# Real-time Rendering of Melting Objects in Video Games

Dhanyu Amarasinghe and Ian Parberry

Technical Report LARC-2013-01

Laboratory for Recreational Computing
Department of Computer Science & Engineering
University of North Texas
Denton, Texas, USA

March, 2013

# Real-time Rendering of Melting Objects in Video Games

Dhanyu Amarasinghe
Dept. of Computer Science & Engineering
University of North Texas
DhanyuAmarasinghe@unt.edu

Ian Parberry
Dept. of Computer Science & Engineering
University of North Texas
ian@unt.edu

## Abstract

We present a method for simulating the melting and flowing of material in burning objects fast enough to be of use in video games where most of the graphical and computational resources are needed elsewhere. The standard practice of using particle engines or fluid dynamics for melting are far too costly for use in this environment. We demonstrate that our method, which is based on systematic polygonal expanding and folding, uses only a fraction of the computational power available by implementing the computation on a very modest GPU using CUDA.

***Keywords*** Hardware acceleration, freeform deformation, polygonal modeling, CUDA, melting objects.

***General Terms*** Video games, visualization, algorithms, performance, design and modeling.

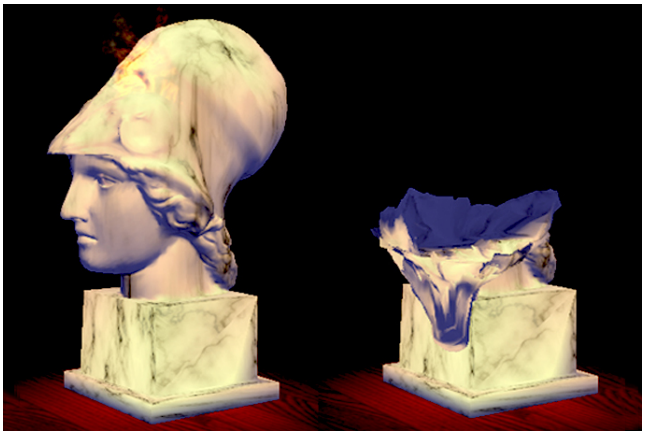

**Figure** 1: A wax like solid model (left) and after the melting effects taken place (right).

## 1. Introduction

One of the major goals in animation research is to be able to simulate the behavior of real-world materials in real-world situations. Maintaining a believable simulation of a familiar phenomenon can be quite a challenging task. In the game industry we require simulations that are visually compelling but not necessarily fully accurate imitations. Our aim is to achieve a visual effect that is as close as possible to that of scientific simulations, but at a small fraction of the computational cost.

We propose a method for emulating how solid objects melt under combustion. Each object in a 3D video game is represented by a polygonal mesh in the shape of its surface, which we will call a *shell model*. The challenge is to maintain a natural fluid-like behavior in this polygonal structure. Our goal is to introduce a polygon refinement method that can apply to any shell model (Figure 1, left), and support compelling visual replication of the melting process through to a burned-out and melted version of the object at the end (Figure 1, right).

The structure of the remainder of this paper is as follows. In Section 2 we describe some prior work. In Section 3 we describe our representation framework for the internal deformation and the key features of such a strategy. Section 4 describes our approach to implementing the structural deformation framework. Section 5 contains a few notes on our optimization techniques and results. Section 6 contains the conclusion and further work. The interested reader can see a video showing the burning and melting of an object using the techniques from this paper at [4].

