Simulation and Animation of Fire

Presentation in Seminar on

Physically-based Methods for 3D-Games and Medical Applications

Denis Steinemann

Overview

Motivation

Methods for simulation of fire
- particle-based
- fluid-based
- flame-based

Flame-based simulation of fire
- fire propagation
- flame animation
- rendering and modelling

Results

Discussion

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
- essentially methods model hot gas

Why simulate fire?

- Dangerous, not easy to control in real-world
- sometimes, it is impossible to create a real-world model (for example, how does fire propagate in a building?)
- adds visual realism to 3D-Applications
- Games, movies

Methods: Requirements

- realistic appearance
- user should be able to interact with fire
- fire spreads in space
- scalability: method should work for small and for large fires
- controllable, easy to handle user-parameters

- trade-off:

  concentration on realistic visual behavior, but faster

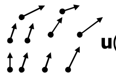
  ⟷

  physically correct, very realistic, but also very slow

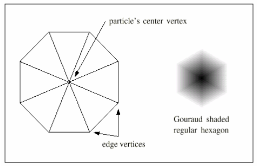
Methods (1): particle-based

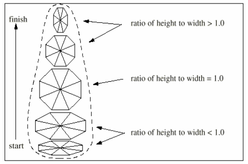
Model fire using particles:

- Hot gas as particles in user-defined velocity field **u**(**x**,t)



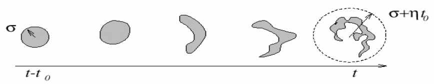
- Render particles (just some examples)
  - with polygons: change aspect ratio over time





Methods (1): particle-based

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



Particle-based methods:

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

[Perry,Picard]

- Demos
  - Stam

Methods (2): fluid-based

Model fire using vector/scalar fields described by partial differential equations (PDE) → **Computational Fluid Dynamics (CFD)**:

- usually three fields:
  - Velocity $\mathbf{u}(\mathbf{x},t)$
  - Density $\rho(\mathbf{x},t)$ } for water & smoke, these two suffice
  - Temperature $T(\mathbf{x},t)$
- fields described by Navier-Stokes equations:

$$\frac{\partial \mathbf{v}}{\partial t} = -(\mathbf{v}\cdot\nabla)\mathbf{v} + \nu\nabla^2\mathbf{v} + \mathbf{f}$$

$$\frac{\partial \rho}{\partial t} = -(\mathbf{v}\cdot\nabla)\rho + \kappa\nabla^2\rho + S$$

$$\frac{\partial T}{\partial t} = \underbrace{-(\mathbf{v}\cdot\nabla)T}_{\text{advect}} + \underbrace{c\nabla^2 T}_{\text{diffusion}} + \underbrace{H}_{\text{sources (user-defined)}}$$

Methods (2): fluid-based

- Navier-Stokes equations hold for any fluid
- for fire, we also need temperature, because it determines the color of the fire (wavelength of light depends on temperature)

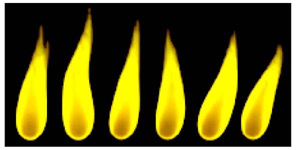
- Solve equations (expensive!); update fields using Euler, etc.
- the *density* field is rendered, using temperature to determine color (yellow fire, blue fire, smoke?)
- Rendering by Raytracing

- Advantages: physically correct, very realistic; shows smoke
- Disadvantages: very slow (5min per frame), complex handling

- Demos:
  - Fedkiw

Methods (3): flame-based

- single flame as fire primitive
- represent fire as a set of flames
- Advantages: faster (avoid solving very costly PDE), scalable, realistic, sharp outlines
- Disadvantages: not physically correct (concentration on visual behavior), not as realistic as with CFD



[Beaudoin, Paquet, Poulin]

Overview

Motivation
Methods for simulation of fire
  particle-based
  fluid-based
  flame-based
Flame-based simulation of fire
  fire propagation
  flame animation
  rendering and modelling
Results
Discussion

Flame-based simulation of fire

- I will now explain a flame-based method in detail
- Paper:
  Realistic and Controllable Fire Simulation
  Philippe Beaudoin, Sébastien Paquet, Pierre Poulin
  Université de Montreal
- **Divide simulation of fire into 3 subproblems:**
  - Fire propagation
  - Flame genesis and animation
  - Rendering and modelling

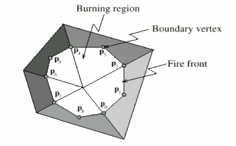
Fire propagation

- Main visual feature in fire propagation is growth of burning zone
- Growth depends on an few locally defined parameters:
  - Fuel density
  - Oxygen supply
  - Surface orientation relative to gravity

Our goal is to track the boundaries between parts of an object that are burning and those which have not yet been reached by the fire

Fire propagation

- Use closed curve on an object surface
- very often, surface is a closed triangle-mesh → closed curve represented as a broken line

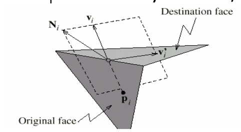


broken line will never leave the surface!

