

An Interactive Simulation Framework for Burning Objects

Zeki Melek John Keyser

Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
E-mail: z0m8905@cs.tamu.edu
Phone: (979) 845-5007
Fax: (979) 847-8578

Technical Report 2005-3-1

March 8, 2005

Abstract

We present a simulation framework to integrate several aspects of the combustion and burning process in a unified and modular manner. A simple three gas flame model is used to simulate a combustion process, while air motion is simulated as a single moving fluid. Solid objects inside the simulation domain can catch fire and start burning. Heat information is transferred from the fluid simulator to a solid simulator, while the solid simulator injects fuel into the fluid simulation. We also present a simple yet effective method for modeling of object decomposition under combustion using level set methods. The interaction between modules is presented as well as a discussion of fluid-solid coupling. All simulation modules run together at interactive rates, enabling the user to tweak the simulation parameters and setup for desired behavior¹.

1 Introduction

The simulation and visualization of natural phenomena has been an area of great interest within the computer graphics community. Simulation of water, smoke, and fire all have potential uses in entertainment and training applications. Recently, a number of advances have made realistic interactive modeling of gaseous phenomena possible. These coarse-grid fluid-dynamic equation solvers are capable of approximating fine-scale swirling motion of air at very fast rates. Recent models have begun to simulate fire based on these fluid simulations. While most fire models implement some fire related phenomena,

¹This work is supported in part by NSF grant CCR-0220047.

there have not been any offering a framework that addresses all the major fire-related phenomena - flame, combustion, and object decomposition.



Figure 1. Burning logs during interactive simulation

1.1 Main Results

We present a framework combining different modules to simulate, for the first time, many aspects of the burning process in a unified simulation. We also present a method for simplified modeling and visualization of fire that is easy to implement and works at interactive rates. We apply state of the art coarse grid fluid dynamics equation solvers to model the motion of fuel, air, and exhaust gases in a unified system. The heat produced by combustion affects the motion of the air within the computational domain, which in turn affects the shape and motion of the flame. In addition, this heat transport allows us to simulate self-ignition of objects away from the flame itself. Burning objects decompose with a decomposition defined as a moving boundary problem, modeled using level sets. This allows us to easily model the complex topology-changing decomposition process. Finally, the ash pieces, split from the burning objects, are simulated as rigid bodies, even affecting the air motion.



Figure 2. Decomposition in action

2 Background and Prior Work

2.1 The Combustion Process

Despite decades of study, combustion remains an active topic of research. In combustion, fuel is pre-heated by ignition, and the gases that are mixed with air (the oxidant) in “good” concentration combust, giving an exothermic reaction. For diffusion flames, the fuel gas mixes with the oxidizing gas, creating a mixture zone[6]. The pure fuel is heated as it moves from its source at the center of the flame toward the reaction zone. As the reaction zone is approached, the increasing amount of oxidizing gas allows the chemical combustion reaction to occur. Combustion continues outward throughout the reaction zone until the stoichiometric contour (flame front) is passed and the reaction is completed. The heat generated during the reaction keeps the reaction zone at or above the combustible temperature, triggering more combustion, thus creating more heat and combustion byproducts.

Once ignited, a diffusion flame stabilizes. Flames tend to be carried upward due to convection of the air that has been heated by the combustion process. The heat produced by the flame can also radiate to vaporize combustible products from nearby matter through chemical decomposition. This process is called pyrolysis [3]. Thus, when an object (such as a piece of wood) burns, the flames are formed from combustible portions of the object being vaporized and then oxidizing in the region just beyond the surface.

2.2 Flame Modeling in Computer Graphics

Within the computer graphics community, early models of fire include Reeves's particles [24] and Perlin's procedural noise function [22]. Inakage[12] uses a physical model to emit light in the regions of combustion for still images. Chiba et al.[5] and Takahashi et al.[29] both use a grid based representation of the space, and flames themselves are modeled by placing geometric primitives around particle trajectories.