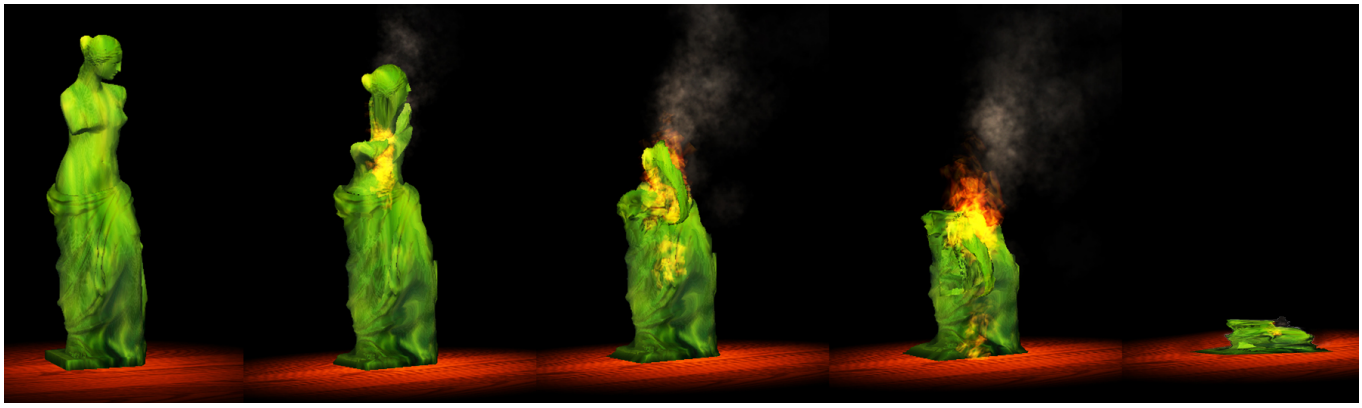

**Figure** 2: The melting of a solid model and the spread of procedural fire.

## 2. Previous Work

This paper extends our previous work on the emulation of burning objects in video games [1–3]. Amarasinghe and Parberry [2] laid down the foundation of our approach and demonstrated the ability to realistically burn a high-polygon count shell model of a toy satellite in real time on a relatively slow GPU. Amarasinghe and Parberry [1] extended this work to models with a very low polygon count by judicious use of procedural triangulation in the areas that are on fire, and demonstrated the ability to realistically burn on the same GPU a 12-triangle shell model of a door. This approach also lent itself easily to dynamic level of detail (LOD) rendering, which is an important method for reducing the polygon count in games. Amarasinghe and Parberry [3] extends this work from shell models to solid objects by procedurally filling in the exposed interior of a burning object.

Model deformation is a popular topic in the Computer Graphics community. We single out the following papers as relevant and significant, but without exception they strive for realism at the cost of performance. Although they are more realistic than our approach, their methods are not real-time and are therefore more useful for offline applications such as motion pictures than for video games.

Particle simulations are most popular techniques used to achieve fluid-like effects in computer graphics. These use a high number of small particles. Examples include Iwasaki [9], Foster and Fedkiw [7], Carlson, [5], Clavet [6], Keiser and Adams [11], and Goktekin and Bargteil [8]. Losasso and Irving [12] introduce a method for liquid or gas simulation using grid-based techniques including vortex confinement and the particle level set method. Müller, Keiser, *et al.* [15] laid a point-based framework for the volume and the surface representation, allowing arbitrarily large deviations from the original shape. Wei and Li [24] introduce a 3D cellular automata approach for melting objects. Terzopoulos and Platt [23] discuss deformable models featuring non-rigid dynamics governed by Lagrangian equations of motion and conductive heat transfer governed by the heat equation for non-homogeneous, non-isotropic media.

Melek and Keyser [13, 14] discuss techniques that were used in selected object deformation due to fire. Terzopoulost and Platt [22] introduce the theory of elasticity to describe deformable materials such as rubber, cloth, paper, and flexible metals. Sederberg and Parry [19] introduce a technique for deforming solid geometric models in a free-form manner. Tadmor and Phillips [21] and Nealen *et al.* [16] use finite element methods to deform complex geometries. Toivanen [20] discusses free deformation of meshes.

Finally, Rasmussen and Fedkiwr [17, 18] introduce high quality flame simulations that we use in our screenshots and videos, but they do not address the topic of object deformation.

