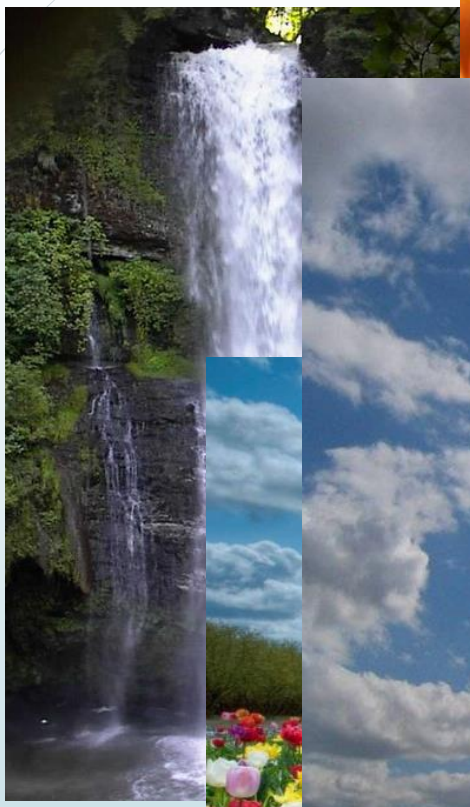
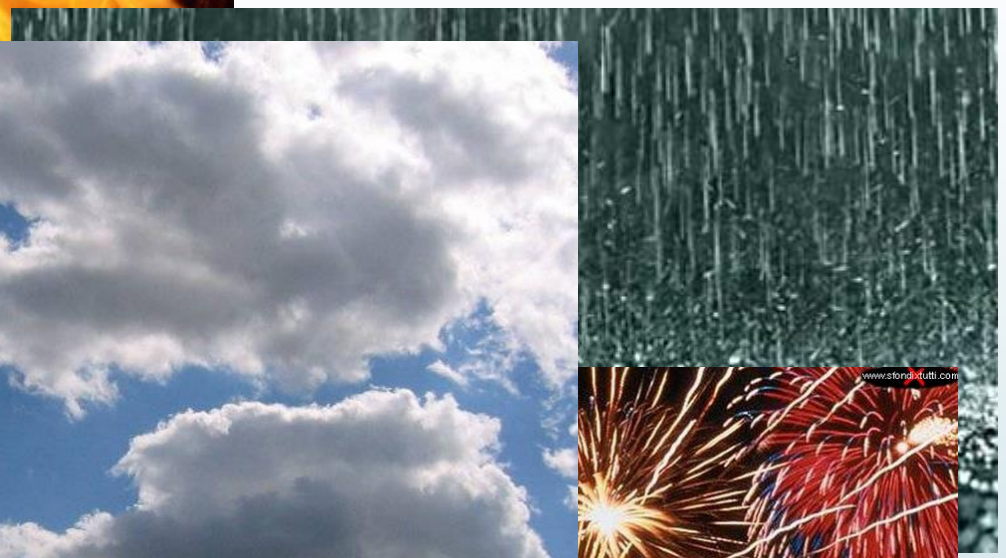
Sistemi Particellari

Corso di Fondamenti di Computer Graphics - 2014/2015

In Natura





Due necessità

- Aumentare il livello di irregolarità e di dettaglio
- Semplificare

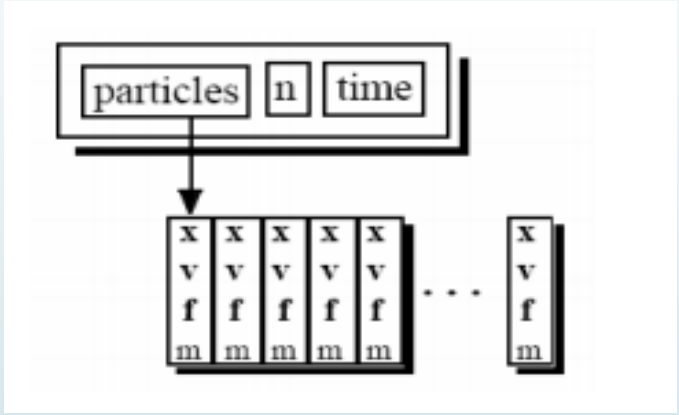
Sistemi Particellari

- Non poche primitive grafiche che definiscono una superficie
- Ma un numero molto elevato di componenti elementari che riempiono un volume

Ogni particella

- Posizione
- Velocità
- Dimensione
- Colore / Texture
- Trasparenza
- Forma
- Tempo di vita

