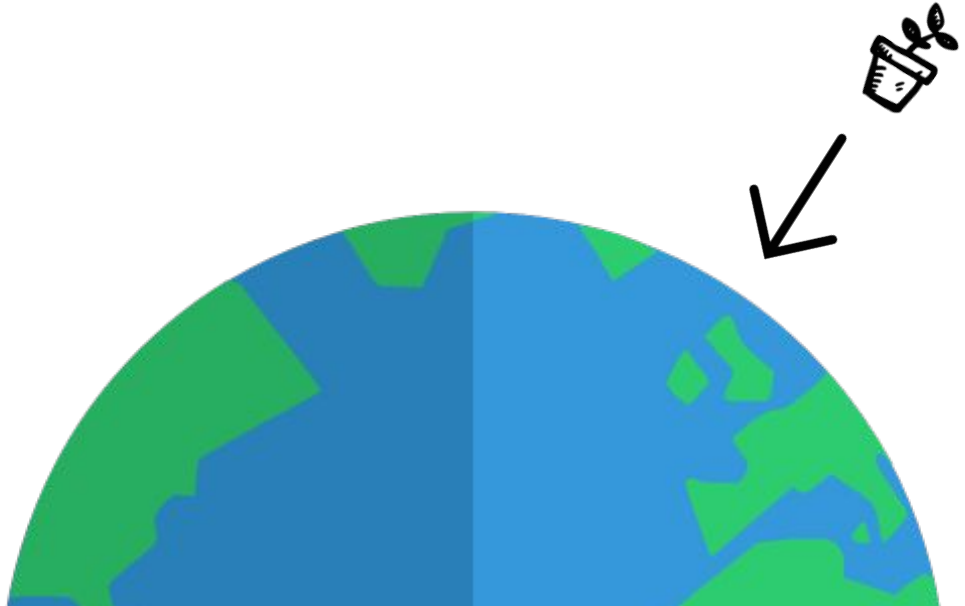
# Real-Time Rendering: Physics

Bauhaus-Universität Weimar

# Gravity

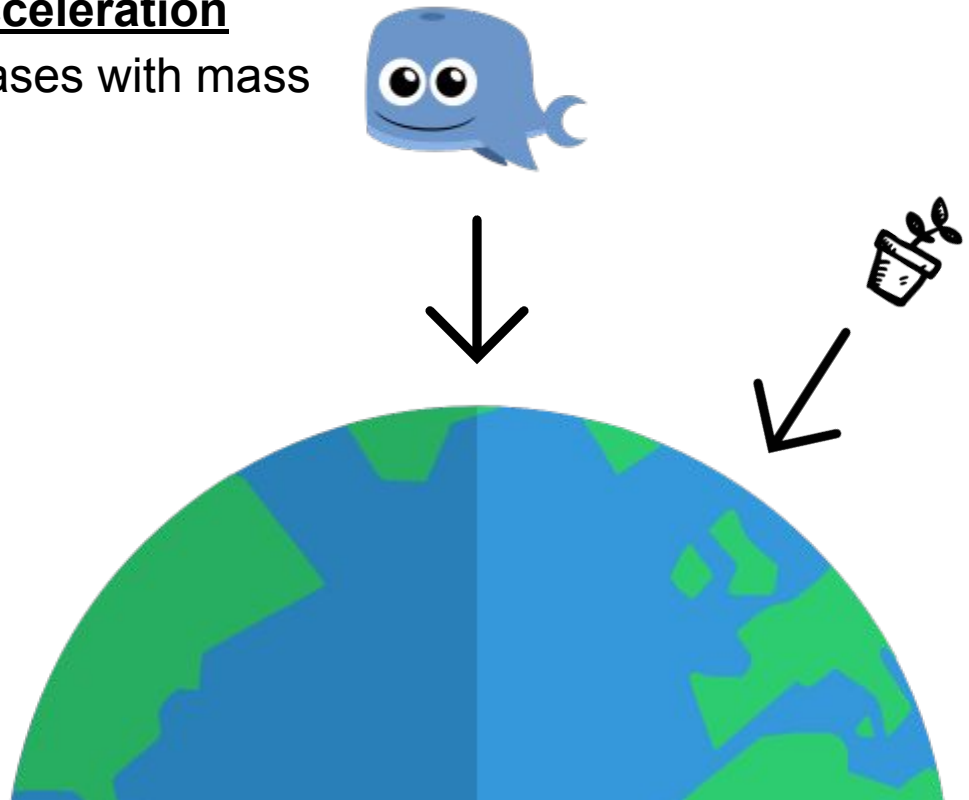
## Real-Time Rendering: Physics

# What is Gravity?



# What is Gravity?

- ➔ Constant **<u>acceleration</u>**
- ➔ Force increases with mass

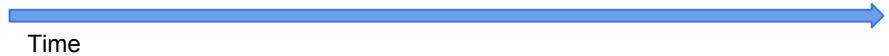


# Velocity & Acceleration

## Velocity

➔ Change of position each frame (timestep)

