

## Lecture 17: Graphical Simulation of Fire

Amorphous objects, such as fire, smoke, water, mist and fog are difficult subjects for graphical rendering since they do not have well defined boundaries, and are naturally mobile. They are currently the subject of considerable research and development because of their commercial importance in the film and games industries. Recent systems, using fluid mechanics have created some very impressive effects, but at high computational cost. These are being deployed in film, where each frame of an animation may be computed off-line. A major challenge remains in real time computer graphics, where effects must be computed at frame rate (typically 25 times/second). In this lecture we will look at fire as an exemplar, and discuss some of the approaches that have been taken to simulate it visually.

The simplest way to create a fire is to hand draw a set of polygons which can be overlaid on an image. This is a time consuming and unadaptable procedure, but it can be quite effective for special effects, such as short lived explosions or gun shot. If a sequence of polygons is used in isolation the effect is unconvincing because fire emits light and will illuminate the surrounding polygons of the scene. A further refinement is to place one or more light sources around the position of the fire, and to control their illumination settings to synchronise with the polygons representing the fire. This adds realism at little computational cost, and so is suitable for real time use in computer games. There are obvious limitations of polygon fires. They are difficult to sustain for any length of time without intrusive repeats. Changes of shape, and spreading of the fire are difficult to create, and translucency of the flames cannot be easily created.

### Particle Models

A better approach to creating fire effects is to try to model the underlying physical process more accurately. Flames are incandescent gases, formed where a combustible gas has an interface with air, whose visual appearance depends on their temperature, pressure and density. The dynamics of flames is determined by diffusion. As the gases diffuse they become less dense and they cool introducing changes of colour and translucency. Clearly this is a complex process, and exact modeling is not possible. However, approximate models can be constructed to create good visual effects.

One way in which physicists model gases is as a collection of particles, and this is indeed the approach that has been taken for most computer graphics systems. Particles form a discrete approximation to a continuous material, and have the advantage that they can be highly mobile, creating shape changes. They can given stochastic behaviour properties to create animations that do not repeat regularly.

As a simple illustration (using water rather than fire), consider the case of a jet of water. At low resolution we can think of the particles as being equivalent to droplets. Each has a mass and is ejected from the nozzle with a particular velocity. Once moving freely in the air each droplet is acted on by gravity and perhaps wind forces and its behaviour is governed by Newtonian mechanics. Simple equations determine the path of each particle during one frame interval, and they can be rendered by simple streaks, or blobs as shown in Figure 1. Using this technique pleasing visual effects can be created with very little computation.

For modeling fire, particles can have a number of attributes which include:

Position	[x,y,z]
Velocity	[u,v,w]
Mass	m
Temperature	$\theta$
Lifetime	t



Figure 1: Particle Fountain

Most systems represent particles as points or small spheres, but it might also be possible to include shape as an attribute. Particles can be created with some stochastic elements to provide variability, and can be made to move using physical rules. They can be given a lifetime, and they can be rendered using a number of techniques to create the effects of flame or smoke. The simplest way to move particles is to give them Newtonian dynamics, that is to say they conform to Newton's first law:  $\mathbf{f} = m\mathbf{a}$  where  $\mathbf{f}$  is a vector force, and  $\mathbf{a}$  is a vector acceleration.

To solve for the particle's motion we need to integrate using:

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} = \frac{d^2\mathbf{x}}{dt^2}$$

where  $\mathbf{v}$  is the velocity and  $\mathbf{x}$  is the position. Since we are working on a fixed frame interval we can carry out the integration using a simple numerical approximation:

$$\mathbf{v}_{t+1} = \mathbf{a}_t\Delta t + \mathbf{v}_t \quad \mathbf{x}_{t+1} = \mathbf{v}_t\Delta t + \mathbf{x}_t$$

An easy extension of this is to introduce damping (or friction) into the equations. This is appropriate for gases since it describes the viscous drag on each particle:

$$\mathbf{f} = m\mathbf{a} + s\mathbf{v}$$

Another possible extension is to allow particles to collide. This however is much harder to do and is computationally very intensive. It requires sub frame calculation, and some specification of the particle size and shape. It involves a large number of particle/particle checks. The effect of collisions can be simulated by applying small random forces to a proportion of the particles at each frame interval.

Particle rendering is a major component in creating a realistic fire. Some possible strategies are (i) to make the colour age dependent (ii) make the colour temperature dependent. The latter strategy means that it is necessary to have some physical modeling that accounts for particle cooling.

