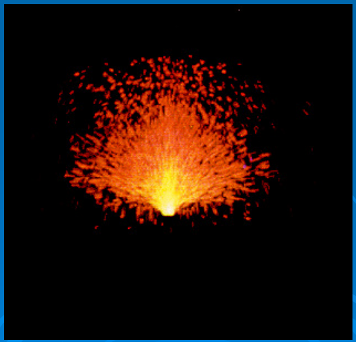
Particle systems

I.Kolingerová

1. Particle systems – basic description
2. Particle generations
3. Particle attributes
4. Particle dynamics
5. Particle removal
6. Particle rendering
7. Systems with mutual interaction
8. Examples



Reference:

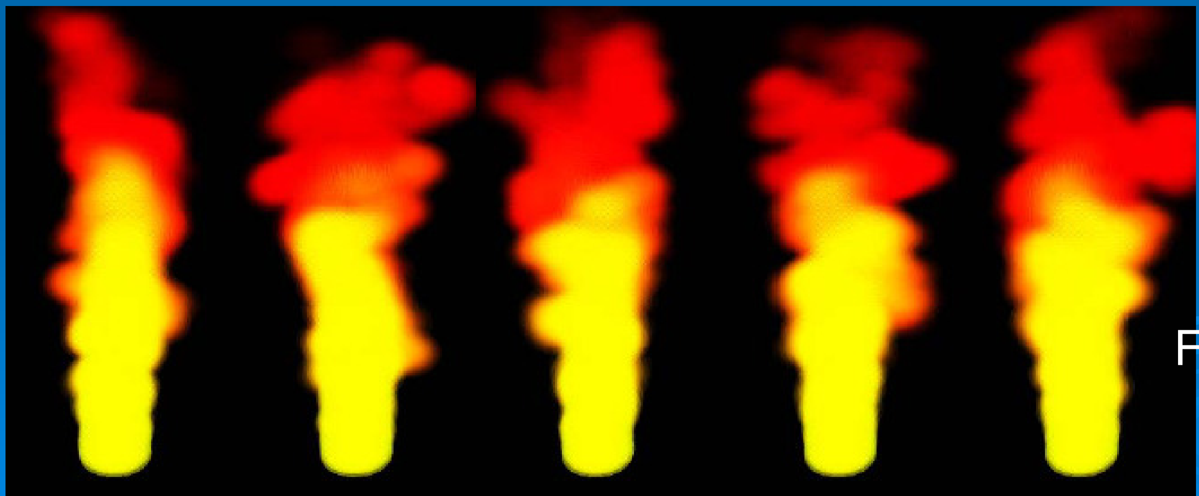
- W.T.Reeves: Particle Systems – a Technique for Modeling a Class of Fuzzy Objects, ACM Transactions on Graphics, Vol. 2, No. 2, 1983, pp.91-108

1. Particle systems – basic description

- A method to generate fuzzy objects - fires, fireworks, explosions, grass, clouds, water ...
- Fuzzy objects – difficult to represent, they do not have smooth, well-defined surfaces, their movements cannot be described by affine transformations
- Particle system – the object is modelled as a cloud of primitive particles defining the object volume
- Particles can be assigned dynamics and a model of their look
- Generating shape: sphere, circle, rectangle,...

- Differences from the usual representations:
  - Instead of a set of simple surface elements, a lot of particles defining the volume
  - Particles are born and die
  - Not totally deterministic, stochastic processes are used to create and change the look
- Advantages:
  - Particles more simple than a polygon
  - Procedural definition, controlled by random numbers => a fast design
  - Level-of-detail (LOD) according to the view parameters can be done
  - “Living" system – the dynamics is easier than with the surface description

- 1st frame computation:
  - Generation of new particles into the system
  - New particles get individual attributes
  - Particles older than prescribed are killed
  - The surviving particles are shifted and transformed according to their attributes
  - Living particles are rendered into the frame buffer



Flames

- Example:

bin/psys.exe

2. Particle generation

- Supervised stochastic process controlls the number of particles entering the system in each frame

$$Nparts_f = MeanParts_f + Rand\ () \times VarParts_f$$

Current number of particles in the frame n

Mean value of the number

Random number with the uniform distribution in <0,1>

Deviation

Or:

$$Nparts_f = (MeanParts_{saf} + Rand\ () \times VarParts_{saf}) \times ScreenArea$$

Screen area covered by the system

Mean value of the number/ Screen area

Deviation of the number/ Screen area

Number of particles can vary in time:

$$Meanparts_f = InitialMeanParts_f + DeltaMeanParts \times (f - f0)$$

The current frame (f)

1st frame when the system is alive (f0)

Mean value of the particles' number in the first frame (InitialMeanParts)

Speed of change (DeltaMeanParts)

- Similarly for $MeanParts_{saf}$
- VarParts without a change
- The change can be quadratic, cubic, stochastic...