Recent models include Lamorlette and Foster[15], combining particles with a flame skeleton, and Nguyen et al.[19], 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. The actual combustion process is not modeled, but an expansion term is introduced to add fine scale detail without using any randomized term. Another recent work by Feldman et al.[9] uses incompressible fluid equations to model suspended particle explosions. By enabling $\nabla \cdot u > 0$, they model an expansive flow without the need to simulate compressible fluids. Ihm et al. [11] used this result to model expansion of chemical reactions.

NIST has developed the CFAST fire simulator [14], giving an accurate simulation of the impact of fire and its byproducts. Bukowski and Séquin [4] have integrated CFAST and the Berkeley Architectural Walkthru program, making the results of CFAST more understandable. A newer simulator implemented by NIST is the FDS [16], which has a very similar framework to what we use to model combustion. Among the major differences here are our fluid simulation and our modeling of the decomposition of the solid objects. Also, while CFAST and FDS are more accurate models, they are not suitable for interactive simulation.

The spread of fire has been another area of research. Perry and Picard[23] represent the flame front using particles, adding new particles as the front expands. Stam and Fiume [27] use a map to define the amount of fuel and temperature on every point on the object, and use turbulent wind fields and warped blobs. Beaudin et al.[2] use a similar but more accurate flame front technique to model the spread, and guarantee that the boundary always lies on the object. Wei et al. use the Lattice Boltzmann Model to simulate fire at interactive rates[30], and add visual detail by using textured splats. They recently [31] extended their model to track flame fronts over volumetric implicit solids. Jones [13] introduced a method that accurately models both thermal flow and the latent heat during the phase change to model melting solids.

Level set methods, which we use in our decomposition model and were introduced by Osher and Sethian[21], are a simple yet powerful approach for computing moving interfaces. They have recently started to gain popularity within the computer graphics community. Many applications, including incompressible and compressible flow problems and a recent thin flame model [19], are solved using level set methods. More information on level set methods and their applications in computer graphics can be found in recent publications [25, 20, 7].

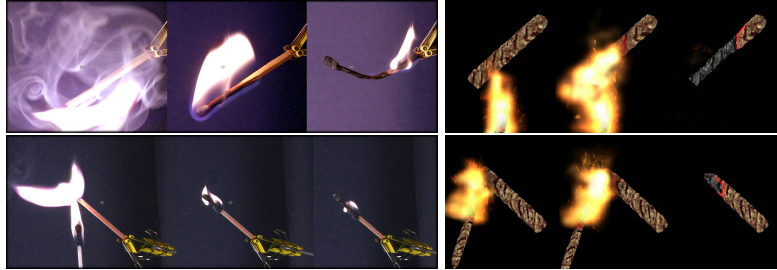


Figure 3. Real vs simulated matches in different orientations

3 Simulation Framework

Our simulation framework consists of two main modules, along with a synchronization and data exchange interface. The two main components are a fire/flame simulation and a solid object simulation; each of these is composed of several sub-modules dealing with different phenomena. The two simulations are coupled together by: pyrolysis—transferring fuel from the solid representation to the fluid representation, heat interpolation—transferring heat information from the fluid simulator to the solid representation, and external forces—changes in air motion due to object movement and the reverse. We will examine each module including its requirements, its interface with the other modules, and our sample implementation.

4 Fire Simulation

The fire module is responsible for air motion, gas distribution, and heat generation. It models the combustion process, which generates heat and drives the air motion.

4.1 Requirements

- It should track distribution of different gases.
- Motion of the hot air needs to be modeled. It should also keep track of which parts of the simulation domain are occupied by solid objects.
- The combustion process, which consumes fuel gas and generates smoke and heat, should be modeled.
- Heat transfer should be modeled explicitly.

4.2 Input/Output to the module

- Fuel sources, including solids in pyrolysis, provide fuel gas to the module.
- The interface with the solid object module also provides as input occupancy information as a form of filled/unfilled voxels, which are taken into account during air motion.
- Fire simulation provides heat information to the decomposition process.
- Moving air can affect, and could be affected by, moving objects.