### Exemplar: The Wrath of Khan



Figure 2: The Explosion in the Wrath of Khan (Reeves(1983))

The first major use of a particle system for creating fire was in the star trek film “The Wrath of Khan” and was published by Reeves et al (SIGGRAPH 1983 17(3) 359-376). The sequence depicts a planet being brought to life by exploding a bomb on it. The particles were emitted from locations arranged in concentric rings around the point of impact of the bomb. By using a timed activation sequence for the particle systems the effect of flames spreading could be simulated. In total 750,000 particles were used - this was a huge number in 1982, but is quite modest in today's terms. The particles were generated with velocity directions distributed normally about the normal vector to the surface as shown in figure 3. The velocity magnitude was also normally distributed. Particles were slowed down by a gravitational force, but extinguished after a period of time so that their parabolic path was not apparent. Each particle was rendered by a short line segment whose colour was selected to simulate the colour characteristics

of different temperatures. Yellow represents the hottest, cooling to red through the particle life time. Since the particles did not emit light themselves, a fake light source was placed above the point of impact on the planet. This can be seen clearly in figure 2.

The fire in The Wrath of Khan was certainly groundbreaking at the time, but it fell short of creating a realistic visual simulation in two respects. First rendering using simple line streaks is unnatural. The colouring successfully fools the eye into accepting that what can be seen is a flame, but the flames do not travel in straight lines. In practice the particles can describe the presence of a combustible gas, but the flame occurs at the boundary of the gas, where it combines with oxygen to burn. Thus rendering each particle individually will not produce a good result. Secondly the motion of the particles is not realistic. Burning gases are lighter than air and so tend to rise and spread out rather than fall under gravitational forces.

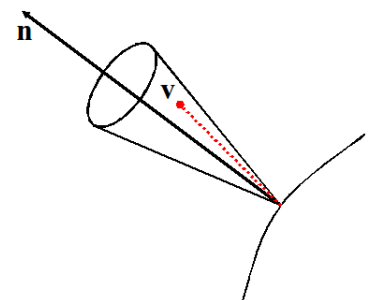


Figure 3: Generating Particles

## Improved rendering: Bitmap Splatting

Bitmap splatting is a technique that involves projecting particles onto the raster, and then rendering them by blending a small and appropriate bitmap with the pixels already rendered. For flames the bitmaps are made up of appropriately coloured pixels, with a higher proportion of yellow for hotter regions, and a high proportion of black (invisible) as the particles cool. The blending allows the possibility of flames being translucent, and makes the visual stimuli correspond better to the visual characteristic of fire. Bitmap splats are also very fast to render which makes them attractive for fires created with small numbers of particles.

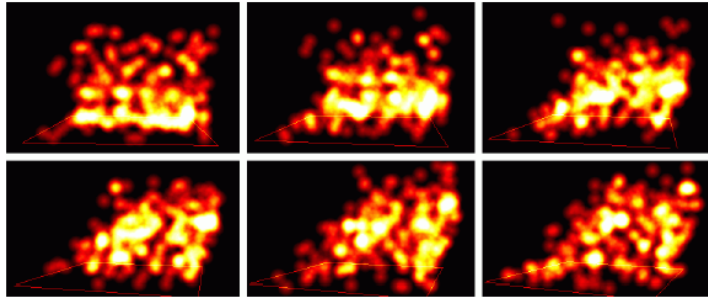


Figure 4: Bitmap Splat fire by Engell-Nielsen and Madsen, University of Copenhagen

## Improved Rendering: Ray Tracing Particle Systems

An improvement in rendering can be achieved using ray casting or ray tracing. As before the flame dynamics is modeled by the particles, but the rendering now takes into account the number and density of the particles without actually rendering them directly. For ray casting, a blob (usually a sphere) is placed around each particle. A ray is cast from each pixel and out into the particle system. If it passes through a blob, it takes on a proportion of the blob colour which depends on the temperature of the blob. A ray that misses all the blobs will be rendered with the colour of the background object it hits. A ray passing through a few blobs will be rendered with a blend of the colours collected from the blobs and the background, and a ray passing through a dense region of particles will take its entire colour from the particles. This will create effects such as translucency of the flames.