Particle hierarchy

- The mean value and dispersion are used for the group of offspring of the given particle

3. Particle attributes

- Starting position – inside a generating shape, 3D point + 2 angles for the orientation
- Initial speed – size and direction

  InitialSpeed = MeanSpead + Rand()xVarSpeed
- Initial size – it gets the average and the maximum dispersion
- Initial colour – it gets the average of R,G,B and the maximum dispersion
- Initial translucency – it gets the average and the maximum dispersion
- Shape – sphere, rectangle, ...
- Life expectancy – number >= 0, decremented

4. Particle dynamics

- The simplest: the position – a function of time
- More complex behavior – external forces influencing the particles
- Movement equations:

$$v = v_0 + \int a\,dt$$

$$p = p_0 + \int v\,dt$$

p – particle position
v – velocity vector
p$_0$, ***v***$_0$ – initial position and velocity
a = ***f***/m (external force/ particle mass)

- Approximation:

$$v_{n+1} = v_n + a\Delta t$$

$$p_{n+1} = p_n + v_{n+1}\Delta t$$

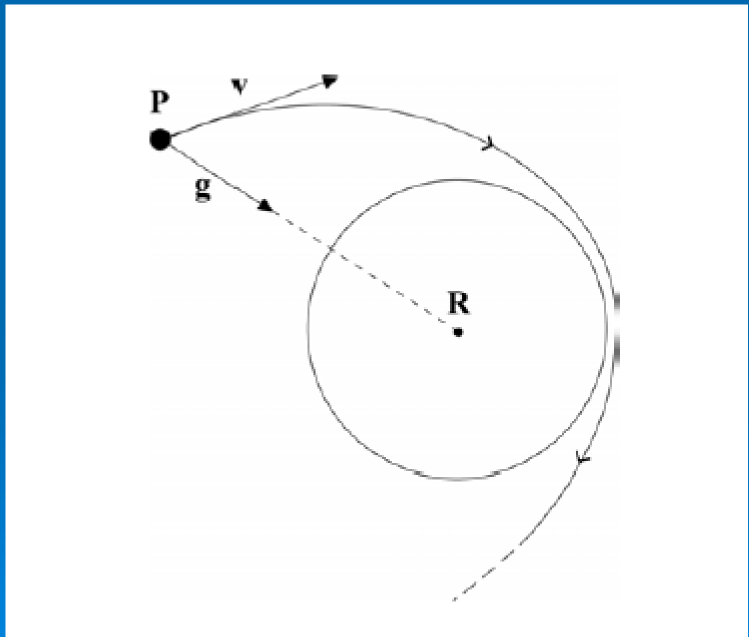
Δt – simulation time step

The most often force - gravitation:

$$f_g = mgd$$

g – gravitational acceleration
d – directional vector

- Visual result: parabolic movement



Particle movement
around the centre of gravitation

Neigbourhood counteraction: against the movement direction

$$f_r = -k_r v$$

k_r – constant of the ngb. counteraction
$\boldsymbol{v}$ – original velocity vector

Reflex from geometrical objects:

$$v' = v - 2(vn)n$$

v´ – particle velocity vector after the reflex
v – original velocity vector
n – normal vector of the reflecting surface

