Simulation and Animation of Fire

Denis Steinemann

Overview

Methods: Requirements

Motivation

What is fire?

"Fuel undergoing a chemical reaction and being converted into hot gas"  
Hot gas moves opposed to gravity and follows external wind

Essential for fire propagation

Methods for simulation of fire

Flame-based

Rendering and Modeling

Fire animation

Applications of physically based methods for 3D games and medical simulation in semantic fire

Requirements

Realistic appearance
User should be able to interact with fire
Fire spreads in space

Discussion

Results

Flame-based simulation of fire

Fire propagation

Fire animation

Rendering and Modeling

Applications of physically based methods for 3D games and medical simulation in semantic fire
Methods (1): particle-based

- Model fire using particles:
  - Hot gas as particles in user-defined velocity field $u(x,t)$
  - Render particles (just some examples)

- Warped-blobs technique: change shape of particle over time

Advantages: quite fast, realistic
Disadvantages: need many particles; fuzzy outlines

Demos: Stam, Picard, Perry

Methods (2): fluid-based

- Usually three fields:
  - Velocity
  - Density
  - Temperature

- Fields described by Navier-Stokes equations:
  - Advect diffusion sources (user-defined)

- Model fire using vector/scalar fields described by partial differential equations (PDE)

- Computational fluid dynamics (CFD): 3D vector field

Advantages: physically correct, very realistic; shows smoke
Disadvantages: very slow (megapixel frame), complex handling

- Rendering by raytracing
  - Advantages: physically correct, very realistic; shows smoke
  - Disadvantages: very slow (megapixel frame), complex handling

Demos: Fedkiw, others
I will now explain a frame-based method in detail.

**Fire simulation**

---

**Fire propagation**

- Flammable materials
- Oxygen supply
- Surface orientation relative to gravity

Main visual feature in fire propagation is growth of burning zone

---

**Overview**

- Fire propagation
- Flame-genesis and animation
- Rendering and modeling

---

**Methods for simulation of fire**

- Particle-based
- Fluid-based
- Flame-based

---

**Overview**

- Fire propagation
- Flame-genesis and animation
- Rendering and modeling

---

**Conclusion**

- Fire simulation
- Flame-genesis and animation
- Rendering and modeling
Use closed curve on an object surface very often, surface is a closed triangle-mesh. A closed curve represented as a broken line will never leave the surface! Boundary vertices have position \( x(t) \) and velocity \( v(x,t) \), depends on local params (fuel, surface orientation). Initial state: many vertices at ignition point, but each with a different initial pointing radially outward in plane of initial face.

Given \( v \) of boundary vertex \( i \), compute new position \( x_i \):

1. Move vertex according to velocity, changing direction of \( v \) if necessary.
2. Insert new vertices if necessary to force curve to stay on surface.

Two possible cases:
- \( x_i(t+h) \) lies on same face as \( x_i(t) \) → ok
- \( x_i(t+h) \) lies outside of the face → adjust \( x_i(t+h) \), using new velocity \( v_i' \):

\[
\eta = \frac{x_i(t+h) - x_i(t)}{h}
\]

Choose \( v_i' \) such that magnitude of velocity \( v_i \) is preserved: \( v_i' \times N = v_i \times N \)

In all cases, the following hold:
- Direction of velocity does not change when origin and destination faces are coplanar
- \( v_i' \) is nonzero and oriented away from the crossed edge
- Continuity of curve is preserved

Burning zone grows over time → boundary expands with each step. Burning zone grows over time → boundary expands with each step.
Denis Steinemann - Simulation and Animation of Fire

Fire propagation

Boundary vertices may become sparse/dense if velocity field changes. Movements according to some user-defined velocities lead to the formation of connected particles. The velocity of each particle is defined by local parameters. Allow only change in magnitude of $v_i$, not in direction, so that the burning zone doesn’t shrink in the next step. We will also need points inside the burning region.

Now that we know how the particles propagate on a surface, it’s time to generate and animate the actual fire:

Frame Genesis and Animation

Flame as fire primitive

Now that we know how fire propagates on a surface, it’s time to generate and animate the actual fire:

Represent a flame as a chain of connected particles. This chain is called skeleton.

Evolution of a fire front:

Relatively complex cow model; 5804 triangles

Demos: Beaudoin, Poulin, Paquet, Beaudoin,
5 steps to set up flame skeletons:

1. Define an air velocity field $V(x,t)$.
   - General form: $V_x$ and $V_y$ let the user define the air velocity field.
   - $V_z$ is constant or smoothly in the domain [0..1].

