Proceedings of the 18th International Conference on Computer Games, pp. 139-143, Louisville, Kentucky, USA, 2013.

Real-time Rendering of Burning Solid Objects in Video Games

Dhanyu Amarasinghe
Ian Parberry
Dept. of Computer Science & Engineering
University of North Texas
Denton, Texas 76203–5017
Email: {DhanyuAmarasinghe | ian}@unt.edu

***Abstract*—Objects in 3D games are typically *shell models*, a polygon mesh representing the shell or skin of the object. While emulation of the behaviour of shell models under combustion is sufficient for many game applications and is fairly well studied, solid objects do in fact burn rather differently than shell objects. We show how to manipulate shell models so that they appear to burn as solid models. Since our burning objects will be only a small part of a video game, computation speed is of the essence. We demonstrate that our method uses only a fraction of the computational power available by implementing the computation on a modest GPU using CUDA.**

***Index Terms*—Procedural content generation, video game, thermal response, combustion, deformation, polygon mesh.**

I. Introduction

Many cutting edge console and PC games compete to attract seasoned players by increasing realism over and above what they are accustomed to in other games. Replicating the details of a physical process such as fire can readily draw the player into willing suspension of disbelief.

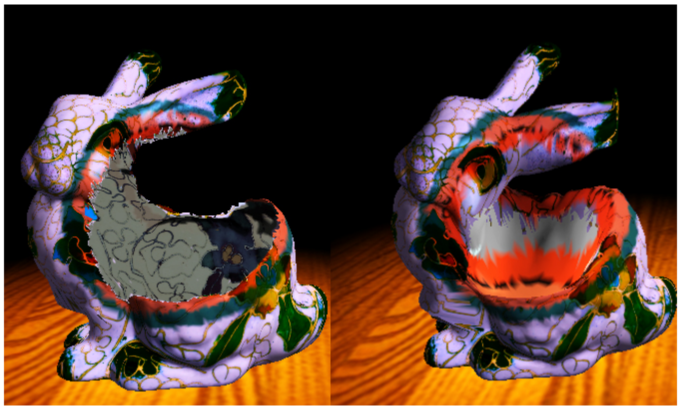

Fig. 1. Shell model deformation (left) vs solid model deformation (right).

The typical object in a 3D video game is represented by a polygon mesh in the shape of its surface, which we will call a *shell model*. The problem of applying shape changes to a shell model by emulating solid object properties without overloading the available computational resources is a challenging one. We propose to tackle the emulation of solid object deformation and consumption under combustion. Solid objects are expected to burn much differently than shells. Aside from the obvious difference of being able to see the inside of a burned-out shell (Figure 1, left), a solid object will melt and deform under heat in a different way (Figure 1, right).

The structure of the remainder of this paper is as follows. In Section II we describe some prior work. In Section III we describe our representation framework for the internal deformation and the key features of such a strategy. Section IV describes our approach to implementing the structural deformation framework. Section V contains a few notes on our optimization techniques and results. Section VI contains the conclusion and further work.

II. Previous Work

This paper extends our previous work on the emulation of burning objects in video games. Amarasinghe and Parberry [1] laid down the foundation of our approach and demonstrated the ability to realistically burn in real time on a relatively slow GPU a high-polygon count shell model of a toy satellite. Amarasinghe and Parberry [2] 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 rendering.

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. Melek and Keyser [3], [4] discuss techniques that were used in selected object deformation due to fire. Terzopoulost and Platt [5] introduce the theory of elasticity to describe deformable materials such as rubber, cloth, paper, and flexible metals. Sederberg and Parry [6] introduce a technique for deforming solid geometric models in a free-form manner. Tadmor and Phillips [7] and Nealen *et al.* [8] use finite element methods to deform complex geometries. Toivanen [9] discusses free deformation of meshes.