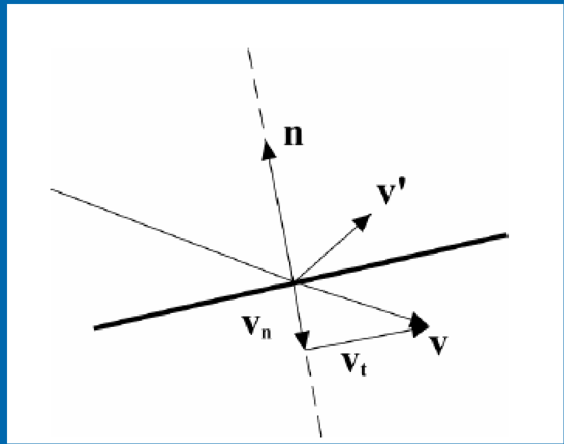
- Particle reflects as late as from a position under the surface to which it penetrated in the previous step of the simulation – it looks unnatural for a bigger simulation step
- Possible solution: compute more exact reflection point from the previous particle position, compute new velocity from it

Further imprecissement: elastic collision:

- Velocity after the reflex has two components, normal and tangent

$$v_n = (vn)n$$

$$v_t = v - v_n$$



- New velocity after the collision is

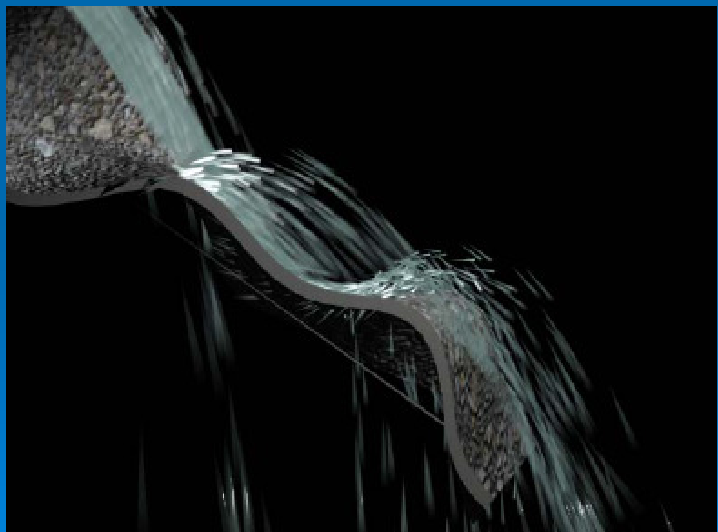
$$v' = (1-\mu)v_t - \varepsilon v_n$$

μ – decreases the tangent component
– friction coefficient

ε – influences the normal component
– flexibility coefficient

5. Particle removal

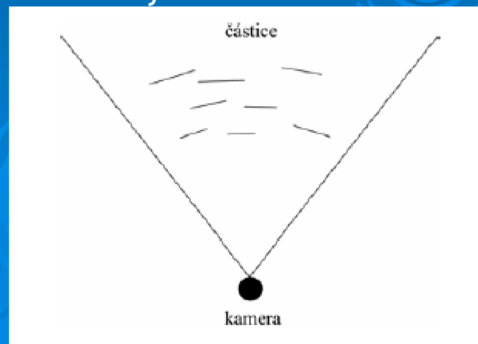
- When life expectancy is achieved
- In case the particle moves in the given direction by more than allowed



Waterfall

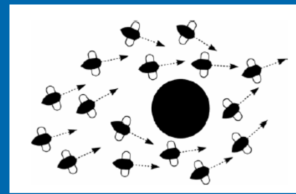
6. Particle rendering

- Mutual overlap of particles and classical objects, transparency, invisibility ....
- Possible simplification:
  - To divide one particle system into more to eliminate intersections with surface-modeled primitives, compilation done later
  - Particles – point light sources – neither shadow-casting, nor invisibility (only the pixels cummulate the light), values only cut to max., no depth-sort needed
- Simplification OK for explosions and fires, not suitable for clouds and water
- Often instead of particles, texture rectangles (sprites)



7. Systems with mutual particle interaction

- Mutual attraction and repulsion, collision detection, splints, animals, birds, people … instead of particles
- Main idea: relatively simple rules how an individual behaves in a flock/herd:
  - Collision avoidance
  - Adaptation to the near-individuals movement
  - Keeping near the flock\herd – direction to the centre of near individuals



- For the i-th particle:

