Towards Fast, Believable Real-time Rendering of Burning Objects in Video Games

Dhanyu Amarasinghe and Ian Parberry

Technical Report LARC-2010-04

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

October, 2010

Towards Fast, Believable Real-time Rendering of Burning 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 framework for emulating the deformation and consumption of polygonal models under combustion while generating procedural fire. Our focus is on achieving the best visual effects possible while maximizing computation speed so that the processing power is available for other tasks in video games. We have implemented and tested our method on a relatively modest GPU using CUDA. Our experiments suggest that our method gives a believable rendering of the effects of fire while using only a small fraction of CPU and GPU resources.

Keywords Hardware acceleration, volume rendering, freeform deformation, procedural, generation, fire modeling, CUDA.

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

1. Introduction

One way a new video game can make an impact on players is by increasing realism over and above what the player is accustomed to in other games. Replicating the details of a physical process such as fire can increase the believability of virtual worlds, and draw the player into willing suspension of disbelief. To date, developers mostly use model-swapping techniques to implement a crude level of model deformation by combustion. We introduce a technique for performing real-time emulation of a burning object such as shown in Figure 1 while maintaining system performance.

Achieving high quality visual effects using minimum computational resources can obviously be very challenging. Modeling an object undergoing combustion includes, for example, heat boundary expansion, flame distribution, fuel consumption, and shape deformation over time. Our aim is to increase believability by a large amount while increasing computation only minimally. It should be emphasized that we are looking for plausability sufficient to trigger willing suspension of disbelief, not a picture-perfect simulation of real world combustion. While it is true that future advances in hardware will make today's slow methods feasible, we prefer to focus on achieving better results on hardware currently available to millions of gamers worldwide.

The structure of the remainder this paper is as follows. In Section 2 we describe some previously published related 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 CUDA implementation. Section 6 contains the conclusion and further work.

For more images and some video from our CUDA implementation, see [1].

2. Previous Work

Melek and Keyser [5, 7] discuss techniques that were used in selected object deformation due to fire, however, these methods are designed to maximize realism at the cost of performance. Sederberg and Parry [11] and Hsu, Hughes, and Kaufman [4] introduce some adaptive techniques of deformation. Müller and McMillan [9] discuss real-time techniques for deformation focussing on selected materials. Toivanen [12] discusses free de-

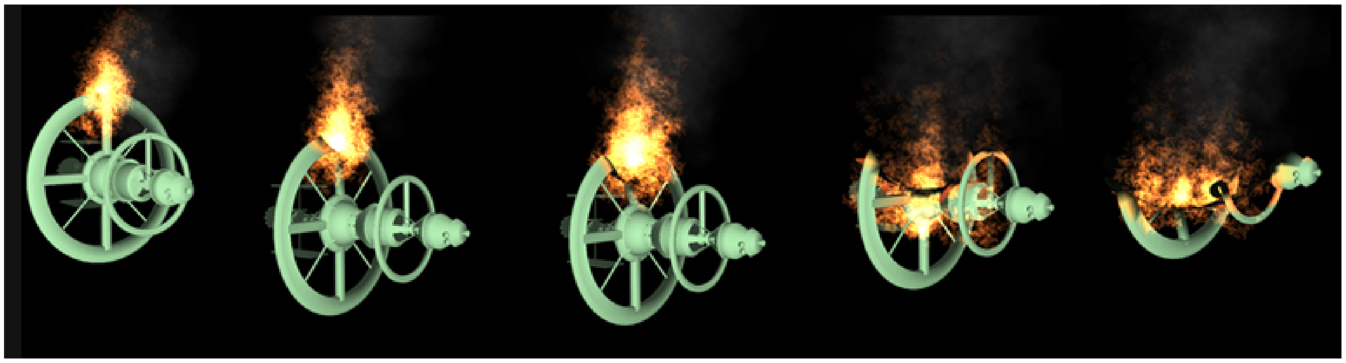

Figure 1: The consumption of a model and the spread of procedural fire.

formation of meshes, but his technique is too computationally intensive for use in video games.

Nguyen and Fedkiw [10] introduce high quality flame simulations, but do not address object deformation. Wei and Zhao [14] use an approach similar to Melek, defining solids as a volumetric implicit field, but also do not discuss object deformation. Wei and Li [13] use splatting techniques that help to increase visual impact. Moidu, Kuffner, and Bhat [8] demonstrate an attempt to animate deformable materials such as paper, but they do not introduce complex heat transfer models. Although the paper did discuss the spring-mass model technique to emulate combusting surfaces, they focus on selective materials such as paper and cloth. Fuller [3] has a useful method for generating procedural volumetric fire in real time using curve-based volumetric free-form deformation.