## 3. Internal Deformation

Numerous chemical reactions take place in the interior of a burning object caused by the heat of combustion. Melek and Keyser [13] use the general term *pyrolysis* to describe this process. Volumetric expansion of heated material is caused by weakening bonds at the molecular level. Internal forces are disturbed by the effect of heat on unstable bond structure, ultimately leading to the consumption of material. Both thermal flow and the latent heat during the phase change bring the model to a state of melting (Jones, M.W. [10]). This causes changes in the shape of the object's affected areas.

The remainder of this section is divided into three subsections. Section 3.1 begins with a review of a simplified model of heat spread from Amarasinghe and Parberry [2]. Section 3.2 gives an overview of the deformation process and defines three new vertex classes, Flow True Vertices, Fixed Point Vertices, and Base Point Vertices. Section 3.3 investigates the melting process at the triangle level using Flow True and Fixed Point Vertices. Section 3.4 describes the motion of Base Point Vertices.

### 3.1 The Heat Boundary

The temperature of a burning object changes over time and space. The elevated temperature generated in the model due to fire effects have a strong influence on the mechanical behavior of the object. Conversely, the mechanical behavior of the given object influences the thermal response due to the thermal conductivity($\psi$) of the material. Absence of thermal equilibrium of the heat flux generates a *heat boundary*. To speed up computation, we approximate expansion of the heat boundary by calculating it around a single fixed point. We have followed the same functionality suggested in Amarasinghe and Parberry [2] by using the approximated heat boundary expansion given by:

$$R^2 = |\sin(\pi\Theta/\Delta r) + \sin(\pi\Theta) + \psi((x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2)|,$$

where $R = r + \Delta r$, the radius $r$ is incremented by $\Delta r$ in each $\Delta t$ time period. The angle $\Theta$ is a random value in order to make the expanding heat boundary irregular in shape. The location of the heat source is $(x_0, y_0, z_0)$. As we discussed in [2], the heat index can be approximated by a constant that depends on the size of the coarse triangles of the model.

In this paper we address the combustion of models with a large number of polygons. If a particular targeted triangle is considerably larger than the rest of the triangles, we can always apply the subdivision techniques from Amarasinghe and Parberry [1]. Thus, the designer can maintain a fixed heat index value that is suitable for the model and maintain the subdivision level accordingly.

The above boundary function creates a roughly spherical but irregular heat boundary around the heat source. In the real world, heat sources reproduce throughout the burning object as flames distribute over time. The multiple source heat boundary expands throughout the model with behavior similar to our single heat source approximation implemented using the above function. Since determination of the authentic heat boundary expansion is computationally expensive, we believe that the use of a single source heat boundary expansion is a viable alternative for use in video games.

We divide the heat boundary into four areas described in more detail in Amarasinghe and Parberry [1]. .

1. The *Virtual Heat Boundary* is spread through the model prior to the actual heat boundary expansion and is used to amortize essential calculations that could apply to the qualified triangles before the deformation process begins.
2. The *Initial Heat Boundary* is the area in which combustion is actively taking place and vertices are preparing to be deformed.
3. The *Combustion Ready Boundary* is where ignition starts.
4. The *Deform Boundary* consists of material that has been burned.

In order to implement melting we will need to add more computation within these boundaries as

described in the remainder of this paper. With composite simulations such as melting, one major challenge is to maintain the relative size of triangles so as to avoid empty triangles or sliver triangles. We will adapt the subdivision procedure introduced in Amarasinghe and Parberry [1] to the areas affected. These calculations will be applied to the essential triangles inside the Virtual Boundary.

### 3.2 The Deformation Process

The heat-induced deformation of an model mesh can be achieved by displacement of its vertices (see Amarasinghe and Parberry [2]). The position of each vertex depends on given properties such as vertex distance, gravitational force, viscosity damping force (Wei and Li [24]), and the internal forces work on each triangle pointing towards the direction of its vertices. However, when the model starts to melt we must also consider the effect of fluid-like behavior on the movement of the vertices.

Although in particle based fluid dynamics particle overlapping is tolerable, in mesh deformation vertex overlapping would result in unacceptable visual artifacts. Therefore, the movement of vertices must be systematic and restricted. We will use three new vertex categories (see Figure 3).