Il Sistema non è altro che l'insieme di queste particelle

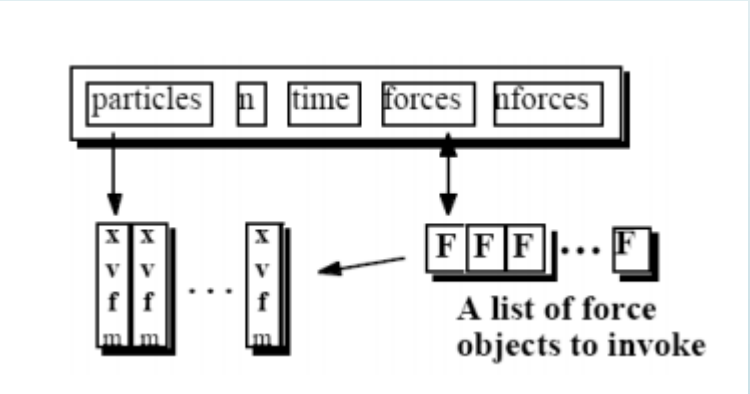



```
struct Particle{
    glm::vec3 pos, speed, acceleration, forceAccumulator;
    unsigned char r,g,b,a; // Color
    float size, weight;
    float lifespan;
};
```

In ogni istante

- Nuove particelle vengono generate
- Quelle che hanno superato il tempo di vita vengono eliminate
- Le particelle possono variare ciascuno dei loro attributi in modo stocastico o per

Azione di Forze




```
struct Force{
    glm::vec3 forceComponents;
    Particle[] particles;
};
```

La più semplice F = m * a (Newtonian Particles)

Tipi di Forze:

- Unarie
  - Gravità
  - Campi elettrici
  - Vento
- Interagenti tra due o più particelle
  - Repulsione
  - Urti
- Locali, dipendenti dalla posizione
  - Presenza di altri oggetti nella scena

Metodo di Eulero

phase space

$$\ddot{\mathbf{x}} = \frac{\mathbf{f}}{m}$$

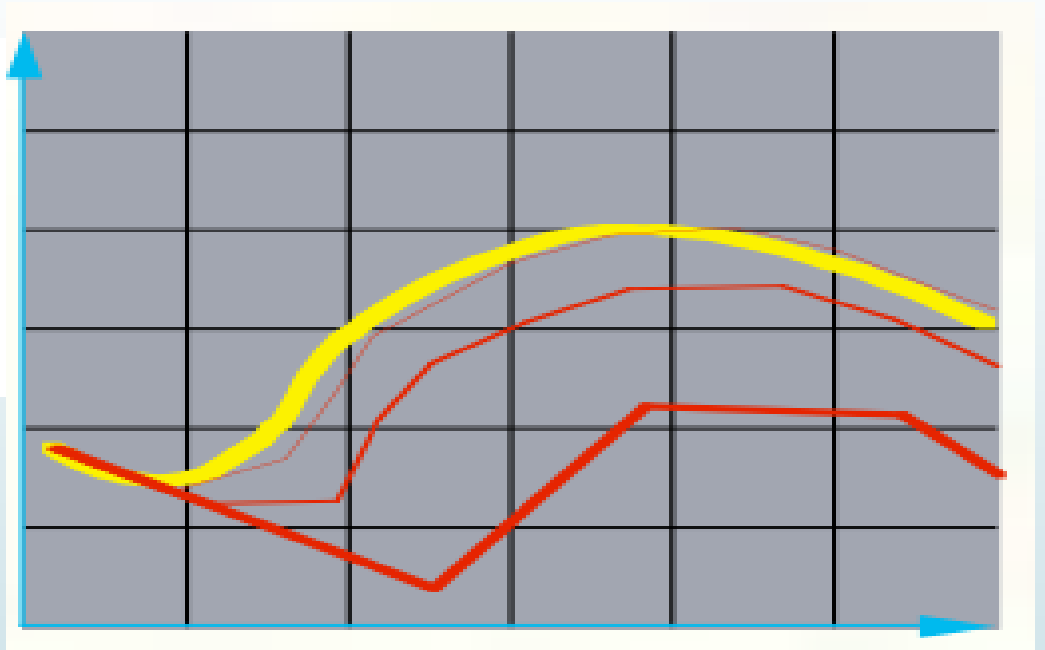
$$\dot{\mathbf{x}} = \mathbf{v}$$

$$\dot{\mathbf{v}} = \frac{\mathbf{f}}{m}$$

2nd order ODE

1st order ODEs

ODE: "ordinary differential equation"