3. Internal Deformation

The geometry and topology of a burning object changes as heat spreads. Melek and Keyser [7] noted that there are multiple internal chemical reactions at various stages of the process, during which its properties may change from solid to liquid and from liquid to gas due to volumetric expansion caused by weakening bonds at the atomic level. The change in bond strength disturbs the stability of the internal forces between atoms. This causes the changes in the shape of the object's affected areas.

The remainder of this section addresses the topic if internal deformation in two subsections. Section 3.1 first discusses how to model heat spread. We then show how to mimic atomic behavior by vertex displacement in Section 3.2.

3.1 The Heat Boundary

In the real world, 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 and conversely, the mechanical behavior of the given object influences the thermal response due to the thermal conductivity of the material. Heat transfer calculations depend on many parameters including environmental factors such as humidity (see Ang and Gumel [2]).

To speed computation, we approximate expansion of the heat boundary by calculating it around a single fixed point. This creates a roughly spherical but irregular heat boundary around the heat source. However, heat sources multiply when the flame distributes throughout the model. Heat spread over a given material depends on the *thermal conductivity* of that material (Melek and Keyser [6]), which indicates its ability to conduct heat. We model thermal conductivity using a *heat index* constant ψ. The value of ψ should depend on the size of the triangles used in the model and the material that the model is made from. We then use the following function to emulate an approximated heat boundary expansion:

$$\begin{aligned} R^2 &= |\sin(\pi\Theta/\Delta r) + \sin(\pi\Theta) + \\ &\quad \psi((x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2)|, \end{aligned}$$

where $R = r + \Delta r$, the radius r is incremented by Δr in each Δt time period. The angle Θ is a random value in order to make the expanding heat boundary irregular in shape (otherwise the heat boundary will be perfectly spherical, which would appear unnatural). The location of the heat source is (x_0, y_0, z_0).

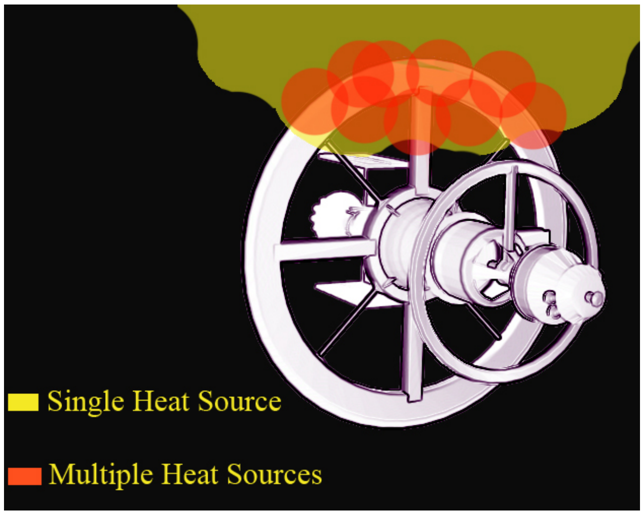

Figure 2: Heat boundary for single vs. multiple heat sources

Figure 2 illustrates the similarity of the approximated heat boundary expansion for single versus multiple heat sources. The multiple source heat boundary expands throughout the model with behavior similar to our single heat source approximation implemented using the above function. Because determination of the authentic heat boundary expansion is computationally expensive, we believe our single source heat boundary expansion is a viable alternative for use in video games.

In Figure 3 we divide the heat boundary into three different areas, the *initial heat boundary area* in which combustion is actively taking place and vertices are being deformed, the *combustion-ready area* in which ignition starts, and the *deformed area* which has been burned and is a candidate for surface removal.

3.2 Deformation

Internal deformation is achieved by displacement of the vertices of the model mesh. The position of each vertex will depend on three properties: vertex distance, gravitational force, and material index. While we may assume that material is a constant over large areas of the model, vertex distance and