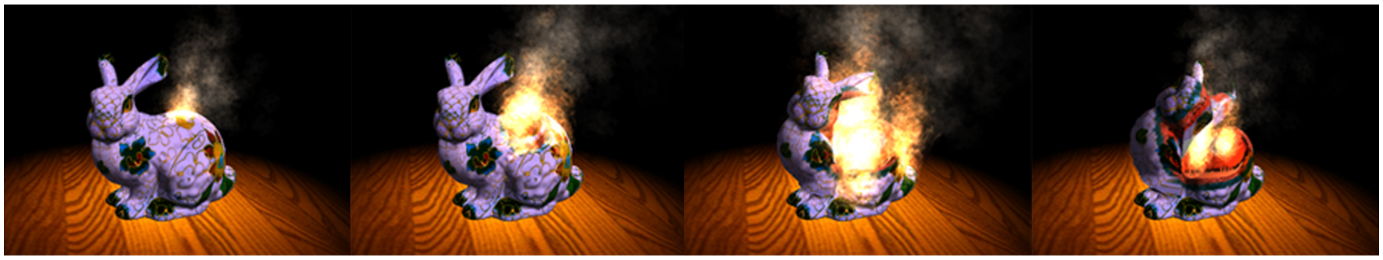

Fig. 2. The combustion of a solid model and the spread of procedural fire.

Finally, Rasmussen and Fedkiwr [10], [11] introduce high quality flame simulations that we use in our experiments, but they do not address object deformation.

III. Internal Deformation

According to Melek and Keyser [3], when an object burns there are assorted interior chemical reactions at various stages that lead its properties to change in a process called *pyrolysis*. 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. This causes changes in the shape of the object's affected areas. We begin by creating a simplified model of heat spread.

A. The Heat Boundary

The temperature of a burning object changes over both time and space. The increase in temperature generated by fire changes the mechanical behavior of the object. Significant thermal response occurs due to the thermal conductivity of the material. Absence of thermal equilibrium of the heat flux generates a *heat boundary*. As in Amarasinghe and Parberry [1], we approximate the expansion of the heat boundary by calculating it around a fixed solitary point using the following function:

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

where $R = r + \Delta r$ indicates that 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. The location of the heat source is (x_0, y_0, z_0). As we discussed earlier in [1], 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 solid models with an arbitrary number of polygons. If the targeted triangle is considerably larger than the rest of the triangles, we can always apply the subdivision techniques from Amarasinghe and Parberry [2]. 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. Figure 3 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 that the use of a single source heat boundary expansion is a viable alternative for use in video games.

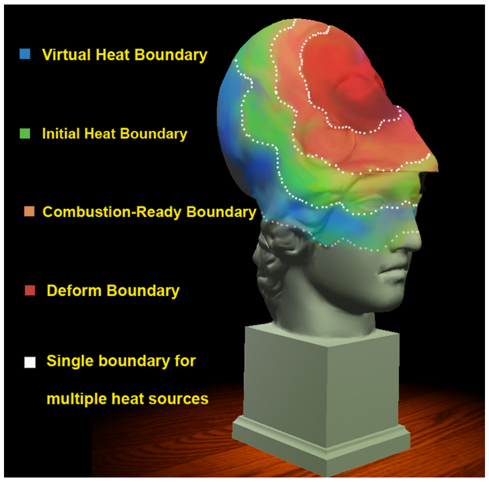
🟦 Virtual Heat Boundary
🟩 Initial Heat Boundary
🟧 Combustion-Ready Boundary
🟥 Deform Boundary
⬜ Single boundary for multiple heat sources

Fig. 3. Heat boundary for single vs. multiple heat sources with different levels of boundary.

As shown in Figure 3, we divide the heat boundary into four different areas. 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. The other three boundaries are those introduced in Amarasinghe and Parberry [1]; the *Initial Heat Boundary* in which combustion is actively taking place and vertices are preparing to be deformed, the *Combustion Ready Boundary* where ignition starts, and the *Deform Boundary* consisting of material that has been burned.