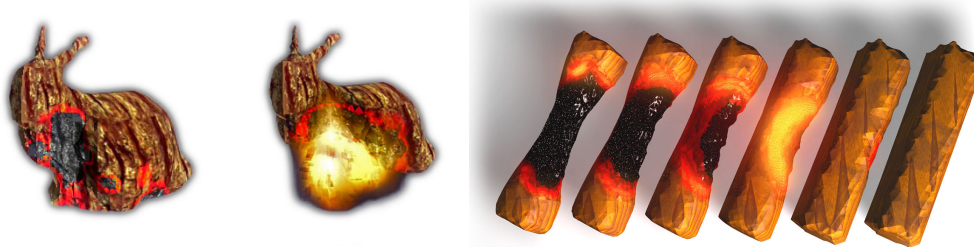


Figure 4. (a) Burning Bunnies during the interactive simulation (b) burning log rendered offline

4.3 Implementation

4.3.1 Hot Air Motion

In our simple flame model, we transport the fuel and exhaust gases with the motion of the air, creating a dynamic three-gas system. Heat is also transported with the flow of the air, enabling us to model heat distribution inside the computational domain. All properties are stored and computed on a 3D grid.

To model the air flow, we use a simple modification of the Stable Fluids [26, 8] approach. We assume there is only one moving gas (air), which is inviscid, incompressible, and constant density. The fluid motion (air) solution is applied to advect three scalar quantities: fuel gas, exhaust gas (including smoke), and heat. The portion of each cell not occupied by fuel gas or exhaust gas is assumed to be oxidizer. Some cells (such as those containing objects to be burned) are marked as filled.

We apply additional forces to more accurately model the gas motion. Fuel and smoke are given negative buoyancy, creating some downward motion. An upward force is applied to cells that are hotter than the surrounding cells. Motion of the solid objects in the scene introduce external forces into the fluid simulator (e.g. as with a fan moving air). Finally, a vorticity confinement term [28, 8] is used to compensate for the numerical dissipation in the system.

4.3.2 Combustion

We simulate the combustion process by combining fuel and oxidizer in a cell, creating additional exhaust gas in its place. If the heat in the cell is beyond the combustion threshold, we decrease the level of fuel gas, increase the level of exhaust gas, and introduce additional heat into the cell. Different types of fuel can be simulated by adjusting the ratios of gas used and produced (the stoichiometric mixture) and the heat released [17, 18]. Recently Ihm et al.[11] generalized this idea for modeling of any chemical reaction.

We define the amount of oxygen in a cell indirectly, by specifying the fuel and exhaust gas in the cell, and assuming that the remainder is oxidizing air. This simplification is not necessarily physically correct, but it enables us to use a single-fluid fuel solver and to model oxygen use and self-extinguishing flames by keeping track of only two gases. Although air is not all oxygen, this is easily accounted for by adjusting the stoichiometric mixture appropriately.

Filled/unfilled information is initialized by checking whether cell centers lie inside solid objects, and the air motion solver is constrained accordingly. Solid objects in pyrolysis provide fuel gas to the module, similar to any generic fuel source.

4.3.3 Heat Transfer

In our system we model heat transfer 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. Also, this model is flexible enough that different simulators can be used for any of the three stages, enabling us to work with slower but more accurate simulations of any portion, if desired.

Basic heat convection in air is handled using semi-Lagrangian advection, which simulates moving air currents carrying heat. On the other hand, radiative heat transfer is approximated as a diffusion process using implicit integration, enabling us to distribute the heat coming from the combustion process. The semi-Lagrangian advection scheme and implicit integration for the diffusion steps follow the same pattern as in the Stable Fluids approach[26]. The heat transfer between the solid and the air is handled during the pyrolysis step which acts as the connection between two simulation modules (see section 5.3.2).

5 Solid Objects

The solid objects module consists of heat transfer, object decomposition, and a rigid body solver. The decomposition module interfaces to the fire simulator to obtain heat near the object, and diffuses this heat into the object. If any part of the solid is in pyrolysis, it outputs fuel gas to the fire simulation, by converting solid fuel held in the combustible solid to gaseous fuel in the fire simulator.

5.1 Requirements

- Information must be provided about which parts of the scene are occupied by the objects.
- Heat triggers pyrolysis, which generates fuel gas.
- During pyrolysis, solid objects lose volume as they decompose.
- Objects should undergo rigid body motion.