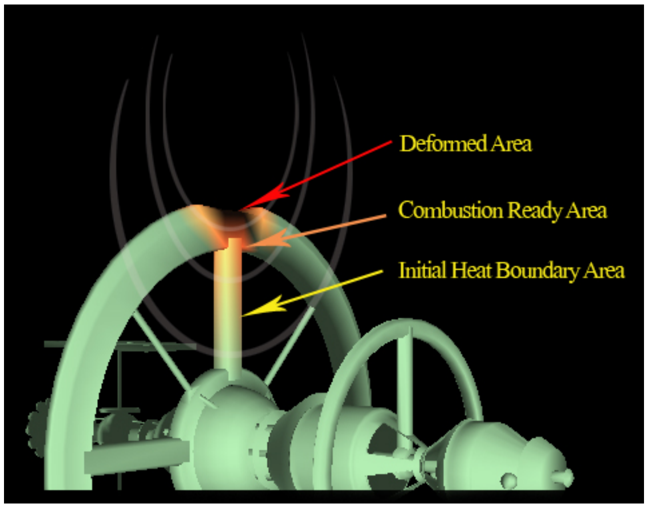

Figure 3: Division of the heat boundary into three parts, (bottom to top) the *initial heat boundary area*, the *combustion-ready area*, and the *deformed area*.

gravitational force are more complicated, and will be examined next.

We consider each vertex of the object to be analogous to an atom, and we take the distance between vertices of a given triangle as the strength of the bond between those vertices. The bond between two vertices of a triangle is taken to be inversely proportional to the distance between them. Vertex displacement is inversely proportional to bond strength, that is, directly proportional to distance, and scaled by material index.

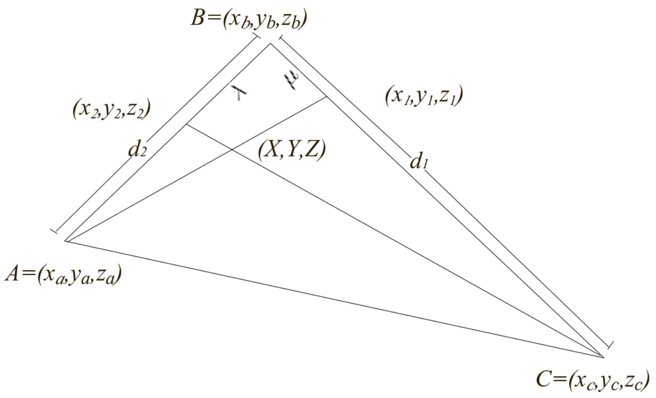

Figure 4: The deformation coordinates of a single triangle.

We make use of the following designer-set constants, $L, \varepsilon, \beta, \rho, \phi$ described in Table 1. The first is an integer, the remainder are real-valued con-

Name	Type	**Description**	**Level**
L	Integer	Flammability	Vertex
ε	$0 < \varepsilon < 1$	Meltability	Edge
β	$0 < \beta < 1$	Displacement scale	Block
ρ	$0 < \rho < 1$	Material density	Block
ϕ	$0 < \phi < 1$	Bond strength	Block

Table 1: Designer set constants.

stants between zero and one. All of the values can be made constant for the entire model, but in principle L can be different for each vertex, ε can be different for each edge, and β, ρ, and ϕ can be different for each block.

Suppose B is a vertex to be displaced in triangle ABC, where $A = (x_a, y_a, z_a)$, $B = (x_b, y_b, z_b)$, and $C = (x_c, y_c, z_c)$. B is to be displaced to (X, Y, Z), as follows:

$$\begin{aligned}
X &= (x_1x_2(y_a - y_c) + x_1x_a(y_c - y_2) \\
&\quad +x_cx_2(y_1 - y_a) + x_ax_c(y_2 - y_1))/ \\
&\quad ((x_a - x_2)(y_c - y_1) - (x_c - x_1)(y_a - y_2)) \\
Y &= (y_1y_2(x_a - x_c) + y_1y_a(x_c - x_2) \\
&\quad +y_cy_2(x_1 - x_a) + y_ay_c(x_2 - x_1))/ \\
&\quad ((y_a - y_2)(x_c - x_1) - (y_c - y_1)(x_a - x_2)) \\
Z &= (z_1z_2(y_a - y_c) + z_1z_a(y_c - y_2) \\
&\quad +z_cz_2(y_1 - y_a) + z_az_c(y_2 - y_1))/ \\
&\quad ((z_a - z_2)(y_c - y_1) - (z_c - z_1)(y_a - y_2))
\end{aligned}$$

where

$$\begin{aligned}
(x_1, y_1, z_1) &= \mu C + (d_1 - \mu)B \\
(x_2, y_2, z_2) &= \lambda A + (d_2 - \lambda)B.
\end{aligned}$$

Figure 4 illustrates the coordinates and parameters used in these equations. The values λ and μ are the contraction amounts along each edge due to the effect of heat. The lengths of BC and BA are d_1 and d_2 respectively. The points (x_1, y_1, z_1) and (x_2, y_2, z_2) are a μ and λ fraction respectively of the length along the edges (respectively BC and BA) of the triangle. The values μ and λ are displacement parameters specifically for vertex B. They measure the amount that the bond between B and its neigboring vertices is changed by temperature.