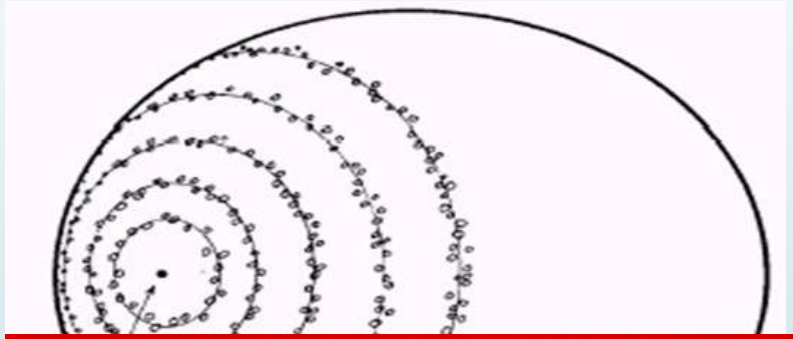
$$\mathbf{x}(t_1) = \int_{t_0}^{t_1} \dot{\mathbf{x}}\, dt + \mathbf{x}(t_0)$$

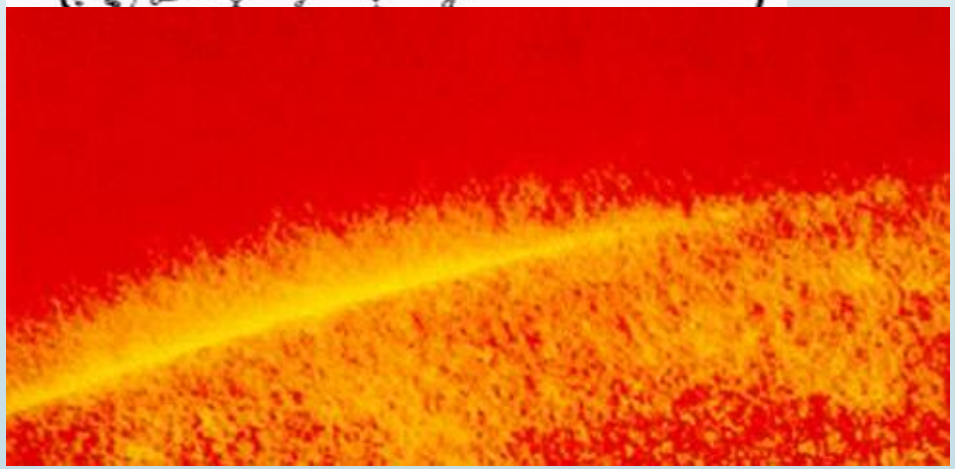
$$\vec{\mathbf{x}}(t + \Delta t) = \vec{\mathbf{x}}(t) + \Delta t \cdot \dot{\vec{\mathbf{x}}}(t) = \vec{\mathbf{x}}(t) + \Delta t \cdot g(\vec{\mathbf{x}}, t)$$

```
int ParticlesCount = 0;
for(int i=0; i<MaxParticles; i++){

        P
        i
                                                                    ta;
        }
        }
        P
}

void                                                                );
                }
}
```

Prima applicazione

- William T. Reeves, Star Trek – L'ira di Kahan





- Generation Shape bordo di una sfera

- Particelle puntiformi emittenti luce
  - Il colore finale del pixel è dato dalla somma dei contributi

Possibili Problemi Aggiuntivi

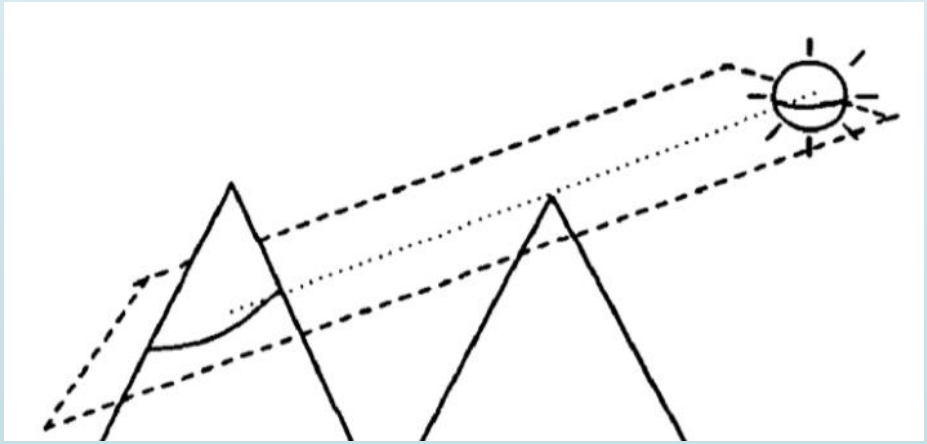
- Particelle non puntiformi ma tridimensionali
  - *Shading*
  - Calcolo della superficie visibile
- Interazione con oggetti generati con tecniche tradizionali di surface rendering
- Sistemi di Sistemi Particellari