1. *Flow True Vertices* are vertices defined inside the Combustion Ready Boundary that are granted free movement within the given space.
2. *Fixed Point Vertices* are vertices located in the non-melting region of the mesh but carry an association with a polygon that contains Flow True Vertices.
3. *Base Point Vertices* are vertices in charge of guiding the flow of the fluid.

### 3.3 Polygonal Melting Properties

Heat weakens the bond strength between adjacent molecules of burning objects. This weakening effect falls off with distance from the heat source. As a result, surface molecules move in the direction of stronger bonds in order to find stable equilibrium between the forces that act on them. This results in stress on the burning area of the object. Melek and Keyser [13] also note that due to multi-

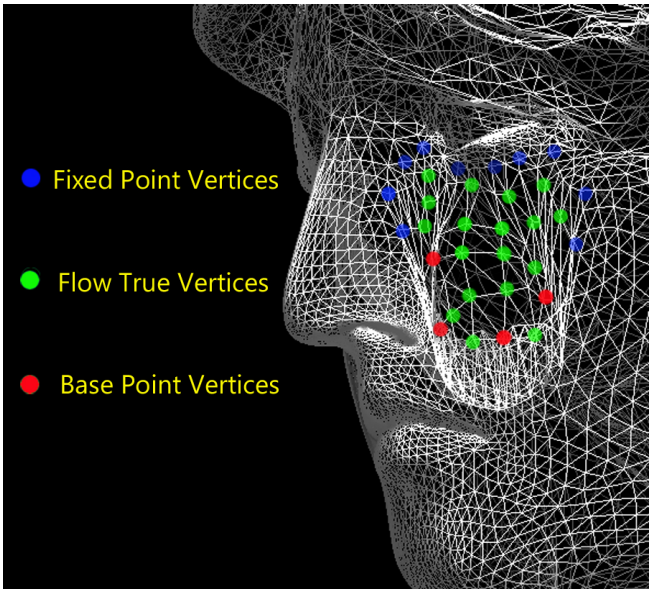


**Figure** 3: Categorized model vertices.

ple internal chemical reactions at various stages of the combustion process, material may change state from solid to liquid. Material under change may exhibit both fluid and solid characteristics. These are often referred to as *viscoelastic fluids*, which have the property that the material initially responds to strain elastically like a solid, but when subjected to increasingly large stresses it flows like a fluid (see Clavet, S. [6]). The melting stage takes place as sub-volume transformation (see Wei and Li [24]), in which the vertices perform free movement on limited space.

Since one of our prime effects of the melting process is to smear viscoelastic-fluid-like characteristics into the Flow True Vertices, our framework models the free movement, velocity and viscosity of the material. We achieve a fluid-like movement to a set of vertices by implementing a polygon folding procedure that we call the *Fractional Folding Method*. The following approach contributes a fluid like behavior to the vertices in the targeted area by eliciting a chain reaction among the Flow True Vertices towards a stable placement. This method will result in polygons maintaining a certain range in sizes during the simulation.

To calculate the effect of melting on a triangle $T$, first identify the top, middle and bottom vertices

(that is, sort them in order of $Y$ coordinate, breaking ties at random). Let $D$ be the minimum edge length of $T$. Compare $D$ with the constant range $\omega Dist/\chi$ to $\omega Dist$, where $\omega$ is the viscosity of the material, $Dist$ is the average distance of a side of a triangle in original mesh (after subdivision takes place if required) and $\chi$ is a constant integer value that defines how small the smallest triangle should be. Vertices are modified as follows:

$$\text{if}(D > \omega\text{Dist})$$
$$(D_{sx}, D_{sz}) = (D_{sx}, D_{sz}) - ((D_{sx}, D_{sz}) - (D_{dx}, D_{dz})) * v/(n+1)$$

$$\text{if}(D < \omega Dist/\chi)$$
$$(D_{sy}, D_{sx}, D_{sz}) = (D_{sy}, D_{sx}, D_{sz}) + ((D_{sy}, D_{sx}, D_{sz}) - (D_{dy}, D_{dx}, D_{dz})) * v/(n+1)$$

where $v$ is the velocity that determines how fast the vertices must move. If a Flow True Vertex appears with $n$ Fixed Point Vertices in the same polygon, we factor the velocity of the movement by $n+1$.