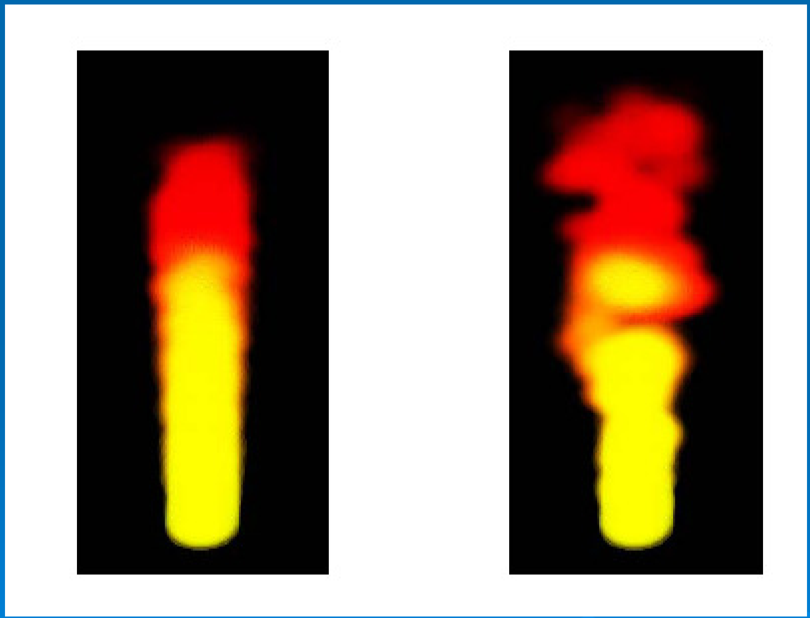
$$\boldsymbol{v}_i' = \boldsymbol{v}_i + \boldsymbol{a}_i \Delta t, \quad \boldsymbol{a}_i = \boldsymbol{f}_{ext}(\boldsymbol{p}_i, \boldsymbol{v}_i)/m, \quad \boldsymbol{f}_{ext} = \boldsymbol{f}_g + \boldsymbol{f}_r + \boldsymbol{f}_{env},$$

$\boldsymbol{f}_g = m_i g \boldsymbol{d}$ - gravitation, $\boldsymbol{f}_r = -\varepsilon \boldsymbol{v}_i$ - environment repulsion,

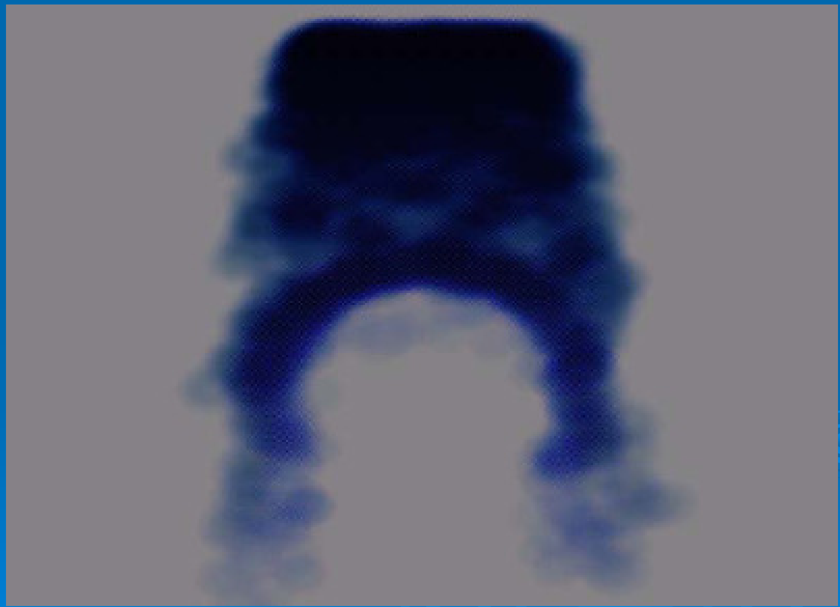
$\boldsymbol{f}_{env} = \boldsymbol{f}_{env}(\boldsymbol{p}_i, \boldsymbol{v}_i)$ - neighbourhood (the user defines)

Environment

- For ex. Perlin noise – causes irregular flame waving and particles clustering into little clouds => approximation to gas turbulations
- Or a repulsing force on the gas surface => flowing around