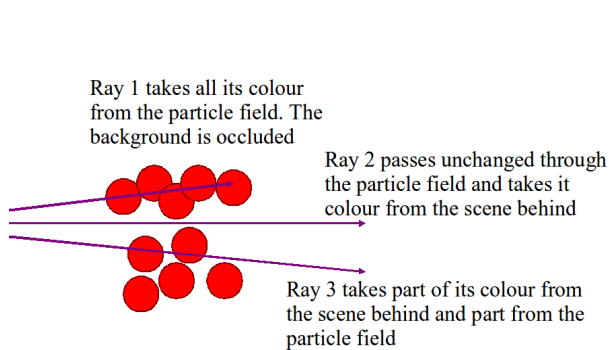


Figure 5: Ray Casting through flames

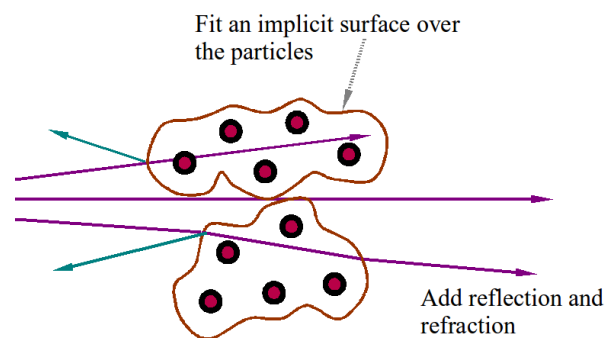


Figure 6: Ray Tracing Flames

A better approach is to use implicit surfaces rather than blobs. A simple way to fit an implicit surface is to consider each particle a point charge, and then calculate an iso-potential surface. This provides a better model since the implicit surface can be considered the interface between the burning gas and the air. The particle motion still controls the dynamics of the flame, but the rendering represents the interface between the combustible gas and the air more realistically. Ray colours can be accumulated as before, with a temperature being computed for each point on the implicit surfaces, and there is now the possibility of calculating refractions and reflections of the rays for even better effects, though this will be a time consuming process.

A further step in adding realism is to use radiosity to make the flame illuminate the surrounding objects. This is a major computation task, since the particles move in each frame and therefore the form factors need

to be re-computed at each step. In order to do this we need to create the implicit surface and then tessellate it using polygons, for example triangles.

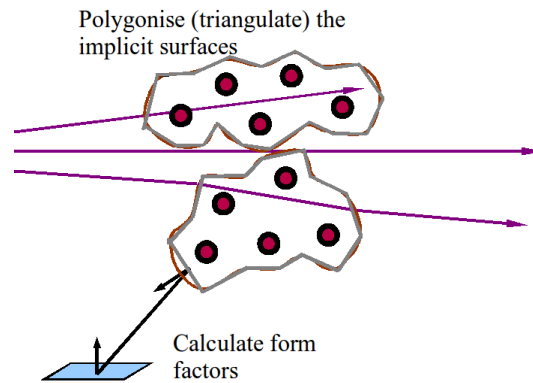


Figure 7: Radiosity effects in rendering flames

## Improved motion

One of the big challenges in simulating fire is making the particles move in a plausible manner. There are a number of heuristic techniques for doing this, and most are based on the creation of a force (or velocity) field in the volume in which the particle system exists. Force fields can create effects such as diffusion. Gases diffuse from more dense regions to less dense ones and cool as part of the process. This process is computed in a discrete manner by dividing the volume where the fire is burning into voxels. At any one time the pressure in a voxel can be taken as proportional to the number of particles in it, and diffusion forces can be applied to each particle before integrating to calculate the velocity and the motion. A further refinement is to remove particles as they come close to the boundary (implicit surface), which is where the gas burns to form  $CO_2$ .

If a velocity field is used then particles starting from any single point will follow a streamline, which can be found by integration. A point (or particle) in a velocity field will change its position in frame time by  $\mathbf{v}\Delta(t)$  where  $\mathbf{v}$  is the velocity at its position in the field. Particles, depending on their positions follow different streamlines, and the combined motion can be used to create the effect of the motion of the incandescent gas. Velocity fields on their own produce static streamlines, and therefore could be too repetitive to create a pleasing visual effect. They could be used in conjunction with force fields to create more varied motion.

More recently, there has been an interest in using fluid dynamics, described by the Navier-Stokes equation, to determine how the gas actually flows, and to create effects such as swirling and vortices which can often be seen in flames. The Navier-Stokes equation is a differential equation connecting the force, velocity, viscosity and pressure at each point within a fluid. It is usual to assume that the fluid is incompressible, and even with flames this assumption gives reasonable results. Numerical solutions to the Navier-Stokes equation can be found for fixed boundary conditions. They could be used to create a dynamic velocity field in 3D which can be used to update the particles, or to solve directly for the gas air interface. Methods based on the Navier-Stokes equation are beyond the scope of this course. More details can be found in: Nguyen, Enright and Fedkiw - Simulation and Animation of Fire.