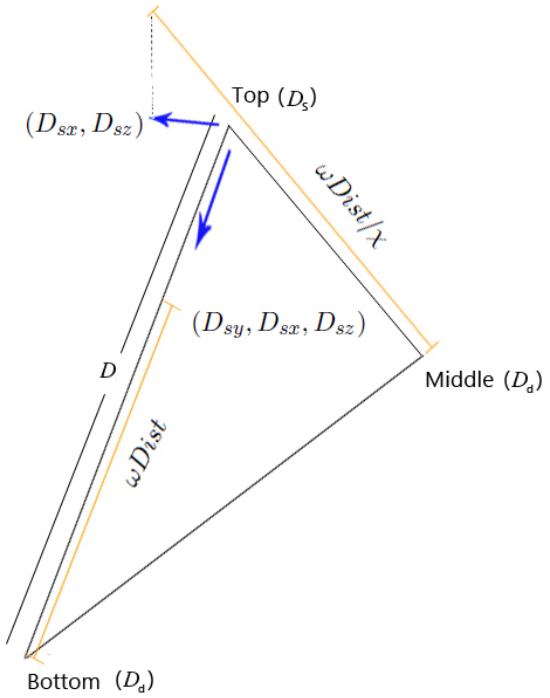


**Figure** 4: The deformation coordinates of a single triangle.

Since upward fluid motion would not make sense, when the contraction of the vertices ($D > \omega dis$) occurs, only x and z directional placement will be considered. After they reach their stable position, the motion of the vertices taking part in Fractional Folding will come to a halt until the next trigger occurs. The bodily flow of the melted substances needs a proper directional trigger to begin movement. We use the flow of Base Point Vertices for this purpose

| Attribute | Values | Description |
|---|---|---|
| $Dist$ | $float$ | Average distance |
| $\omega$ | $0 < \omega < 1$ | Viscosity |
| $\chi$ | $int(1-10)$ | Adjustment factor |
| $v$ | $0 < v < 1$ | Velocity |

Table 1: ARP Attribute Set.

### 3.4 Base Point Movement

We identify Base Point Vertices by adapting the *Block Sampling Method* from Amarasinghe and Parberry [1]. This method divides the object into uniform blocks and treats each block as a single unit, propagating changes to neighboring blocks. We choose one Base Point Vertex per block and keep them in an inactive state. They become active when they are reached by the Combustion Ready Boundary. Since flow of the viscous fluid must pass over the solid surface of the object, we perform a collision check with the points in surrounding bounding boxes. In addition, usually fluid flows towards the bottom of the object due to gravitational influence. We first find the neighboring block that contains the lowest Base Point Vertex, then, transform coordinates towards the lowest point. The transformation of the Base Point Vertex is given by:

$$(x, y, z) = (x, y, z - x_d, y_d, z_d)\omega + (x_d, y_d, z_d)$$

where the $(x_d, y_d, z_d)$ are the destination coordinates and $\omega$ is the viscosity.

A block absent one of its left, right, front or back neighbors is considered to be an *edge block*. Whenever an edge block has no bottom neighbor, Base Point Vertices fall under gravity: $Y = Y - \omega\vec{g}/(Y_{crb} - Y)$, where $\vec{g}$ is the gravity vector and $Y_{crb}$ is the $Y$ distance between the Combustion Ready Boundary and $Y$. This will provide a deceleration

when the vertices move away from the flames. Similarly, the rest of the Flow True Vertices will follow the Base Points.

## 4. Structural Deformation

In melting material the structural changes largely depend on the weight that is supported by the melting area. When combustion starts from the top of an area we assume that there is no weight supported and the structural changes can be ignored. When combustion starts from the middle or bottom, the weight effect causes a significant shape change. We model this by applying the following calculations to the area outside of the Combustion Ready Boundary and above the ignition point.