5.2 Input/Output to the module

- Heat information should be provided from an external source.
- Air motion can affect rigid body motion, hence air motion near the solid should be provided.
- During pyrolysis fuel gas is injected into the fire simulation module with some direction.
- The motion of the solid object can affect the air motion, the motion vectors along the solid air boundary should be provided to the air motion module.

5.3 Implementation

Individual solid objects are represented with two complementary representations that change as the objects burn and decompose, one indicating the amount of fuel in each cell, and one indicating the total material in a cell (fuel and residual ash) [1].

5.3.1 Pyrolysis

Once the solid 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.

$$\frac{\partial}{\partial t}d_g = r_s \frac{\partial}{\partial t}V \quad (1)$$

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 section 5.3.3. 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.

5.3.2 Heat Transfer

The heat transfer between the solid-air interface is done during the pyrolysis step. First, each solid-air boundary voxel is marked in the solid domain. For each boundary cell, heat exchange takes place using the heat differential between the filled cell and the unfilled neighbors. Although it is possible to use a more accurate heat radiation simulator, this simplified model works fine for most cases and enables us to model convection currents from a heated solid. If additional accuracy is required, a direct radiant heat term can be incorporated by using a (likely non-interactive) radiosity calculation for the heat coming directly from the visible part of the flame. Heat transfer inside solids is modeled by simulating diffusion using the heat equation:

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

k is a constant based on density, thermal conductivity, and specific heat of the material. Simulating heat diffusion is relatively simple, and an implicit integrator similar to that for heat diffusion in air (as above) can be used. 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 k in our simulation.

5.3.3 Decomposition

Our model for decomposition is the first model for defining decomposing objects under combustion in the CG literature. The decomposition of the burning solid is modeled as a moving boundary in the distance field representation of the solid using level set methods [21, 20]. The decomposing solid is defined implicitly as

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

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

We define the decomposition of the object with two components: d (the amount) and \vec{D} (the direction) of decomposition. The amount of decomposition, d , is defined proportional to the amount of solid fuel released as fuel gas in a time step.

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

V is the solid fuel representation and k_1 ($0 \leq k_1 \leq 1$) 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. Our 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 the fuel in the cell is gone.

$$\frac{\partial}{\partial t} V(\vec{x}) = dt \begin{cases} k_2, & T(\vec{x}) \geq T_{sp} \text{ and } V(\vec{x}) \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

$T(\vec{x})$ is the temperature at \vec{x} . k_2 ($-1 \leq k_2 \leq 0$) controls how fast fuel gas is released. Together with the self-pyrolysis temperature threshold (T_{sp}), it controls how volatile the solid will be. For the direction, we used for our decomposition direction the normal from the implicit model, which ensures that the boundary continually moves inward.

$$\vec{D}(\vec{x}) = -\vec{N}(\vec{x}) = -\frac{\nabla\phi(\vec{x})}{|\nabla\phi(\vec{x})|} \quad (7)$$

$$\frac{\partial\phi}{\partial t} = -k_1 \frac{\partial}{\partial t} V(\vec{x}) \frac{\nabla\phi(\vec{x})}{|\nabla\phi(\vec{x})|} \quad (8)$$

Instead of using the normal, we can define the decomposition direction using the gradient of the fuel consumption, which is defined as a monotonically decreasing function in this case. Our tests show that this does not give visibly different results than using a simple normal (for an in-depth analysis, we encourage the reader refer to [1]).

5.3.4 Rigid body simulation

As objects decompose, thin or disconnected pieces of ash can be created. Since we separate the object decomposition simulation from the flame simulation, we can simulate rigid body motion of those separated pieces in parallel with the other simulations. To detect disconnected pieces, the polygons created from the implicit representation for visualization are examined. When pieces are determined to be separate, the volume and distance fields split accordingly, creating two or more separate solid objects. Collisions are determined by comparing objects pairwise using the distance field of the objects (for a similar use, see Guendelman et al. [10]). Impact forces cause colliding objects to move apart.

Air motion from the fluid simulation is applied to the object by adding the forces from the fluid grid, scaled for object mass. Although the effect is very small for large objects, it can be useful to model the

floating of very small ash pieces. The object motion also affects air motion; the movement of the solid boundary applies an external force in the fluid simulator. This simple air-solid interaction is fast, fits into our framework easily, and gives pleasing results.