- Boundary vertices have
  - Position **x**(t)
  - Velocity **v**(**x**,t), depends on local params (fuel, surface orientation)
- Initial state: many vertices at ignition point, but each with a different $\mathbf{v}_{init}$ pointing radially outward in plane of initial face

Fire propagation

- Given $\mathbf{v}_i$ of boundary vertex i, compute new position $\mathbf{x}_i$:



$$\mathbf{x}_i(t+h) = \mathbf{x}_i(t) + h \cdot \mathbf{v}_i(t)$$

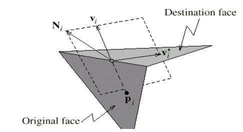
- two possible cases:
  - $\mathbf{x}_i$(t+h) lies on same face as $\mathbf{x}_i$(t) → ok
  - $\mathbf{x}_i$(t+h) lies outside of the face
    → adjust $\mathbf{x}_i$(t+h), using new velocity $\mathbf{v}_i'$

$$\mathbf{v}_i' = \eta(\mathbf{N}_i \times \mathbf{v}_i) \times \mathbf{N}'$$

  - $\mathbf{v}_i$: orig. velocity; $\mathbf{N}_i$: linearly interpolated normal at crossing point
  - η chosen to preserve magnitude of velocity
  - N´: normal to destination face

Fire propagation

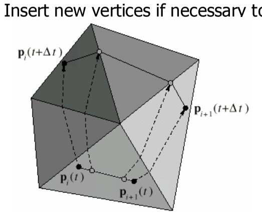
- in all cases, the following hold:
  - Direction of velocity does not change when origin and destination faces are coplanar
  - $\mathbf{v}_i'$ is nonzero and oriented away from the crossed edge
  - continuity of curve is preserved



Fire propagation

- Burning zone grows over time → boundary expands with each time step

→ 2-step update:
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

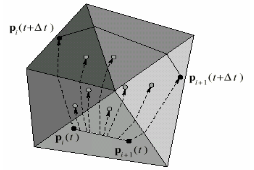


compute new vertex by using binary search to find point on original line that is displaced directly onto edge (velocity interpolated from endpoint velocities)

Fire propagation

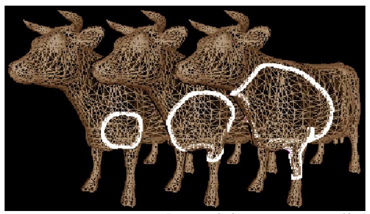
- Boundary vertices may become sparse/dense → insert/delete vertices if that's the case
- $\mathbf{v}_i$ is defined by local parameters. Allow only change in magnitude of $\mathbf{v}_i$, not in direction, so that the burning zone doesn't shrink
- in the next step, we will also need points *inside* the burning region

  → for a given line segment, displace random point on segment just like boundary vertex, but with interpolated velocity and a time step [0..h]



Fire propagation

- Evolution of a (fire) front:
  - Relatively complex cow model; 5804 triangles



[Beaudoin, Paquet, Poulin]

  - Demos: Beaudoin

Overview

Motivation
Methods for simulation of fire
- particle-based
- fluid-based
- flame-based

Flame-based simulation of fire
- fire propagation
- flame animation
- rendering and modelling

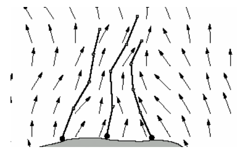
Results
Discussion

Flame Genesis and Animation

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

- single flame as fire primitive
- represent fire as a set of flames
- Flame = stream of gas that follows the air surrounding it

→ ***Represent a flame as a chain of connected particles that moves according to some user-defined air velocity field. This chain is called* skeleton.**



skeletons dancing in the wind

Flame Genesis and Animation

- 5 steps to set up flame skeletons:

  1. define air velocity field
  2. plant flames on surface
  3. define flame skeletons
  4. grow/shrink flames
  5. allow for detached flames

Flame Genesis and Animation

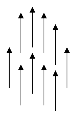
1. Define an air velocity field V(x,t)

- User-defined
- general form:

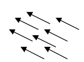
$$\mathbf{V}(\mathbf{x},t)=\sum_{i=0}^{n} b_i(\mathbf{x},t)\mathbf{V}_i(\mathbf{x},t)$$

  - b_i blending function; constant or smoothly in [0..1]
  - lets user treat specific features of animated fire separately by giving each $\mathbf{V}_i$ certain characteristics