We use the Block Sampling Method described in Amarasinghe and Parberry [3]. We construct a bounding box around the object, and then decompose it into a grid of smaller axially aligned bounding boxes which we shall call *blocks*. Define the *weight* of a block to be the number of vertices inside it. We keep track of the orientation of each block as a triple of Euler angles. Weights of the blocks are decided according to the number of vertices inside it. Empty blocks of weight zero are discarded.

Although a change to one block may affect all of the blocks in the model, to reduce the computational load we apply changes to only immediate neighboring blocks, and rely on subsequent iterations to propagate the effects further. The change of the weight in each block results in a slight rotation of the box around its midpoint. The direction of the rotation will be determined by the placement of the displaced vertex compared to the midpoint of the box.

Stability will change due to the rotation of the immediate neighboring boxes. The change in roll angle $R$ (pitch and yaw are similar) for a block is: $R = \gamma\rho\pi/NM$, where $\gamma$ is a scaling factor chosen by the designer, $\rho$ is a measure of the material density of the model in that block, $N$ is the number of vertices in the block, and $M$ is the current number of nonempty neighboring blocks.

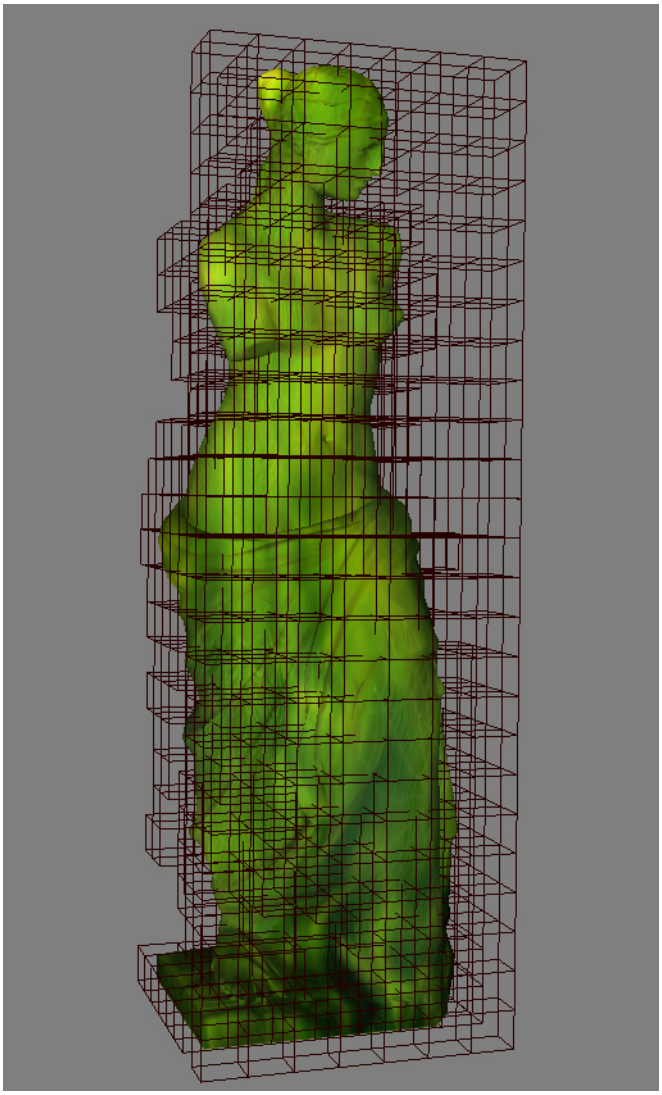

**Figure** 5: The model subdivided into blocks.

## 5. Results

Our algorithm was implemented in CUDA on relatively modest hardware; An Intel®Core™2 Duo CPU P8400 @ 2.26GHz processor with an NVidia GeForce 9800 GTS graphics card. We were able to maintain 60fps frame rate up to 40k triangle model with balanced settings (quality vs. performance) in the graphic card. This performance will of course be much better on the current generation of graphics hardware, and thus able to run in parallel with other rendering tasks and game-related computation. We used approximately 2000 fire particles and 500 smoke particles to demonstrate the visual effects.