6 Module Interface and Execution

By representing the solids on different grids from the fluid simulator, the solid module can be executed as a separate (but synchronized) thread. It is also possible to handle multiple solids with different material properties simultaneously as multiple threads. The simulations of the air flow and of the burning solid can be run at different rates. The two simulations are joined through the pyrolysis step.

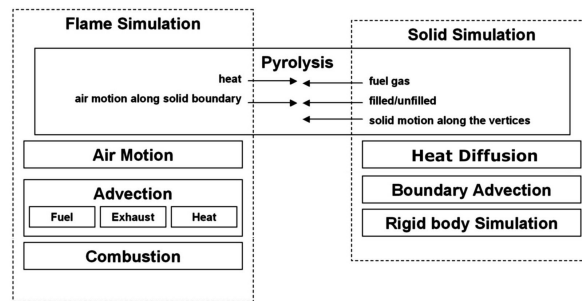


Figure 5. Integration of simulation modules

- A pyrolysis time step begins with the initialization of filled/unfilled information from the solids into the flame module. The solid grids are examined to find whether the point at the center of a fluid grid cell is inside any solid. If so, that fluid grid cell is marked filled, restricting air flow in the fluid simulation.
- The air-solid boundary cells are traced and the heat differential between air and solid is used to exchange heat. Heat diffusion is done for the air and solids separately, using different diffusion rates.
- The next step is transferring forces in the fluid motion field. The fluid flow exerts forces on moving solids, and moving solids generate forces within the fluid field.
- The fire module takes one step in the fluid dynamic simulation, using the forces and filled/unfilled information set in the previous steps. The resulting air motion field is used to advect fuel and smoke gases, as well as heat. The resulting gas distribution combined with the heat is used for the combustion process. In each cell, fuel gas (and oxidizer) is consumed, and smoke and heat are formed.
- Solid objects are simulated within their own space. Each cell is checked for whether it can undergo pyrolysis or not. The pyrolysis cells convert solid fuel into fuel gas as a function of heat. This fuel gas is passed to the flame simulator. The fuel conversion differential drives the object decomposition process, as described earlier. Decomposing objects are checked to see if they have separated into two or more parts, and rigid body motion is determined.

One should note that the modules presented in this paper are not necessarily the best possible models. More or less accurate modules could replace any module in the system, allowing a tradeoff in time and accuracy, as long as they support the input/output interfaces described.

7 Parameters and Control

There are many parameters that can be used to define the flame motion as well as solid decomposition behavior; many are referenced in the previous sections. Among the key parameters are:

- *Flame control*: Oxygen use rate (stoichiometric mixture) controls the size of the reaction zone as well as the survival of the flame under low oxygen conditions. Output heat determines how easily a flame becomes self-sustaining and spreads.
- *Solid decomposition*: Pyrolysis temperature of the material makes a solid volatile or non-volatile. Diffusion rate controls the spread of the flame front on the solid. The fuel/residue rate controls how quickly and how far the object decomposes. The fuel gas multiplier increases/decreases the fuel gas coming from the burning solid, and thus the size of flames.

This parameter set allows us to simulate a wide and complex variety of flame and burning effects, which were not possible in other CG fire simulators. We can model simple self-sustaining flames, such as from a candle, or unstable flames where the heat output may not be enough to sustain the flame. We can simulate a log which is burning slowly and steadily or one bursting into flames. Matches will burn differently depending on their orientation (Figure 3). Heat transfer lets us ignite objects even at a distance from the flame, or create convection currents from local heat sources, even with no flame. We can simulate effects such as a room filling with gas, then igniting, resulting in a fireball (it is not an actual explosion, because of the incompressibility assumption on the fluid solver). We can also model the extinguishing of a flame by using up all available oxygen in a closed environment. Note that the work presented in this paper is the first work addressing most of those different physical phenomena in one unified system.

8 Simplifications and Analysis

Even though our framework for simulation is general, the individual modules we have proposed make a number of simplifying assumptions, which lead to some shortcomings. We describe below some of our key assumptions, simplifications, and shortcomings, along with the reason we accept them.