In addition we use a *displacement adjustment parameter* β to allow for the variation in triangle size from one model to another. ρ denotes a *material density index*. When both vertices of an edge are inside the heat boundary, bond strength is weaker by a factor of ϕ than when one vertex is outside of the heat boundary.

Burning objects are consumed by combustion, and combustion subsides when there is nothing left to consume. We model this with a *flammability* value L at each vertex. The counter decreases each time vertex displacement is processed. A the level counter of zero indicates that there are no consumable resources left at the vertex. The designer sets the initial flammability value for each vertex to mimic the effect of having different parts of the model constructed from physical materials of varying flammability such as wood or metal. λ is then defined to be $\beta\rho L/d_2$ if A is outside the heat boundary, and $\phi\beta\rho L/d_2$ otherwise μ is defined similarly, replacing d_2 with d_1.

Finally, among all of the external forces, gravity plays a major part in every physical based simulation. Let ε be a constant that represents the amount that the model melts due to heat, and $\vec{g}$ be the gravity vector. Then the effect of gravity is computed as follows: $Y = Y - \varepsilon\vec{g}$.

4. Structural Deformation

Deformation of a burning object can be caused by factors such as the expansion and weakening of the internal bonds, and the relative weights of cantilevered parts of the object. Exact calculation of these complex processes is costly. Therefore, we simplify the process by considering only the weight of a given point of the structure. The weight changes of the burning structure will occur due to consumption of the object by fire. Following Melek and Keyser [7], we divide the object into uniform blocks and treat each block as a single unit, propagating changes to neighboring blocks.

We start by constructing an oriented bounding box around our object, then decompose it into a grid of smaller axially aligned bounding boxes which we will call *blocks*. Deciding the number of blocks per model is up to the designer. Higher numbers of smaller blocks will make the effect more realistic at the cost of lower performance.

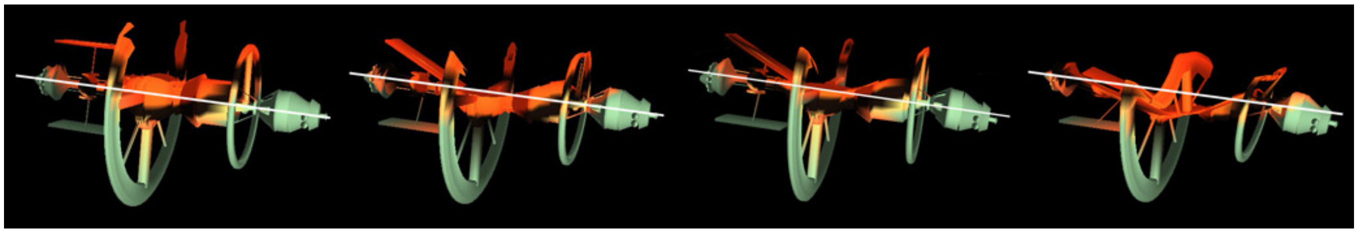

Figure 5: Structural deformation of the object: On the left we see the original model, and moving left to right we see the results of three separate runs of our implementation with increasing values of ε. Notice how the model has slumped compared to the straight line in white.

We weight the blocks according to the number of vertices in them, and discard the empty ones of weight zero. Figure 6 shows a model subdivided into blocks. Only nonempty blocks are shown. We use block weight to model flame distribution, under the assumption that a block with more vertices contains more material, and thus will produce more flames.

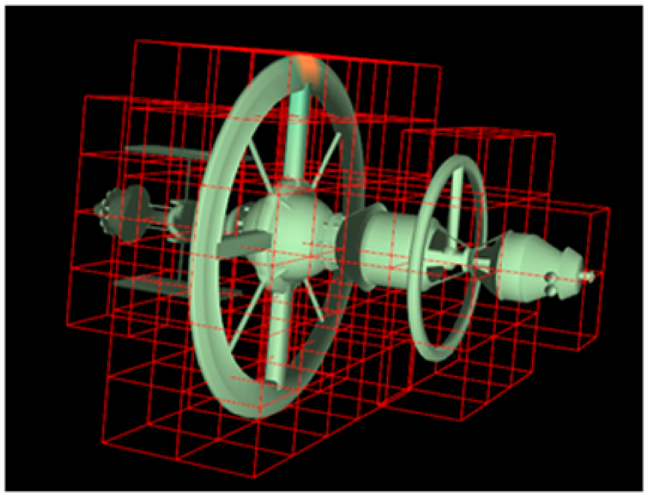

Figure 6: The model subdivided into blocks.