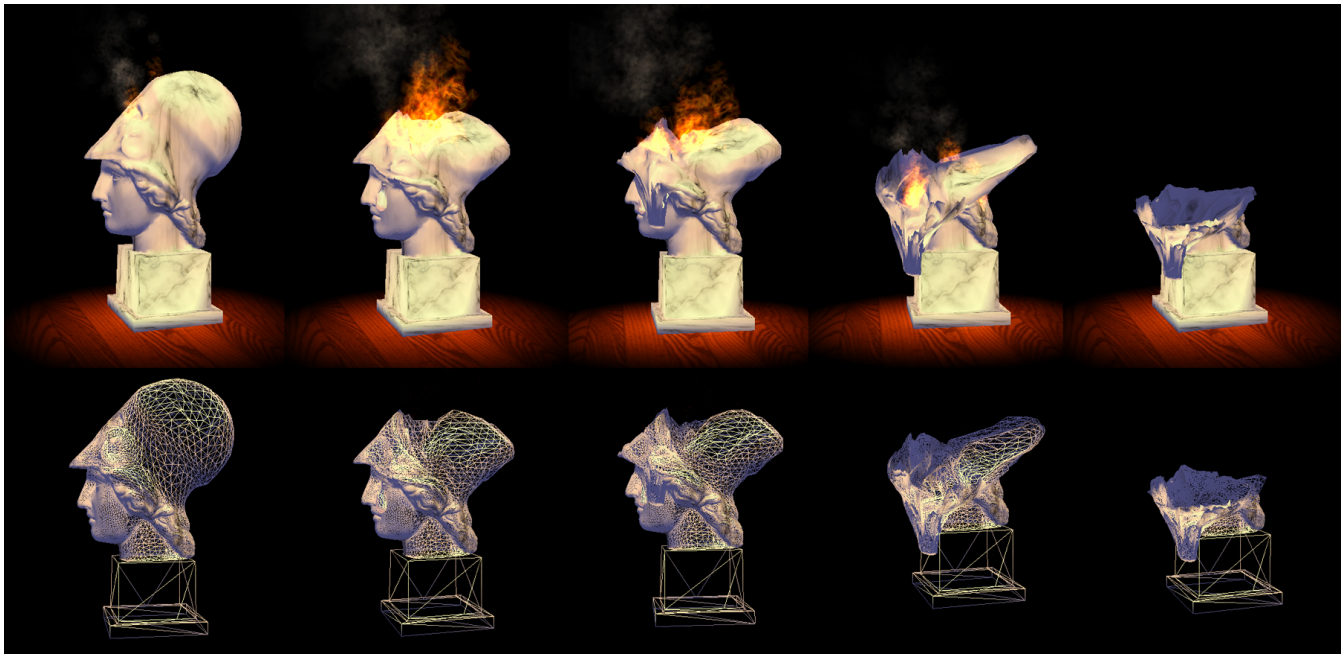

**Figure** 6: Melting sequence of a wax form solid model

## 6. Conclusion

We have described a method for the real-time melting of a wax type solid model during combustion by procedurally generated fire, extending our previous work on shell models [1–3]. We were able to successfully perform our simulation on models of various mesh resolution and topology on less than cutting-edge hardware. We believe that our approach maintains a reasonable amount of realism sufficient to trigger willing suspension of disbelief in the game player. Our simulations perform well on various models ranging from a dozen to hundreds of thousands of triangles. The interested reader can see a video showing the burning and melting of an object using the techniques from this paper at [4]. Open problems remaining include the introducing an efficient and effective heat boundary expansion method other than single point heat source.

## References


[1] D. Amarasinghe and I. Parberry. Fast, believable real-time rendering of burning low-polygon objects in video games. In *Proc. 6th Internat. North American Conf. on Intelligent Games and Simulation (GAMEON-NA)*, pages 21–26. EUROSIS, 2011.

