where we focus on object properties in structuring the simulation rather than on representations. We have presented in detail the first known method for our driving problem, modeling the decomposition of a burning object. It takes into account a variety of physical properties, including topological changes, ash production, heat distribution, and fuel release, all within the multi-representation framework.

We have generalized the multi-representation approach used for our burning object model to describe a framework for representing other models within physically-based simulations. This generalization creates a separation between object properties and their representations, and organizes the simulation structure around those properties. This approach simplifies the design of complex physically based simulations. We describe the nature of such a representation, and address the methods needed to supply interactions between the multiple representations.

5.1 Other Applications

We have already shown how our framework can be used for our motivating example of decomposing objects. The multi-representation framework is also suitable for several other complex physically based simulations. Several of these have been alluded to throughout the paper. We will briefly outline a couple of other such simulations. These examples are presented strictly as illustrations of the concepts and to show generality, not as detailed models of a simulation structures as was done for our motivating case. For obvious reasons we will not go into the detail used for our motivating case. A summary of the sample cases can be found in table 1.

Some of the other possibilities are:

- *Simulation of fracturing of a bone.* Properties to be simulated include the bone shape, density, and internal stresses. Representations used could include surface models, statistical models [17], and MRI-generated volumetric grids.
- *Simulation of an aquifer.* Separate representations might be used for the model of the water content, the permeability of the soil, other soil characteristics, and pressure. If the simulation were to include ground changes due to water depletion (e.g. sinkhole formation), geometric and topological changes might take place.
- *Simulation of a freezing water environment.* Properties such as heat, water flow, and phase would need to be kept track of, probably in multiple representations (e.g. a fluid representation would probably not be sufficient for the modeling of ice crystal formation).
- *Simulating a manufacturing process of a deformable or changing material.* We would want to be able to represent the (changing) geometry, internal forces, and likely properties such as heat or chemical reaction.

Acknowledgments

This work is supported in part by NSF grant CCR-0220047.

6. REFERENCES

- [1] P. Beaudin, S. Parquet, and P. Poulin. Realistic and controllable fire simulation. *Proc. of Graphics Interface '01*, pages 159–166, 2001.
- [2] M. Carlson, P. J. Mucha, B. VanHorn, and G. Turk. Melting and flowing. *ACM SIGGRAPH Symposium on Computer Animation*, 2002.

	Property	Representation	PMFs
Decomposition	Heat Solid Fuel Left Object Boundary	Volume grid Volume grid Signed distance and polygon	<ul style="list-style-type: none"> • Heat Exchange Combustion process generates heat • Pyrolysis Solid fuel is converted into fuel gas for the cells above self-pyrolysis threshold ◦ Level Set Method Moving the boundary using change in solid fuel ◦ Polygonization from boundary ◦ Heat diffusion inside the object
Fracturing Bone	Bone geometry Porous material Stress Strain Fracture	Polygonal model Statistical model Volume grid Volume grid Tetrahedral decomposition	<ul style="list-style-type: none"> • External force A set of forces are applied on a test point on the bone ◦ Shock propagation ◦ Fracture ◦ Increased stress due to fractures
Aquifer	Water content Water flow Permeability Pressure Heat	Volume grid Vector field Statistical model Volume grid Volume grid	<ul style="list-style-type: none"> • Water above the ground affects pressure ◦ Heat affects permeability ◦ Pressure, heat and permeability affects flow
Freezing Water	Heat Water flow Phase Crystal geometry	Volume grid Vector field Boundary surface as level set Polygon	<ul style="list-style-type: none"> • Water flow is affected by the surface • Exchange heat with external ◦ Flow affects crystal formation ◦ Crystal formation affects flow
Deformable Manufacturing	Stress Strain Heat Chemicals Geometry	Volume grid Volume grid Volume grid Volume grid Implicit and polygon	<ul style="list-style-type: none"> • External force changing geometry • Heat exchange ◦ Stress ◦ Heat drives chemical reaction changing chemical distribution and changing heat (endothermic vs exothermic)

Table 1: Examples of uses of the multi-representation framework for various simulations. The top row describes our decomposing burning object example. The following rows describe models that could be used for bone fracture, an aquifer, freezing water, or deformable material manufacturing simulations. ‘•’ denotes external and ‘◦’ denotes internal PMFs.

- [3] J. Dorsey, A. Edelman, H. W. Jensen, J. Legakis, and H. K. Pedersen. Modeling and rendering of weathered stone. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 225–234. ACM Press/Addison-Wesley Publishing Co., 1999.
- [4] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. *Proc. of ACM SIGGRAPH '01*, pages 15–22, 2001.
- [5] B. E. Feldman, J. F. O’Brien, and O. Arikian. Animating suspended particle explosions. In *Proc. of ACM SIGGRAPH '03*, pages 708–715, aug 2003.
- [6] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, 2003.
- [7] I. Ihm, B. Kang, and D. Cha. Proceedings of the 2004 acm siggraph / eurographics symposium on computer animation (sca-04). pages 203–212, 2004.
- [8] W. W. Jones, G. P. Forney, R. D. Peacock, and P. A. Reneke. A technical reference for cfast: An engineering tool for estimating fire and smoke transport. 2000.
- [9] K. B. McGrattan, H. R. Baum, R. G. Rehm, A. Hamins, G. P. Forney, J. E. Floyd, S. Hostikka, and K. Prasad. Fire dynamics simulator technical reference guide. *Tech. Rep. NISTIR 6783*, 2002.
- [10] Z. Melek and J. Keyser. Interactive simulation of fire. *Proc. of Pacific Graphics '02*, pages 431–432, 2002.
- [11] Z. Melek and J. Keyser. Interactive simulation of burning objects. *Proc. of Pacific Graphics '03*, pages 462–466, 2003.
- [12] D. Nguyen, R. Fedkiw, and H. W. Jensen. Physically based modeling and animation of fire. *Proc. of ACM SIGGRAPH '02*, pages 721–728, 2002.
- [13] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, pages November 33–41, 1993.
- [14] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag, 2002.
- [15] S. Osher and J. A. Sethian. Front propagating with curvature dependent speed: Algorithms based on

hamilton-jacobi formulations. *Journal of Computational Physics*, (79):12–49, 1988.

- [16] C. H. Perry and R. W. Picard. Synthesizing flames and their spreading. *Proc. of Fifth Eurographics Workshop on Animation and Simulation*, pages 105–117, 1994.
- [17] C. Schroeder, W. C. Regli, A. Shokoufandeh, and W. Sun. Representation of porous artifacts for bio-medical applications. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 254–257. ACM Press, 2003.
- [18] J. Stam. Stable fluids. *Proc. of ACM SIGGRAPH '99*, pages 121–128, 1999.
- [19] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion process. *Proc. of ACM SIGGRAPH '95*, pages 129–136, 1995.
- [20] X. Wei, W. Li, K. Mueller, and A. Kaufman. Simulating fire with texture splats. In *IEEE Visualization '02*, pages 227–235. IEEE Computer Society, 2002.
- [21] Z. F. A. K. Ye Zhao, Xiaoming Wei and H. Qin. Voxels on fire. *IEEE Visualization*, 2003.