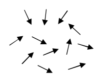
- an example: $\mathbf{V}(\mathbf{x},t)=-k\mathbf{g}+\mathbf{V}_L(\mathbf{x},t)+\mathbf{V}_S(\mathbf{x},t)$



-**kg**



large-scale wind-field $\mathbf{V}_L$

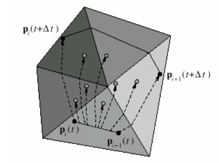


small-scale turbulences $\mathbf{V}_S$

Flame Genesis and Animation

2. Plant flames on surface, inside burning zone

- use extra points inside burning zone that were additionally created during the evolution of the fire front

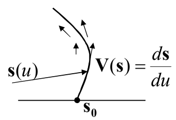


- vary intensity of fire by varying density of points inside burning zone

Flame Genesis and Animation

3. Define flame skeleton

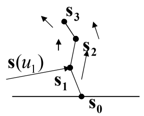
- we have: - root $\mathbf{s}_0$ of flame
  - velocity field **V**(**x**,t)
- we want: curve segment **s**(u), every point tangent to instantaneous **V**(**x**)



- **s**(u) is the solution of the differential equation $\mathbf{V}(\mathbf{s})=\frac{d\mathbf{s}}{du}, \mathbf{s}(0)=\mathbf{s}_0$

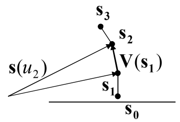
Flame Genesis and Animation

- in general, **s**(u) is hard to find analytically
- skeleton is a broken-line approximation of **s**(u)



- solve numerically using Euler
  - skeleton vertices: $\mathbf{s}_i = \mathbf{s}(u_i)$

$$\mathbf{s}_{i+1} = \mathbf{s}_i + \frac{l(t)}{n}\mathbf{V}(\mathbf{s}_i)$$

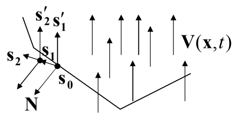


  - n predefined, (n<10); n+1 = # of vertices in skeleton
  - $l(t)$ determines length of skeleton (see later)

Flame Genesis and Animation

- Problem: skeleton may penetrate object, because **V(x**,t) does not consider geometry

  → adjust vector field locally by adding vector **N** normal to the surface at root $\mathbf{s}_0$

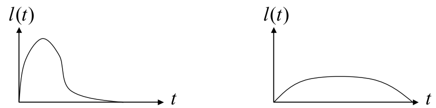


  - $|\mathbf{N}|$ large enough for $\mathbf{s}_i$ to be outside of object

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 $l(t)$
- length function depends mostly on fuel (→ „fuel map" on surface)
- examples:



explosive, fast burning | slow, steady burning

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

Flame Genesis and Animation

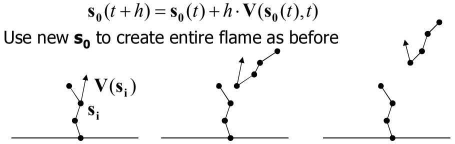
- technique described so far works well for quiet flames
- in highly turbulent fires, flames can take off into the air, drift upwards for some time and then disappear

5. Detach flames

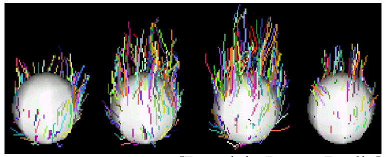
- if at a vertex $\mathbf{s}_i$ of a skeleton, $|\mathbf{V}(\mathbf{s}_i)| > threshold$
- insert new flame whose root $\mathbf{s}_0$ equals $\mathbf{s}_i$ *and is free*
- the root then can move with the velocity field in time, updated at every time step:

$$\mathbf{s}_0(t+h) = \mathbf{s}_0(t) + h \cdot \mathbf{V}(\mathbf{s}_0(t), t)$$

- Use new $\mathbf{s}_0$ to create entire flame as before



- 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

[Beaudoin,Paquet,Poulin]

- Demos:
  - Beaudoin

Overview

Motivation
Methods for simulation of fire
- particle-based
- fluid-based
- flame-based

Flame-based simulation of fire
- fire propagation
- flame animation
- rendering and modelling

Results
Discussion

Rendering and Modelling

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

→ **dress up the skeletons**

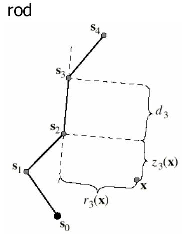
- 4 steps:

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

Rendering and Modelling

1. Define implicit surface

- physical analogy: electrical potential induced by a uniformly charged rod



$E_i(\mathbf{x})$: „potential of **x**, induced by segment i of the skeleton"

$$E_i(\mathbf{x}) = \int_{p=0}^{d_i} \frac{1}{\sqrt{(p - z_i(\mathbf{x}))^2 + r_i(\mathbf{x})^2}}\,dp$$