[2] D. Amarasinghe and I. Parberry. Towards fast, believable real-time rendering of burning objects in video games. In *Proc. 6th Annual Internat. Conf. on the Foundations of Digital Games*, pages 256–258, 2011.

[3] D. Amarasinghe and I. Parberry. Real-time rendering of burning solid objects in video games. Technical Report LARC–2012–01, Laboratory for Recreational Computing, Dept. of Computer Science & Engineering, Univ. of North Texas, 2012.

[4] D. Amarasinghe and I. Parberry. `http://larc.unt.edu/ian/research/fire4/`, 2013.

[5] M. Carlson, P. Mucha, R. Van Horn III, and G. Turk. Melting and flowing. In *Proc. 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 167–174. ACM, 2002.

[6] S. Clavet, P. Beaudoin, and P. Poulin. Particle-based viscoelastic fluid simulation. In *Proc. 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 219–228. ACM, 2005.

[7] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proc. 28th Annual Conference on Computer graphics and Interactive Techniques*, pages 23–30. ACM, 2001.

[8] T. Goktekin, A. Bargteil, and J. O'Brien. A method for animating viscoelastic fluids. In *ACM Transactions on Graphics*, volume 23, pages 463–468.

ACM, 2004.

[9] K. Iwasaki, H. Uchida, Y. Dobashi, and T. Nishita. Fast particle-based visual simulation of ice melting. In *Computer Graphics Forum*, volume 29, pages 2215–2223. Wiley Online Library, 2010.

[10] M. Jones. Melting objects. *The journal of WSCG*, 11(2), 2003.

[11] R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutré, and M. Gross. A unified lagrangian approach to solid-fluid animation. In *Proc. Eurographics/IEEE VGTC Symposium on Point-Based Graphics*, pages 125–148. IEEE, 2005.

[12] F. Losasso, G. Irving, E. Guendelman, and R. Fedkiw. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):343–352, 2006.

[13] Z. Melek and J. Keyser. An interactive simulation framework for burning objects. Technical Report 2005-03-1, Dept. of Computer Science, Texas A&M University, 2005.

[14] Z. Melek and J. Keyser. Driving object deformations from internal physical processes. In *Proc. 2007 ACM Symp. on Solid and Physical Modeling*, pages 51–59, New York, NY, USA, 2007. ACM Press.

[15] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proc. 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 141–151. Eurographics Association, 2004.

[16] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25 (4):809–836, 2006.

[17] D. Q. Nguyen, R. Fedkiw, and H. W. Jensen. Physically based modeling and animation of fire. In *Proc. 29th Annual Conf. on Computer Graphics and Interactive Techniques*, pages 721–728, New York, NY, USA, 2002. ACM Press.

[18] N. Rasmussen, D. Nguyen, W. Geiger, and R. Fedkiw. Smoke simulation for large scale phenomena. *ACM Transactions on Graphics*, 22(3):703–707, 2003.

[19] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. *ACM SIGGRAPH Computer Graphics Quarterly*, 20(4):151–160, 1986.

[20] J. Simo and F. Armero. Geometrically non-linear enhanced strain mixed methods and the method of incompatible modes. *Internat. J. for Numerical Methods in Engineering*, 33(7):1413–1449, 1992.

[21] E. Tadmor, R. Phillips, and M. Ortiz. Mixed atomistic and continuum models of deformation in solids. *Langmuir*, 12(19):4529–4534, 1996.

[22] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *ACM SIGGRAPH Computer Graphics Quarterly*, 21(4):205–214, 1987.

[23] D. Terzopoulos, J. Platt, and K. Fleischer. Heating and melting deformable models. *The Journal of Visualization and Computer Animation*, 2(2):68–73, 1991.

[24] X. Wei, W. Li, and A. Kaufman. Melting and flowing of viscous volumes. In *Proc. 16th International Conference on Computer Animation and Social Agents*, pages 54–59. IEEE, 2003.