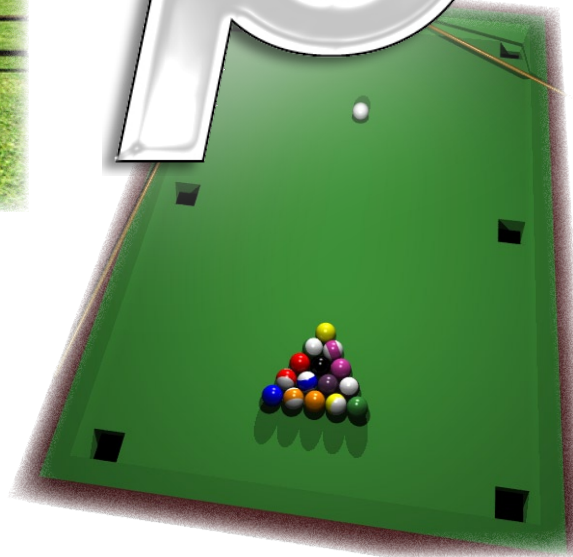
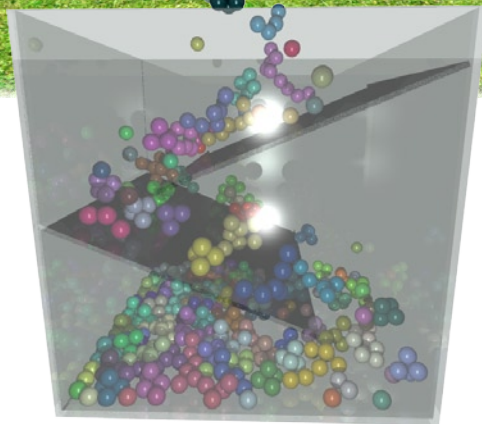
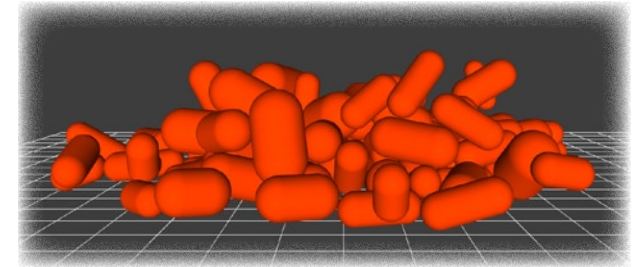
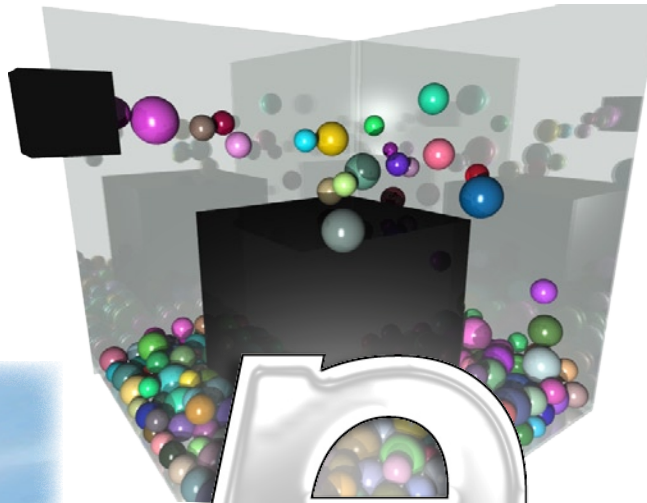


# The $\epsilon$ Rigid Body Physics Engine

asim  
2007

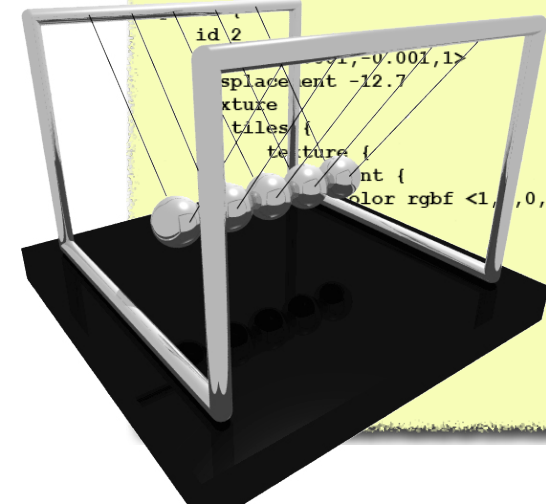


# $\epsilon$

```
//Sphere 1 with a red color texture
sphere {
  id 1
  position <2.0,3.5,-1.6>
  radius 3.2
  density 1.2
  linear <0.0,0.2,0.3>
  angular <0.1,0.2,0.0>
  texture {
    pigment {
      color rgb <1,1,0>
    }
  }
}
```

```
//Plane 2 with a tiled texture
```

```
id 2
  position <1,-0.001,1>
  displacement -12.7
  texture {
    tiles {
      texture {
        pigment {
          color rgbf <1,0,0.8>
        }
      }
    }
  }
}
```