- *Single fluid model*. Although air, fuel gas, and exhaust gases will have different properties, we choose to model the system with a single fluid. We therefore do not capture properties such as the mixing of fluids of different viscosity, which hurts our ability to capture some fine-scale detail. However, our model still captures the larger-scale motion of heated air, and is significantly cheaper computationally. We also believe that the fine detail is more of a rendering issue than a key to simulation, and such details can be better added as a visual postprocess.
- *Constant density air*. Our assumption that the sum of the air, fuel, and exhaust gas in a cell is constant is somewhat unrealistic. In particular, the effects of combustion, such as the pressure

generated and the resulting density changes are not directly modeled (we do simulate some effects via the buoyancy forces).

- *Expansion of the reaction zone.* One specific property of the combustion reaction is expansion of the gas, which gives a more round shape to the flames. This could be added to our model by introducing outward forces at the reaction zone, similar to Nguyen et al. [19], or setting $\nabla \cdot u$ to be larger than zero at the reaction zone, similar to Feldman et al. [9].
- *Decoupled heat transfer.* Separating heat transfer into three stages, rather than using a unified simulation is somewhat inaccurate. However, the inaccuracies do not appear significant, and the decoupling allows us to use different solvers of varying accuracy for the portions of the simulation we care about.
- *Simplified decomposition model.* The actual physical decomposition process of burning objects is quite complicated and would involve detailed modeling of the solid structure at a very low level. We believe that our moving boundary captures the primary large-scale effects, including the production of residual ash.
- *Simplified solid simulation.* We have only modeled rigid body motions (letting solids and ash fall and collide), and have not modeled internal stresses and strains on the objects.
- *Coarse resolution simulation for interactivity.* One might argue the need for a low resolution physically based simulation, since there are some production quality simulation models around ([19]). These other methods, though, are non-interactive, miss key parts of the entire burning process, and are difficult to fine-tune. We present a framework to model many aspects of fire related phenomena together in one single simulation framework, and the coarse simulation used here is *one* of the possible implementations. This particular implementation runs at interactive rates, enabling the user to fine tune the simulation behavior quickly, giving plausible results. As mentioned below, having the interactive method can allow us to more easily control behavior in more complex simulations.

9 Visualization

We have implemented the method presented above and tested it on a number of examples. Our simulations are run on the CPU of an ordinary P4 1.8GHz machine with 512MB of RAM and a 32MB GeForce2 card. We are able to simulate a flame interactively (~ 5 fps) for an environment defined on a 20^3 simulation grid. This simulation includes the simulation of air motion, burning and decomposition for one solid, and our simple visualization. The grids used to describe the solid are 30^3 for the distance field representation and 20^3 for the fuel set representation.

Our simple visualization system is OpenGL based. It is intended primarily to demonstrate the simulation methods, and is *not* intended to generate production-quality images. We emit some particles from the flame front to breakup the gaseous look of the flames, yet they do not participate in the simulation. We use multiple textures to display wood in various stages of burning, switching from one to another as the burning progresses. Here one should note that in our simulation we do not track the flame front geometrically, rather it is the outcome of our underlying simulation.

Our visualization is far from optimal, yet this paper is not a rendering paper. We have presented only a basic interactive rendering approach - better methods that make use of current graphics hardware (such as fragment shaders, and 3D textures) could give better visualization of our interactive simulation, still at interactive rates. Our video shows some of the cases our system can handle using faster-than-life decomposition rates.

10 Conclusion

We have presented a method that for the first time integrates a fluid-based flame simulator, combustion model, heat transfer model, solid pyrolysis model, solid decomposition model, and rigid body simulator into a common framework. We have presented basic simulation approaches for each of these modules, and described the interface mechanisms between them. Although we have made a number of simplifying assumptions, our system is capable of modeling basic flame and decomposition behavior. This occurs at interactive frame rates even on older PC hardware. The simulation model we have presented is flexible, allowing for easy replacement of simulators in any one part of the model, fairly independently of the other portions of the model. The parameters available in our model give a great deal of control over both the way objects burn, and the way the resulting flames behave, and we can achieve a variety of complex physically-based effects at interactive rates.