B. The Deformation Process

Surface removal as practiced in our prior paper Amarasinghe and Parberry [1] is less useful in solid models than in shell models because the consumption of material in a solid model simply reveals more material just underneath it. Consequently solid models have more triangles to deform than shell models, and these need to be managed efficiently and effectively. In order to achieve this we categorize model triangles into three major types as shown in Figure 4. Those are called Boundary Qualified Triangles, Combustion Qualified Triangles, and Deforming Triangles. *Boundary Qualified Triangles* are the triangles located inside the Virtual Heat Boundary. These can be completely or partially contained within the heat boundary, depending on the size of the triangle. If the latter is the case, triangle subdivision must take place.*Combustion Qualified Triangles* are the ones that are ready to take part in the first round of deformation.

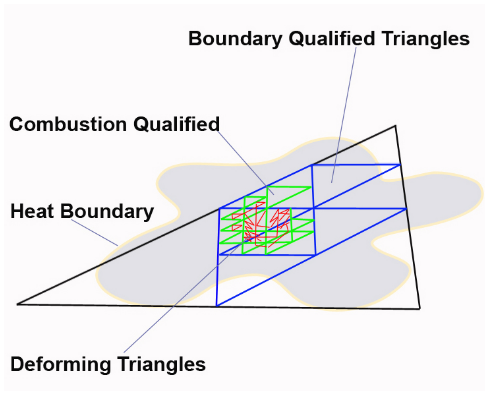

Fig. 4. Categorized Model Triangles

C. Inward Contraction Displacement

In shell models the heat-induced deformation of an object is achieved by displacement of the vertices of the model mesh (see Amarasinghe and Parberry [1]), where the position of each vertex depends on given properties such as vertex distance, gravitational force, and material index, and the internal forces work on the triangle pointing towards the direction of its vertices. However, when the model represents a solid object we must also apply *inward contraction forces* to the vertices.

In burning objects, the extending heat waves weaken the bond strength between adjacent molecules. This weakening effect falls off as a function of the distance from the heat source. As a result, surface molecules move towards the stronger bonds in order to find stable equilibrium between the acting forces. This results in contraction of the burning area of the object. Melek and Keyser [3] also noted that due to multiple internal chemical reactions at various stages of the combustion process, material may change state from solid to liquid and from liquid to gas. Both these cause reduction of the mass in affected areas of the burning object. In most cases this will cause an inward concave shape in the consumed area. To illustrate this phenomenon in a simulation we have applied what we call the *Inward Contraction Displacement Technique* to calculate the inward movement of the vertices of the Deforming Triangle. The idea of this technique is to identify for each triangle a virtual point covered by the affected polygonal boundary in distance (see Figure 5) and use this to calculate the the local inward displacement.

First we must identify the inward direction of the Combustion Qualified Triangle or the Deforming Triangle. Secondly, the distance of the virtual point must be proportional to the size of the qualified triangle. However, calculating random virtual points to meet the necessary requirements on continuously deforming polygons is not an efficient solution. Therefore, our best approach to succeed this task is to employ the face normal of the object and calculate the inverse directional coordinates. To maintain the proportional distance between the virtual point and the triangle surface, we factor the normal vector coordinate by the length of either side of the triangle (d_1 or d_2 in Figure 5). That is,

$$(X_{in}, Y_{in}, Z_{in}) = -D \cdot (X_{fn}, Y_{fn}, Z_{fn}),$$

where (X_{in}, Y_{in}, Z_{in}) is the inward contraction point and (X_{fn}, Y_{fn}, Z_{fn}) is the face normal of the targeted polygon. The distance of the either side of a polygon is represented by D. Deforming Triangles are the triangles that actually performing the deformation of the burning object. The displacement of its vertices is addressed in the following subsection.

D. Vertex displacement

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_d, Y_d, Z_d), as follows:

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

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 5 illustrates the coordinates and parameters used in these equations. The values λ and μ are the displacement amounts of each triangle due to the effect of heat on the vertex. 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 μ and λ fraction of the length along the edges (respectively BC and BA) of the triangle. The values μ and λ are displacement parameters for vertex B. They measure the amount that the bond between B and its neighboring vertices is changed by temperature.