2. Plant flames on surface, inside burning zone.
   - Use extra points inside burning zone that were added during the evolution of the fire front.
   - Vary intensity of fire by varying density of points inside burning zone.

3. Define flame skeleton.
   - We have: - root $s_0$ of flame
   - Velocity field $V(x,t)$.
   - We want: curve segment $s(u)$, every point tangent to $V(x,t)$.

   - Allow for detached flames.

5. Denote an area by $V^k(x,t)$.
Denis Steinemann - Simulation and Animation of Fire

Flame Genesis and Animation

4. Grow and shrink flames

for realistic behavior, flames are newly created and may disappear after some time

for this, use length function

length function depends mostly on fuel ("fuel map" on surface)

examples:

explosive, fast burning
slow, steady burning

dov peak of length function for different flames to make fire look more realistic

5. Detach flames

if at a vertex $s_i$ of a skeleton, $s_i$ equals $s_0$

the root then can move with the velocity field in time, updated at every time step:

Use new $s_0$ to create entire flame as before
Denis Steinemann - Simulation and Animation of Fire

Since detached flames have short life span, adjust their length function.

Skeletons are well-suited for quick previewing, which speeds up the iterative process of obtaining the desired effects. Flame skeletons on a burning sphere.

Demos:

Beaudoin, Poulin, Paquet, Beaudoin, Denis Steinemann - Simulation and Animation of Fire

Overview

Rendering and Modelling

1. Define implicit surface which describes the outline of a single flame.
2. Use implicit surfaces of single flames to model a complete fire.
3. Model color variations seen inside fire.
4. Render the final image.

Skeletons are good for previewing, but they are not sufficient for photorealistic rendering.

Denis Steinemann - Simulation and Animation of Fire

Rendering and Modelling

1. Define implicit surface:

\[ \phi(x) = \frac{E_i(x)}{E_i(x) + E_j(x)} \]

where:

\[ E_i(x) = \frac{\int_{i=1}^{n} p \cdot d \phi \cdot d \rho \cdot d \theta}{\int_{i=1}^{n} p \cdot d \phi \cdot d \rho \cdot d \theta} \]

and:

\[ \phi(x) = \left[ \left( \frac{d_i}{d_i - d_j} \right)^n \right] \]

2. Resulting equation:

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

For a complete flame description, the above equation is:

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

3. Results and rendering:

For a complete flame description, the above equation is:

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]

\[ \sum_{i=1}^{n} p \cdot d \phi = \sum_{i=1}^{n} p \cdot d \phi \]
Problem: \( E(x) \) does not distinguish between root and top of flame, but flames are thin at the top and bulged at the root. Therefore, transform \( x \) to height-dependent \( x' \):

\[
\begin{align*}
&z(x) \\
r(x)
\end{align*}
\]

use cylindrical coordinates \( r(x), z(x) \) and \((s_0, s_n)\) relative to segment \((s_0, s_n)\):

\[
E(x') < E(x)
\]

near top of flame, \( x' \) is farther away from the skeleton than \( x \). Therefore, iso-surface closer to skeleton.

Putting it all together:

- Flame:
  - Two iso-surfaces \( I_1(x), I_2(x) = v \)
  - New iso-surface \( I(x) = I_1(x) + I_2(x) = v \)

- Use marching cubes algorithm to obtain a closed (polygon) surface.

3. Compute iso-surfaces to model color variation

Compute iso-surfaces of \( I(x) \)

we get enclosed surfaces to each of which we can assign a color.

- Use marching cubes algorithm to obtain a closed (polygon) surface.

4. Render fire using iso-surfaces

use raytracing (raycasting)

- Cast rays toward polygons of iso-surfaces
- Calculate length of a ray inside each layer
- Add up color contributions from different layers for each ray
- Omit light-scattering inside fire
That's how it finally looks:

"Demos:"

Beaudoin, Paquet, Beaudoin

Results

Surfaces with a few hundred triangles, 400MHz, 256MB RAM

- Propagation: 15 fps
- Skeleton animation (100 skeletons): 4 fps
- Complete with rendering: 4 frames per minute

Not real-time, but compared to fluid-based methods, flame-based approach is much faster.

Discussion

Features of introduced flame-based method:

- sleek, smooth outlines
- relatively simple, but nonetheless realistic
- fast
- allows for quick previewing (skeletons)
- easily scalable
- features intuitive handling

Limitations:

- Fire cannot spread to other, disconnected objects
- Rendering rather simple (no self-lighting, no illumination of other objects)
- no smoke

Further Possibilities:

- Charring – draw a black splat on object surface when flame becomes extinct
- start new propagations on other objects
References


