Physically Based Sound for Computer Animation and Virtual Environments

Fire

Doug James
Stanford University
“Fire” Reference:

These equations indicate that both the velocity and the pressure are significant in the simulation of fire. Substituting the solid fuel plus a correction that accounts for the expansion.

In figure 8, we simulate a campfire using two cylindrically shaped nozzles, each emitting a stream of flames. The velocities are defined at each voxel adjacent to the image plane.

The images of the flames arising from the explosion are particularly clear with the use of high-resolution simulations. The motion of the implicit surface is defined through the use of particle systems, which are computationally efficient and produce visually appealing blue cores.

The particle systems can be used to simulate a variety of fire and smoke effects, as shown in the figure. The results are compared to traditional methods, and the particle systems are shown to be significantly faster and more flexible.

The renderings in the figure show different particle systems, with the third order Lagrangian equations coupled to the Navier-Stokes equations for smooth and high-resolution simulations. The third order Lagrangian equations were used for the simulations of a burst near a wall, where the simulation of a burst near a wall only uses a single particle.

The results are presented in a table format, with two columns and two rows, showing the particle systems and the corresponding simulation times.

The table shows that the particle systems are significantly faster than traditional methods, with a reduction in simulation time of up to 6x.

The particle systems are also shown to be more flexible, allowing for the simulation of complex fire effects, such as the emission of a flame from a dragon.

The simulations are computationally intensive, but the use of multiple independent GPUs allows for the parallel processing of the simulations, significantly reducing the processing time.

The particle systems are also shown to be more visually appealing than traditional methods, with the blue cores indicating the movement of the particles. This is particularly useful for simulations of flames and fire, where the visual characteristics are critical.

The particle systems are shown to be particularly useful for simulating a wider range of fire effects, including those that are difficult to simulate with traditional methods. The particle systems are shown to be particularly useful for the simulation of smoke and water.

The simulations are shown to be particularly useful for visual effects, with the use of particle systems allowing for the simulation of a wide range of visual effects, including the emission of a flame from a dragon.

The simulations are shown to be particularly useful for the simulation of smoke and water, with the particle systems allowing for the simulation of a wider range of visual effects, including the emission of a flame from a dragon.
Sound Synthesis

Aerodynamic Sound

[Dobashi et al. 2003]

[Dobashi et al. 2004]
Background

• Combustion sound components [Chrighton et al. 1992; Poinsot and Veynante 2005]

• Aerodynamic noise (eg. [Dobashi et al. 2003])
  • Resulting from turbulent flow

• “Direct combustion noise”
  • Produced by density fluctuations resulting from heat release
  • Dominant source of combustion sound [Ihme et al. 2009]
Background

• Combustion sound wave equation

\[
\frac{1}{c_0^2} \frac{\partial^2 p}{\partial t^2} - \nabla^2 p = -C \frac{\partial q}{\partial t}
\]

• Convert to integral equation

\[
p(x, t) = \frac{C}{4\pi} \frac{\partial}{\partial t} \int_{\mathbb{R}^3} \frac{1}{r} q(y, t - r/c_0) \, d^3y
\]

Combustion heat release rate

\[
sound \approx \frac{d}{dt} \int_{\mathbb{R}^3} q(y, t) \, d^3y
\]
Background

\[
\text{sound} \approx \frac{d}{dt} \int_{\mathbb{R}^3} q(y, t) d^3y
\]

• **Problem:** Flame solvers from the graphics community do not realistically model heat release

• Heat release modeled artistically, via simple functions, etc.

• **Goal:** express heat release in terms of quantities we have access to
Flame Recording
1/10th speed
Flame Recording
Problem Statement

• Time stepping 3D fluid simulation to resolve all frequency content impractical

• Flame solvers used in the graphics community do not model complex combustion chemistry

• Increasing temporal resolution would not be effective
Overview

Spectral Bandwidth Extension

- Input Signal $p(t)$
- Window Function $w(t)$
- Input Amplitude $p(t)$
- Noise Spectrum $S^{-3/2}$
- IFFT

Windowed Signal $p_w(t)$
Windowed Noise $N_w(t)$

OR

Sound Texture Synthesis

$p(t)$ $t$
Low-frequency Sound Model

\[ \text{sound} \approx \frac{d}{dt} \int_{\mathbb{R}^3} q(y, t) d^3 y \]

- **Problem:** Flame solvers from the graphics community do not realistically model heat release

- Heat release modeled artistically, via simple functions, etc.