We use a *displacement adjustment parameter* β to allow for the variation in triangle size from one model to another. The

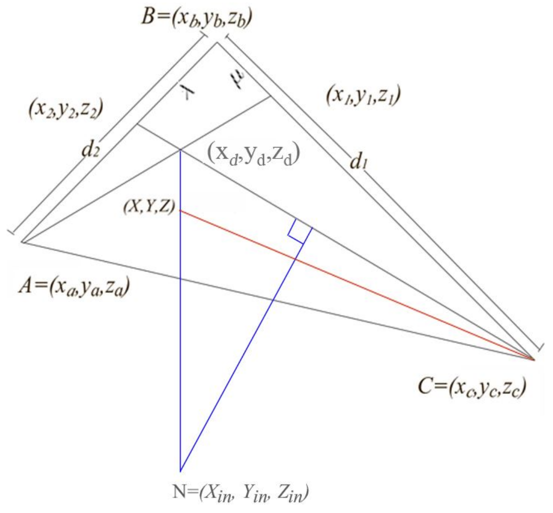

Fig. 5. The deformation coordinates of a single triangle.

designer must set this value as part of the design process. 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.

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), where L is the *flammability* of the vertex, defined as follows. Burning objects are consumed by combustion, and combustion subsides when there is nothing left to consume. We set a *flammability* value L at each vertex. This counter decreases each time vertex displacement is processed. After the flammability index reaches zero, there are no consumable resources left at the vertex. The designer sets the initial flammability index for each vertex. This gives the designer the ability to vary flammability from place to place in the model, thus mimicking the effect of having the model constructed from different physical materials such as wood or metal. The final displacement values of X, Y, Z are $X = \lambda_b \cos(\Theta)\sin(\alpha)$, $Y = \lambda_b \sin(\Theta)$, $Z = \lambda_b \cos(\Theta)\cos(\alpha)$, where

$$\begin{aligned}\alpha &= \tan^{-1}(X_d - X_{in}/Z_d - Z_{in})\\ \Theta &= \tan^{-1}(Y_d cos\alpha/Z_d - Z_{in})\end{aligned}$$

λ_b is either λ or μ depending on the corresponding distance D is d_1 or d_2.

Among all of the external forces, gravity plays a significant part in almost every physical 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: $Y = Y - \varepsilon\vec{g}$.

IV. Structural Deformation

As we described in Amarasinghe Parberry [1], the structural changes in a burning object are the result of various factors including the expansion and the weakening of the internal bonds, and the relative weights of cantilevered parts of the object. The precise calculation of these complex processes is costly. Therefore, we introduced the *block sampling method* as a computationally less expensive solution to maintaining realism while performing systematic structural change. The block sampling method divides the object into uniform blocks and treats each block as a single unit, propagating changes to neighboring blocks. The following describes the modifications needed to adapt it to solid objects.

This method starts by constructing an axially aligned bounding box around the solid object, and then decomposing 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 will use the block weight as an approximation of flammability, under the assumption that a block with more vertices contains more material, and thus will produce more flames. The difference with burning solids is that there are no surface removal techniques associated with burn level adjustments as in Amarasinghe and Parberry [1]. Furthermore, the weight changes of each block are not significant enough without the effect of level adjustment. As a solution for these concerns, we maintain a counter to monitor the time of combustion per each block. Weights of the blocks are decided according to the number of vertices factored with the counter. The empty ones of weight zero are discarded.

The parameters of each block contain the amount of the midpoint rotation, the number of vertices, the list of connected neighboring blocks, and the counter. Since all the blocks are interconnected, a change to one block may affect all of the blocks in the model. To maintain the computation complexity in low level, 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.

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.