# The $\oplus$ Rigid Body Physics Engine



The  $\oplus$  Rigid Body Physics Engine  
 Klaus Iglberger, Martin Ketzer, Hannes Wengenroth, Ulrich Rude  
 University of Erlangen-Nürnberg, Computer Science 10 LSS



## Features

- Focus on accurate rigid body simulations
- Störmer-Verlet time discretization
- Quaternions for the rotation treatment
- Collision handling
- Simulation of arbitrary complex rigid bodies
- 4 primitive geometries
- Dynamic agglomeration and rupture
- Calculation of contact forces and torques
- Motion constraints
- Easy to use C++ interface/framework
- Automatic memory management
- Description language
- Ray-Tracing visualization with POV-Ray
- Real-time visualization system

```

#include <vector>
#include <quintic>
using namespace p;

int main()
{
    World world = CreateWorld();
    world.CreateSphere( 0.5, 0.5, 1.0 );
    Sphere sphere = world.CreateSphere( 1, 2.0, 0.5, 1.4, 3.2, 0.8 );
    Plane plane = world.CreatePlane( 2, 0.01, -0.01, 1.0, 12.7 );
    RayTracer ray = CreateRayTracer();
    ray.AddObject( sphere );
    ray.AddObject( ColorSegment( 1, 0 ) );
    ray.AddObject( Plane );
    ray.AddObject( ColorSegment( 0, 0, 0 ) );
    ray.Render();
}
                
```

Features of the  $\oplus$  rigid body physics engine

## Collision handling

- Simulation time step
- Resting/Colliding contacts
- Force calculation

## Collision handling

- Apply gravity to all rigid bodies
- Find all contact points between the rigid bodies
- Adjustment of the coefficient of restitution for all contacts
- Treatment of the colliding contacts ( $v_{rel} < 0$ )
- Treatment of the resting contacts ( $v_{rel} = 0$ )
- Move the rigid bodies object

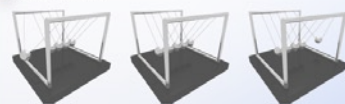
- Determination of all contact points at a certain point in time
- Distinction of colliding and resting contacts
- Treatment of the resting contacts: calculate contact forces to prevent the rigid objects from penetrating each other

$$\mathcal{N} + b, \text{ where } a_i \geq 0, f_i \geq 0 \text{ and } f_i a_i = 0 \text{ (} f_i a_i = 0 \text{)}$$

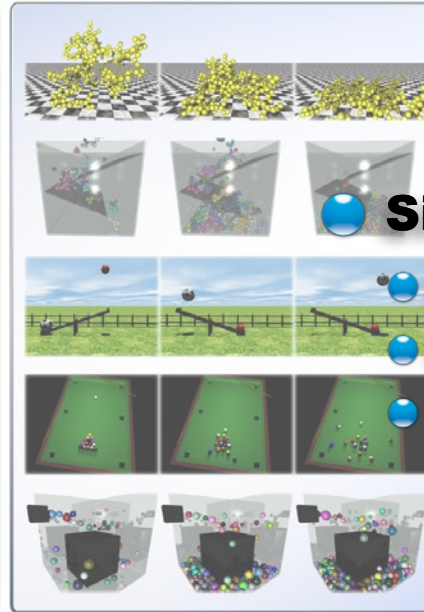
$$\text{Quadratic program } \rightarrow \min f^T (\mathcal{N} + b) \text{ subject to } \begin{cases} \mathcal{N} + b \geq 0 \\ f \geq 0 \end{cases}$$

- $a$  Relative acceleration at the  $n$  contact points
- $f$  Acting force at the  $n$  contact points
- $a_i$  Relative acceleration at contact point  $i$
- $f_i$  Acting force at contact point  $i$
- $A$  Masses and contact geometries of the rigid bodies
- $b$  External and inertial forces in the system

- Similar treatment of the colliding contacts
- Newton's cradle:



## Simulation examples



Simulation examples

Billard simulation

Object generator

...

## Current work / new extensions

### Current work

- Treatment of static and dynamic friction
- Performance optimization/parallelization
- Still open for Bachelor-, Master- and Diplomatheses