Flames with and without a noise function



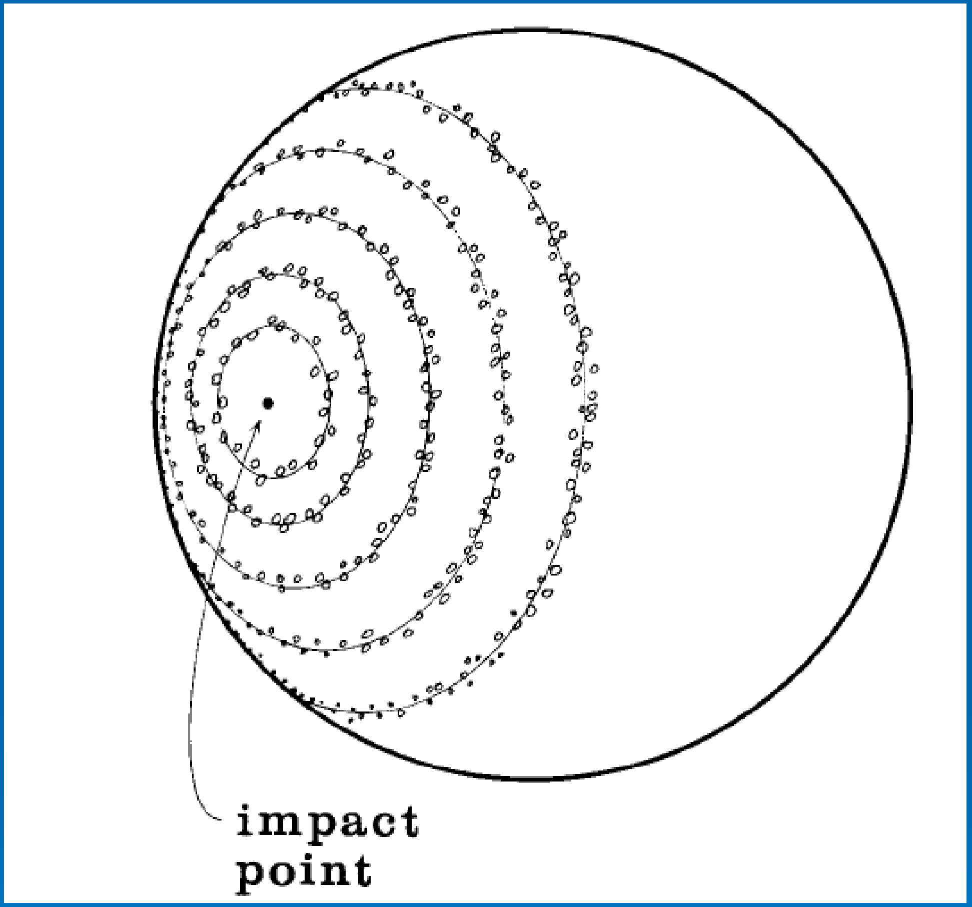
Sphere runarounded by particles

8. Examples

Wall of fire and explosion (1):