V. Results and Optimization

We have implemented automatic level of detail (abbreviated LOD) rendering into our simulation using techniques presented earlier in Amarasinghe and Parberry [2]. Figure 6 illustrates our LOD algorithm applied to the burning of a solid block of wood. The images shown in this paper are from a CUDA implementation of our algorithm applied to different models. Since there is no strict time line for the combustion of

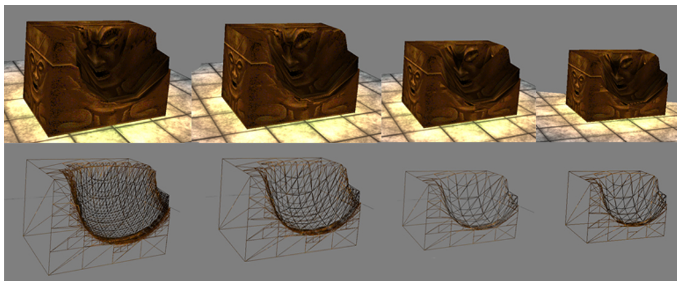

Fig. 6. Level of Detail (LOD)

the model, we can always control complexity of the simulation by limiting the number of deforming triangles at a time.

Optimization is possible since our deformation is always applied mostly to the affected areas of the object. The continuous deformation of given polygon can be controlled by parameter settings such as the *flammability* value L. In particular, the shape and size of a Deforming Triangle can be drastically changed. Overly-exaggerated deformation reduces realism. In order to maintain efficient simulation without heavy resource usage, once a Deforming Triangle's flammability value L exceeds some limit we remove the polygon from the group of Deforming Triangles and add more from the set of Combustion Qualified Triangles into the group. By following this practice we gained more control over the simulation with better performance while maintaining realism.

We used approximately 2000 fire particles and 500 smoke particles to demonstrate the visual effects. Our algorithm was implemented in CUDA on relatively modest hardware; An Intel®Core$^{\text{TM}}$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 45k 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.

VI. Conclusion

We have described a method for the real-time deformation and consumption of a solid model during combustion by procedurally generated fire, extending our previous work on shell models [1], [2]. 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.

Open problems remaining include the efficient and effective modeling of melting objects such as candles.

References

[1] 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*, 2011, pp. 256–258.

[2] ——, "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)*. EUROSIS, 2011, pp. 21–26.

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

[4] ——, "Driving object deformations from internal physical processes," in *Proc. 2007 ACM Symp. on Solid and Physical Modeling*. New York, NY, USA: ACM Press, 2007, pp. 51–59.

[5] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," *ACM SIGGRAPH Computer Graphics Quarterly*, vol. 21, no. 4, pp. 205–214, 1987.

[6] T. Sederberg and S. Parry, "Free-form deformation of solid geometric models," *ACM SIGGRAPH Computer Graphics Quarterly*, vol. 20, no. 4, pp. 151–160, 1986.

[7] E. Tadmor, R. Phillips, and M. Ortiz, "Mixed atomistic and continuum models of deformation in solids," *Langmuir*, vol. 12, no. 19, pp. 4529–4534, 1996.

[8] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," *Computer Graphics Forum*, vol. 25, no. 4, pp. 809–836, 2006.

[9] 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*, vol. 33, no. 7, pp. 1413–1449, 1992.

[10] 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*. New York, NY, USA: ACM Press, 2002, pp. 721–728.

[11] N. Rasmussen, D. Nguyen, W. Geiger, and R. Fedkiw, "Smoke simulation for large scale phenomena," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 703–707, 2003.

Dhanyu Amarasinghe

Dhanyu Amarasinghe is a PhD candidate in the Department of Computer Science and Engineering at the University of North Texas. His research topic is the real-time procedural consumption and distortion of 3D game objects under combustion. His PhD adviser is Ian Parberry.

Ian Parberry

Ian Parberry is a Professor in the Department of Computer Science and Engineering at the University of North Texas. With over 3 decades of experience in research and education, he is a pioneer of game programming in academia. He is the author of seven books, four of them on game programming, and more than 70 articles on a wide range of computing subjects including algorithms, complexity theory, parallel computing, neural networks, and game programming. He is on the Editorial Boards of the Journal of Computer Game Design and Development, IEEE Transactions On Computational Intelligence and AI In Games, and serves on the Society for the Advancement of the Science of Digital Games, which organizes the Annual Foundations of Digital Games conference.