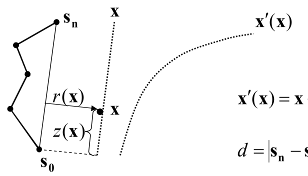
$$= \sinh^{-1}\left(\frac{z_i(\mathbf{x})}{r_i(\mathbf{x})}\right) - \sinh^{-1}\left(\frac{z_i(\mathbf{x}) - d_i}{r_i(\mathbf{x})}\right)$$

$$d_i = |\mathbf{s_i} - \mathbf{s_{i-1}}|$$

- $z_i(\mathbf{x})$ and $r_i(\mathbf{x})$: cylindrical coordinates of **x** relative to segment i
- for a complete flame, „potentials" add up: $E(\mathbf{x}) = \sum_{i=1}^{n} E_i(\mathbf{x})$

Rendering and Modelling

- Problem: E(**x**) does not distinguish between root and top of flame, but flames are thin at the top and bulged at the root
  → transform **x** to height-dependent **x**´(**x**):



$$\mathbf{x}'(\mathbf{x}) = \mathbf{x} + \exp\left(\frac{2z(\mathbf{x})}{d} - 1\right) r(\mathbf{x})$$

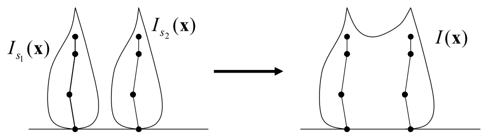
$$d = |\mathbf{s}_n - \mathbf{s}_0|$$

- z(**x**) and r(**x**): cylindrical coords of **x** relative to segment ($\mathbf{s}_0$,$\mathbf{s}_n$)
- use E(**x**´(**x**)) instead of E(**x**)
- near top of flame, **x**´(**x**) is farther away from the skeleton than **x**
  → E(**x**´(**x**)) smaller than E(**x**) → iso-surface closer to skeleton

Rendering and Modelling

2. Use implicit functions to model complete fire

Putting it all together:
- Flame: $I_s(\mathbf{x}) = E(\mathbf{x}'(\mathbf{x}))$
- Fire: $I(\mathbf{x}) = \sum I_s(\mathbf{x})$



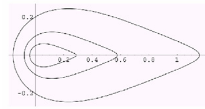
two iso-surfaces $I_{s1}(\mathbf{x})$,$I_{s2}(\mathbf{x})$=v new iso-surface $I(\mathbf{x})=I_{s1}(\mathbf{x})+I_{s2}(\mathbf{x})$=v

Rendering and Modelling

- we have the *shape* of flames (and fire)
- now we also want *color*
- Color depends on temperature (fire emits radiation of different wavelength). Temperature in flame mainly depends on distance to base and to the center of the flame

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

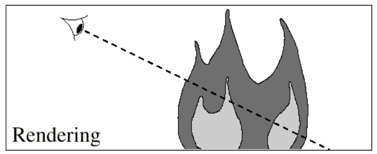


- use marching-cubes algorithm to obtain a closed (polygon) surface

Rendering and Modelling

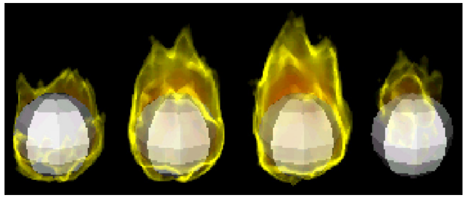
4. Render fire using iso-surfaces

- 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



Rendering and Modelling

That's how it finally looks:



[Beaudoin,Paquet,Poulin]

- Demos:
  - Beaudoin

Results

- Surfaces with a few hundred triangles, 400MHz, 256MB RAM
  - Propagation: 15 fps
  - Skeletion 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

Discussion

- Limitations:
  - Fire cannot spread to other, disconnected objects
    → start new propagations on other objects
  - Rendering rather simple (no self-lighting, no illumination of other objects by fire)
  - no smoke
- Further Possibilities:
  - Charring – draw a black splat on object surface when flame becomes extinct

[Perry,Picard]

References

[1] P. Beaudoin, S. Paquet, P. Poulin. Realistic and Controllable Fire Simulation. 2001.

[2] C.H. Perry and R.W. Picard. Synthesizing flames and their spreading. In *Fifth Eurographics Workshop on Animation and Simulation*, pages 105-117, September 1994.

[3] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH 1995 Conference Proceedings*, pages 129-136, August 1995.

[4] D.Q. Nguyen, R. Fedkiw, and H.W. Jensen. Physically Based Modelling and Animation of Fire. 2001