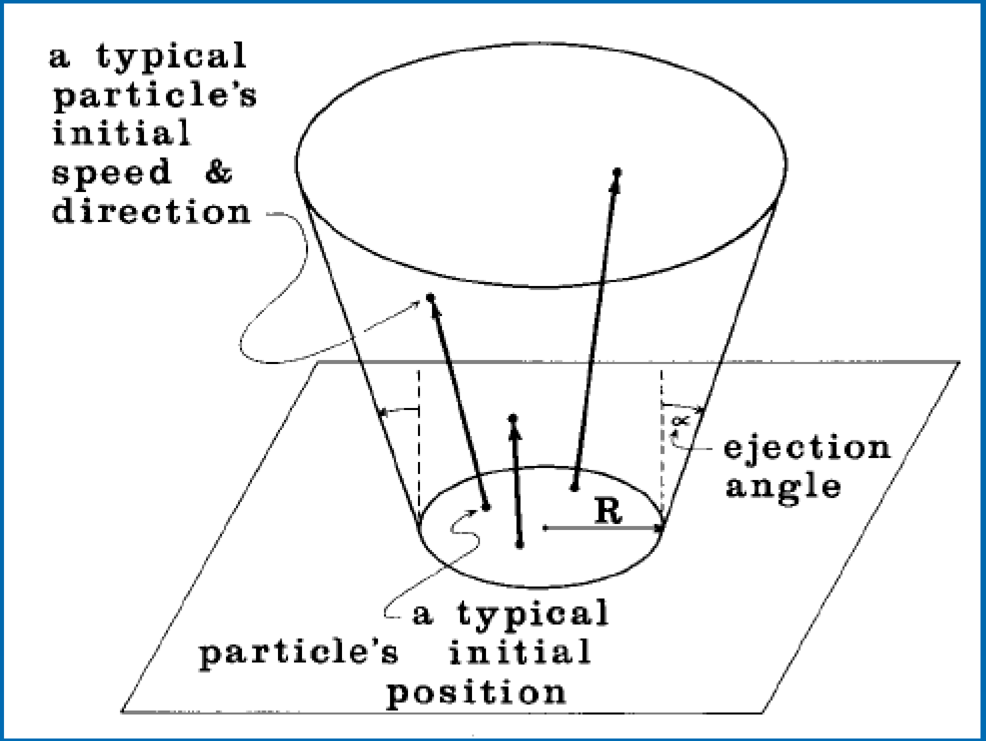
- Genesis Demo from StarTrek II: The Wrath of Khan (Paramount, 1992) – sequence generated in Lucasfilm – dead planet changed to alive by Genesis bomb explosion – after the explosion, walls of fire spread from the impact point, mountains and other terrain features are born
- 2-level particle system with the centre in the impact point, concentric circles of two levels

Wall of fire and explosion (2):



Distribution of particle systems on the planet surface

Wall of fire and explosion (3):



2nd level particles system (its appearance immitates an explosion)

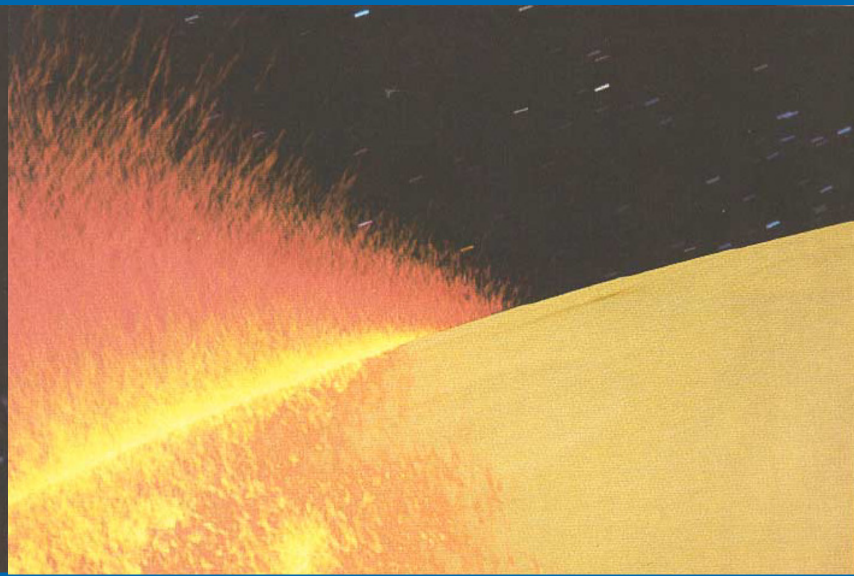
Wall of fire and explosion (4):

Initial explosion

Wall of fire and explosion (5):