We store for each block the amount of rotation, the midpoint of each box, the number of vertices, and the list of neighboring blocks. Since all the blocks are interconnected, a change to one block may affect all of the blocks in the model. To limit the computation required we apply changes to only immediate neighboring blocks, and rely on time to propagate the effects further.

The weight of each block changes as vertices are removed during the process. The change of the weight in the block is indicated by 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. Interestingly, we have found applying a random rotation also gives satisfactory results of the structural deformation.

For video game applications the random rotation may actually be sufficient, and more efficient since calculating the position of displaced vertices can be costly. Stability will change due to the rotation of the immediate neighboring boxes. In order to cope with this we keep track of neighbors of each subunit by maintaining a data structure that contains neighbor indices, rotation amount, number of vertices, etc.

We keep track of the orientation of each block as a triple of Euler angles. The change in roll angle R (pitch and yaw are similar) for a block is: $R = \gamma\rho\pi/NM$, where γ is a scaling factor chosen by the designer, ρ 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 shows an example of the resulting deformation. Mapping of deformation from the block level to the vertex level is done by a vertex shader in our CUDA implementation.

Surface removal is handled at the block level. A block is removed when it contains no vertices, provided its removal does not disconnect the object into separate parts.

5. CUDA Implementation

All the images of a burning model shown in this paper are screenshots from a CUDA implementation of our algorithm. The flames are generated using

3000 fire particles and 1800 smoke particles. The model has almost 16K triangles. The animation runs at 70fps) on relatively modest hardware; An Intel®Core^{TM}2 Duo CPU P8400 @ 2.26GHz processor with an NVidia GeForce 9800 GTS graphics card. This performance will be much better on the current generation of graphics hardware

6. Conclusion and Future Work

We have proposed a method for the real-time deformation and consumption of a polygonal model during combustion by procedurally generated fire. We have focused on the performance with a reasonable amount of realism sufficient to trigger willing suspense of disbelief in the game player. We believe this our method is the first of its kind. It takes into account a variety of physical properties including material density indexes, material indexes, heat distribution, gravity, structural and internal deformation, and flame distribution.

Our method performs well on a model with fairly high polygon count and small triangles. It remains to apply our results to models with a low polygon count. We suggest that triangle subdivision is an intelligent first move. Most of the models used in video games are so-called *shell models*. Deformation of shell models is different from solid models, and they should burn differently. Emulation of consumption and deformation of solid models is left as an open problem, as is a more realistic approach to the heat boundary.

References

[1] D. Amarasinghe and I. Parberry. Fire, 2010. `http://www.eng.unt.edu/ian/research/fire/`.

[2] W. Ang and A. Gumel. A boundary integral method for the three-dimensional heat equation subject to specification of energy. *Journal of Computational and Applied Mathematics*, 135(2):303–311, 2001.

[3] A. R. Fuller, H. Krishnan, K. Mahrous, B. Hamann, and K. I. Joy. Real-time procedural volumetric fire. In *I3D '07: Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, pages 175–180, New York, NY, USA, 2007. ACM.

[4] W. Hsu, J. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 177–184. ACM, 1992.

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

[6] Z. Melek and J. Keyser. Multi-representation interaction for physically based modeling. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, pages 187–196. ACM, 2005.

[7] Z. Melek and J. Keyser. Driving object deformations from internal physical processes. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, pages 51–59, New York, NY, USA, 2007. ACM.

[8] S. Moidu, J. Kuffner, and K. S. Bhat. Animating the combustion of deformable materials. In *ACM SIGGRAPH 2004 Posters*, page 90, New York, NY, USA, 2004. ACM.

[9] M. Müller, L. McMillan, J. Dorsey, and R. Jagnow. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic Workshop on Computer Animation and Simulation*, pages 113–124. Citeseer, 2001.

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

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

[12] J. Toivanen. A Non-Linear Mesh Deformation Operator Applied to Shape Optimization.

[13] X. Wei, W. Li, K. Mueller, and A. Kaufman. Simulating fire with texture splats. In *IEEE Visualization*, pages 227–234, 2002.

[14] Y. Zhao, X. Wei, Z. Fan, A. Kaufman, and H. Qin. Voxels on fire. *Visualization Conference, IEEE*, page 36, 2003.