- **Goal:** express heat release in terms of quantities we have access to
Low-frequency Sound Model
Modeling Heat Release

- Premixed flame assumption:
  - Reactants mixed prior to combustion
  - Combustion occurs rapidly when ignition temperature is reached

**Result:** heat release confined to “flame front” separating burnt and unburnt gasses
Low-frequency Sound Model

Modeling Heat Release

\[
\text{sound} \approx \frac{d}{dt} \int_{\mathbb{R}^3} q(y, t) d^3 y
\]

\[
\text{sound} \approx \frac{d}{dt} \int_{S(t)} q \, dS
\]
Low-frequency Sound Model
Modeling Heat Release

- Approximate heat release with velocity flux [Strahle 1972; Clavin and Siggia 1991; Chrighton et al. 1992]

\[
sound \approx \frac{d}{dt} \int_{S(t)} q dS \approx \frac{d}{dt} \int_{S(t)} u \cdot n dS
\]
Low-frequency Sound Model

Flame Front Surface

\[
\text{sound} \approx \frac{d}{dt} \int_{S(t)} u \cdot n \, dS
\]

- “Blue core” flame solvers [Nguyen et al. 2002; Hong et al. 2007]
- Explicitly model flame front surface level set
- Other solvers which explicitly model fuel (e.g., Houdini’s Pyro FX solver)
- Track rate at which fuel is consumed at each voxel and build an iso-surface
Low-frequency Sound Model

Results

\[ \text{sound} \approx \frac{d}{dt} \int_{S(t)} u \cdot n \, dS = \frac{d}{dt} I(t) \]

Compute

\[ I(t_0), \quad I(t_1), \quad I(t_2), \ldots \]

Simulation time steps
Overview

Sound Texture Synthesis
Sound Texture Synthesis

Motivation

• Noise-based bandwidth extension introduces higher frequencies
  • Missing temporal structure present in real flame sounds
  • No “style control”

• **Idea:** Using low-frequency results as a guide, introduce high-frequency detail based on recorded flame sounds

Related work on texture synthesis:

• Images: [Heeger and Bergen 1995; Efros and Leung 1999; Wei and Levoy 2000; Efros and Freeman 2001] (and many others)

• Sounds: [Dubnov et al. 2002; Strobl et al. 2006; McDermott et al 2009; Marelli et al. 2010]
Texture Synthesis Basics
[Wei and Levoy 2000]

Figure 3: The top row shows the Gaussian pyramid of the input texture, with successive lower resolution images toward the

Analysis:

Step 1

The example texture image

gathered from the neighborhood vectors,

Build a Gaussian pyramid

\( F(\mathbf{x}, \mathbf{y}, i) \)

\( s \)

\( N \)

\( T \)

\( r \)

\( w \)

\( \alpha \)

\( \beta \)

\( \gamma \)

\( \delta \)

\( \varepsilon \)

\( \zeta \)

\( \eta \)

\( \theta \)

\( \iota \)

\( \kappa \)

\( \lambda \)

\( \mu \)

\( \nu \)

\( \xi \)

\( \omicron \)

\( \pi \)

\( \rho \)

\( \sigma \)

\( \tau \)

\( \upsilon \)

\( \phi \)

\( \chi \)

\( \psi \)

\( \omega \)

\( \alpha \)

\( \beta \)

\( \gamma \)

\( \delta \)

\( \varepsilon \)

\( \zeta \)

\( \eta \)

\( \theta \)

\( \iota \)

\( \kappa \)

\( \lambda \)

\( \mu \)

\( \nu \)

\( \xi \)

\( \omicron \)

\( \pi \)

\( \rho \)

\( \sigma \)

\( \tau \)

\( \upsilon \)

\( \phi \)

\( \chi \)

\( \psi \)

\( \omega \)

\( \alpha \)

\( \beta \)

\( \gamma \)

\( \delta \)

\( \varepsilon \)

\( \zeta \)

\( \eta \)

\( \theta \)

\( \iota \)

\( \kappa \)

\( \lambda \)

\( \mu \)

\( \nu \)

\( \xi \)

\( \omicron \)

\( \pi \)

\( \rho \)

\( \sigma \)

\( \tau \)

\( \upsilon \)

\( \phi \)

\( \chi \)

\( \psi \)

\( \omega \)

\( \alpha \)

\( \beta \)

\( \gamma \)

\( \delta \)

\( \varepsilon \)

\( \zeta \)

\( \eta \)

\( \theta \)

\( \iota \)

\( \kappa \)

\( \lambda \)

\( \mu \)

\( \nu \)

\( \xi \)

\( \omicron \)

\( \pi \)

\( \rho \)

\( \sigma \)

\( \tau \)

\( \upsilon \)

\( \phi \)

\( \chi \)

\( \psi \)

\( \omega \)

\( \alpha \)

\( \beta \)

\( \gamma \)

\( \delta \)

\( \varepsilon \)

\( \zeta \)

\( \eta \)

\( \theta \)

\( \iota \)

\( \kappa \)

\( \lambda \)

\( \mu \)

\( \nu \)

\( \xi \)

\( \omicron \)

\( \pi \)

\( \rho \)

\( \sigma \)

\( \tau \)

\( \upsilon \)

\( \phi \)

\( \chi \)

\( \psi \)

\( \omega \)

\( \alpha \)

\( \beta \)

\( \gamma \)

\( \delta \)

\( \varepsilon \)

\( \zeta \)

\( \eta \)

\( \theta \)

\( \iota \)

\( \kappa \)

\( \lambda \)

\( \mu \)

\( \nu \)

\( \xi \)

\( \omicron \)

\( \pi \)

\( \rho \)

\( \sigma \)