Il caso di un prato d'erba

Illuminazione

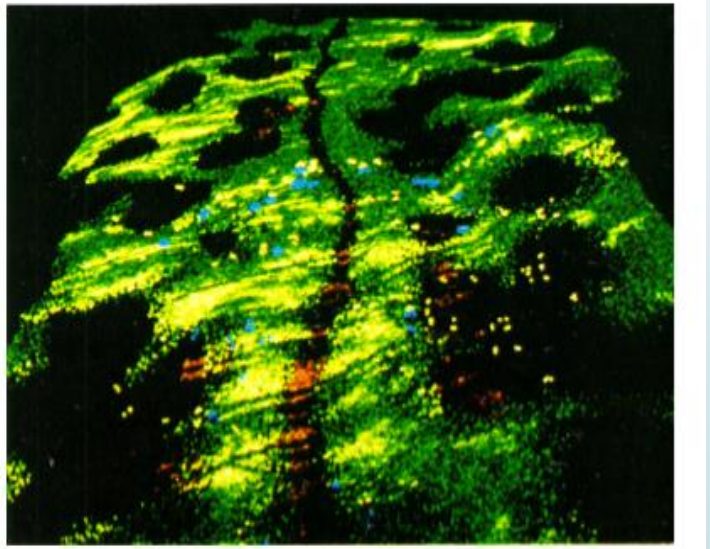
- Le particelle sono diffusive, sarebbe teoricamente possibile calcolare in modo esatto l'illuminazione con tecniche tradizionali, ma dato l'altissimo livello di dettaglio sarebbe computazionalmente oneroso.

→ Tecniche approssimate



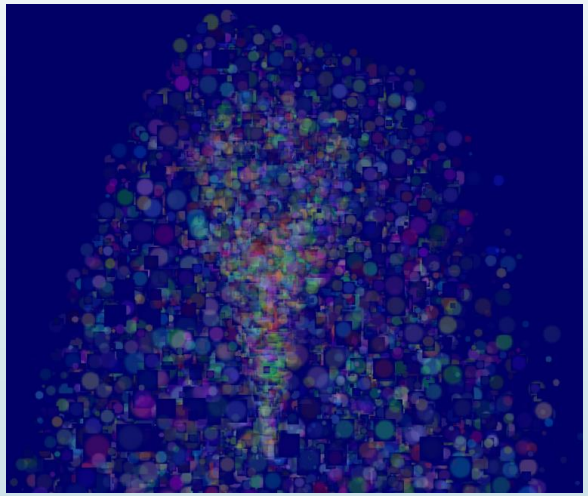
Interazione con altri oggetti che generano ombre

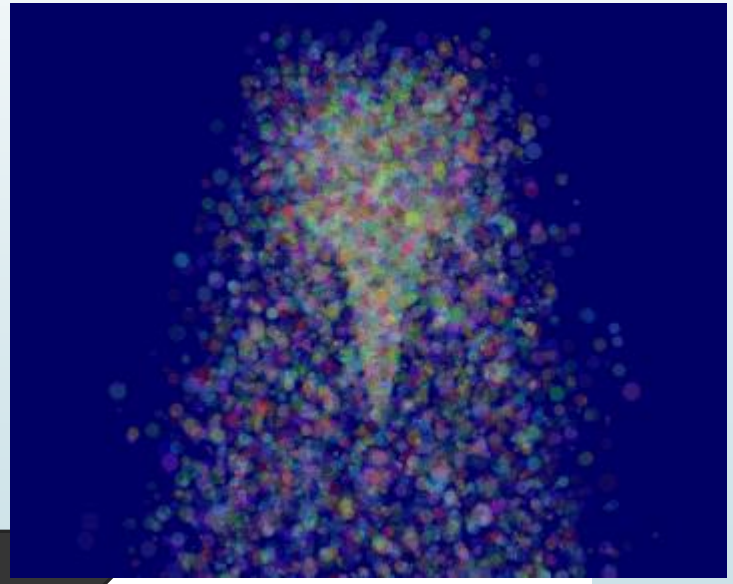
- Shadow mask



Superficie visibile

- Necessario un ordinamento delle particelle





```
void SortParticles(){
    std::sort(&ParticlesContainer[0], &ParticlesContainer[MaxParticles]);
}
```