There are several avenues open for future work, particularly for any one module. One key area that we believe would improve our framework is stress calculations. We are investigating ways to include limited internal stress calculations into the solid decomposition process to simulate objects fracturing as they burn up. Finally, we are developing a method to use our interactive simulation to constrain higher-resolution offline simulation. This opens two major uses for the interactive simulation: quickly previewing the simulation behavior, and setting up initial conditions for the interesting part of the simulation (and thus “fast-forwarding” the simulation).

References

- [1] Anonymous. Multi-representation interaction for physically based modeling. ACM Symposium on Solid and Physical Modeling, 2005. Currently in submission.
- [2] P. Beaudin, S. Parquet, and P. Poulin. Realistic and controllable fire simulation. *Proc. of Graphics Interface '01*, pages 159–166, 2001.
- [3] R. Borghi, M. Destriau, and G. D. Soete. *Combustion and flames, chemical and physical principles*. Editions Technip, Paris, 1995.
- [4] R. Bukowski and C. Séquin. Interactive simulation of fire in virtual building environments. *Proc. of ACM SIGGRAPH '97*, 1997.
- [5] N. Chiba, K. Muraoka, and M. Miura. Two dimensional visual simulation of flames, smoke and the spread of fire. *The Journal of Visualization and Computer Animation*, (5(1)):37–54, 1994.
- [6] C. Detienne. *Physical Development of Natural and Criminal Fires*. Charles C. Thomas, 1994.

- [7] R. Fedkiw. Simulating natural phenomena for computer graphics. *Geometric Level Sets in Imaging, Vision and Graphics*, edited by S. Osher and N. Paragios, 2002.
- [8] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. *Proc. of ACM SIGGRAPH '01*, pages 15–22, 2001.
- [9] B. E. Feldman, J. F. O'Brien, and O. Arikan. Animating suspended particle explosions. In *Proc. of ACM SIGGRAPH '03*, pages 708–715, aug 2003.
- [10] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, 2003.
- [11] I. Ihm, B. Kang, and D. Cha. Animation of reactive gaseous fluids through chemical kinetics. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 203–212. ACM Press, 2004.
- [12] M. Inakage. A simple model of flames. *Proc. of CG international*, pages 71–81, 1990.
- [13] M. W. Jones. Melting objects. *Journal of WSCG*, 11(2):247–254, 2003.
- [14] 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.
- [15] A. Lamorlette and N. Foster. Structural modeling of natural flames. *Proc. of ACM SIGGRAPH 02*, pages 729–735, 2002.
- [16] 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.
- [17] Z. Melek and J. Keyser. Interactive simulation of fire. *Proc. of Pacific Graphics '02*, pages 431–432, 2002.
- [18] Z. Melek and J. Keyser. Interactive simulation of burning objects. *Proc. of Pacific Graphics '03*, pages 462–466, 2003.
- [19] D. Nguyen, R. Fedkiw, and H. W. Jensen. Physically based modeling and animation of fire. *Proc. of ACM SIGGRAPH '02*, pages 721–728, 2002.
- [20] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag, 2002.
- [21] 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.
- [22] K. H. Perlin. An image synthesizer. *Proc. of ACM SIGGRAPH '85*, (19(3)):287–296, 1985.
- [23] 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.
- [24] W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, (2(2)), 1983.

- [25] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.
- [26] J. Stam. Stable fluids. *Proc. of ACM SIGGRAPH '99*, pages 121–128, 1999.
- [27] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion process. *Proc. of ACM SIGGRAPH '95*, pages 129–136, 1995.
- [28] J. Steinhoff and D. Underhill. Modification of the euler equations for vorticity confinement: Application to the computation of interacting vortex rings. *Physics of Fluids*, (6):2738–2744, 1994.
- [29] J. Takahashi, H. Takahashi, and N. Chiba. Image synthesis of flickering scenes including simulated flames. *IEICE Transactions on Information Systems*, (E80-D(11)):1102–1108, 1997.
- [30] 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.
- [31] Z. F. A. K. Ye Zhao, Xiaoming Wei and H. Qin. Voxels on fire. *IEEE Visualization*, 2003.