\( \tau \)

\( \upsilon \)

\( \phi \)

\( \chi \)

\( \psi \)

\( \omega \)

\( \alpha \)

\( \beta \)

\( \gamma \)

\( \delta \)

\( \varepsilon \)

\( \zeta \)

\( \eta \)

\( \theta \)

\( \iota \)

\( \kappa \)

\( \lambda \)

\( \mu \)

\( \nu \)

\( \xi \)

\( \omicron \)

\( \pi \)

\( \rho \)

\( \sigma \)

\( \tau \)

\( \upsilon \)

\( \phi \)

\( \chi \)

\( \psi \)

\( \omega \)

\( \alpha \)

\( \beta \)

\( \gamma \)

\( \delta \)

\( \varepsilon \)

\( \zeta \)

\( \eta \)

\( \theta \)

\( \iota \)

\( \kappa \)

\( \lambda \)

\( \mu \)

\( \nu \)

\( \xi \)

\( \omicron \)

\( \pi \)

\( \rho \)

\( \sigma \)

\( \tau \)

\( \upsilon \)

\( \phi \)

\( \chi \)

\( \psi \)

\( \omega \)

\( \alpha \)

\( \beta \)

\( \gamma \)

\( \delta \)

\( \varepsilon \)

\( \zeta \)

\( \eta \)

\( \theta \)

\( \iota \)

\( \kappa \)

\( \lambda \)

\( \mu \)

\( \nu \)

\( \xi \)

\( \omicron \)

\( \pi \)

\( \rho \)

\( \sigma \)

\( \tau \)

\( \upsilon \)

\( \phi \)

\( \chi \)

\( \psi \)

\( \omega \)

\( \alpha \)

\( \beta \)

\( \gamma \)

\( \delta \)

\( \varepsilon \)

\( \zeta \)

\( \eta \)

\( \theta \)

\( \iota \)

\( \kappa \)

\( \lambda \)

\( \mu \)

\( \nu \)

\( \xi \)

\( \omicron \)

\( \pi \)

\( \rho \)

\( \sigma \)

\( \tau \)

\( \upsilon \)

\( \phi \)

\( \chi \)

\( \psi \)

\( \omega \)
Sound Texture Synthesis

Feature Training

Training sound (e.g., recorded flame sound)
Sound Texture Synthesis
Low to High Frequency Synthesis

1D Gaussian pyramid

Physically based low frequency sound
Sound Texture Synthesis: Signal Synthesis

- **i)** Compute feature for current window
- **ii)** Add window to signal
- **iii)** Add window to signal

- **Current synthesis window**
- **Next window**
Sound Texture Synthesis

Results
Dragon

Dragon height: 1.15m
Simulated with: Houdini Pyro FX solver
Candle

Wick height: 4cm

Simulated with: Houdini Pyro FX solver
Burning Brick

Brick width: 20cm

Simulated with: Houdini *Pyro FX* solver
Style Control

Torch

**Torch length:** 70cm

**Simulated with:** Houdini fire solver (blue core model with detonation shock dynamics [Nguyen et al. 2002; Hong et al. 2007])
Training clips taken from *Ultimate Fire* sound library

http://www.therecordist.com

Training clip #1

Training clip #2

Training clip #3
## Simulation/Timing data

**Simulation times:** ~2-3 hours  
**Iso-surface construction times:** ~10s per time step

<table>
<thead>
<tr>
<th>Scene</th>
<th>Domain size (m)</th>
<th>Domain resolution (voxels)</th>
<th>Animation length (s)</th>
<th>Bandwidth extension time (s)</th>
<th>Sound texture synthesis time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burning brick</td>
<td>2.0x1.7x0.9</td>
<td>180x151x78</td>
<td>5</td>
<td>64</td>
<td>20</td>
</tr>
<tr>
<td>Candle</td>
<td>0.15x0.15x0.12</td>
<td>115x120x96</td>
<td>6</td>
<td>151</td>
<td>86</td>
</tr>
<tr>
<td>Dragon</td>
<td>4.1x2.9x2.1</td>
<td>200x142x102</td>
<td>9</td>
<td>223</td>
<td>56</td>
</tr>
<tr>
<td>Flame jet</td>
<td>1.5x1.4x0.69</td>
<td>140x134x65</td>
<td>10</td>
<td>256</td>
<td>54</td>
</tr>
<tr>
<td>Torch</td>
<td>1.5x1.4x1.8</td>
<td>104x100x128</td>
<td>5</td>
<td>63</td>
<td>41</td>
</tr>
</tbody>
</table>
Acknowledgements

Webpage (with code)  http://www.cs.cornell.edu/~chadwick/fire

Renderer  Side Effects Software’s Mantra

Support  Side Effects Software (for Houdini 3D animation tools and Mantra renderer)
          The National Science Foundation (HCC-0905506)
          The Natural Sciences and Engineering Research Council of Canada
          The Alfred P. Sloan Foundation
          The John Simon Guggenheim Memorial Foundation
          Intel (Intel Science and Technology Center for Visual Computing)
          Pixar
          Autodesk
          Vision Research