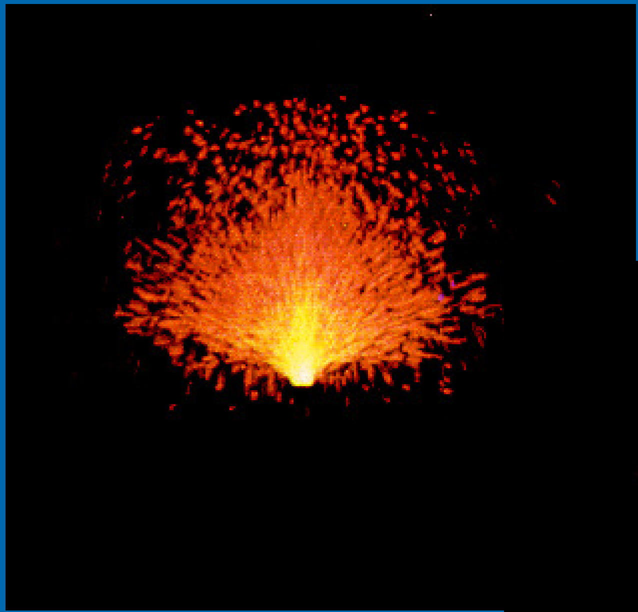
Proceeding firewall

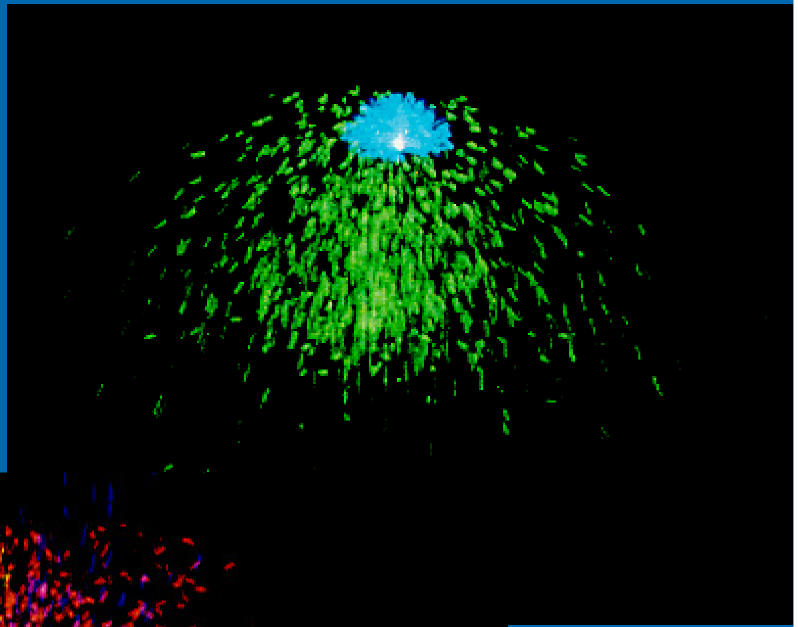
Wall of fire and explosion (6):



Further examples from the same application

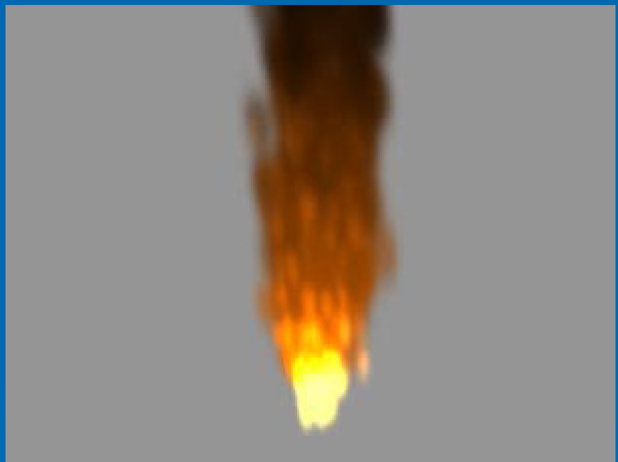
Firework:





Grass:

3D Max: sprey, snow, superspray, blizzard, movement can be further influenced bz the so-called space-warps, materials can be attached and modified according to the particle age



2 particle systems – fire and smoke (intensity is added to or subtracted from the background)

Waterfall – particles are reflected from the object

Snow: the user defines objects instead of particles

Forrest: the area is subdivided into squares, in each a centre is randomly shifted but only inside the square, about ten per cent of points are eliminated, in the other a simple tree model from several cylinders with a suitable texture is generated, the result is OK from a sufficient distance

Dynamic simulation: a basket of balls spread on stairs, the emitting object – the basket, particles - the balls. The ball properties – initial velocity, direction, flexibility, color (texture), mass, all balls move downstairs, they can collide, bounce from the balustrade, etc.

Bubbles in soda water: the inner surface of the glass generates them, they have a different size, direct upwards, die on the surface

Useful also for complex physical simulations – particle tracing: to model a complex physical field with a limited accuracy in case of limited time for computation, we visualize the particle trajectories, we accent , e.g., the places with big trajectory changes and so with bigger friction by red

Plant growth simulation: they avoid obstacles, react on the light