➔ Rate of change = **speed**



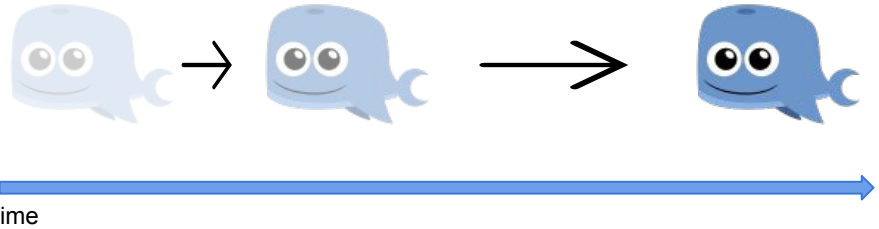
# Velocity & Acceleration

## Velocity

- ➔ Change of position each frame (timestep)
- ➔ Rate of change = **speed**

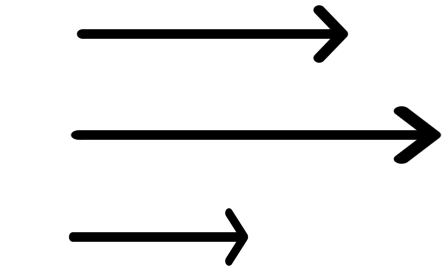
## Acceleration

- ➔ Change of speed each frame
- ➔ Gravity = acceleration towards earth



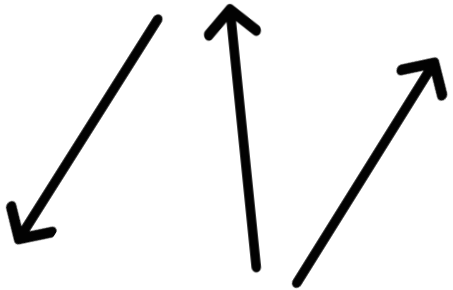
# How to code?

- ➔ Position is point in space
- ➔ Velocity and acceleration as **vectors**
  Which hold **direction** & **magnitude**



magnitude

- ➔ Add acceleration vectors to velocity
- ➔ Add velocity vectors to position



direction

# How to simulate Gravity?

Objects have…

➔ Position
➔ Velocity (speed & direction)

For each step / iteration…

➔ Update an object’s velocity by adding gravity’s acceleration ‘downwards’
➔ Update an object’s position with new velocity
➔ Render object at new position

# Particle Systems

## Real-Time Rendering: Physics

Bauhaus-Universität Weimar

# Particle Systems

**Particles**

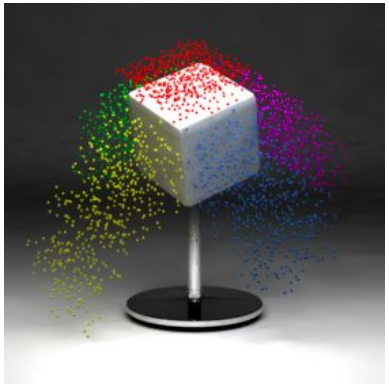
- ➔ Points, meshes, sprites
- ➔ Mass, velocity, position, lifetime

**Emitters**

- ➔ Particle sources
- ➔ 1D point, 2D shapes, 3D objects
- ➔ Specifies a system's properties

Static

Animated

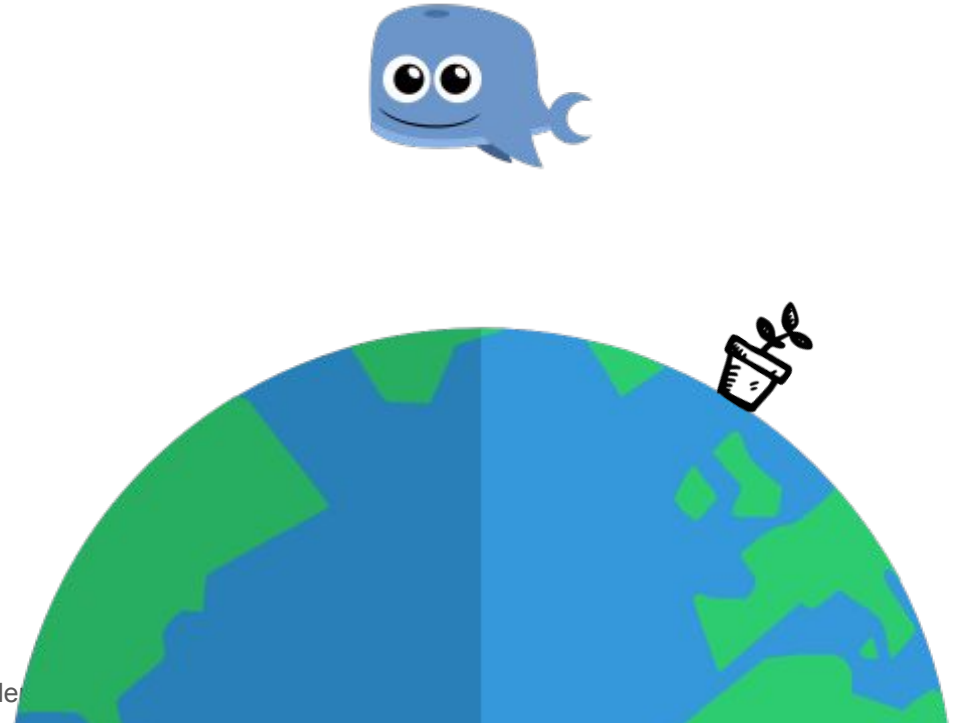


By Halixi72 at the English language Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=58485548

# Collision Detection

## Real-Time Rendering: Physics

# Collision Handling



# Collision Handling

➔ Collision Detection

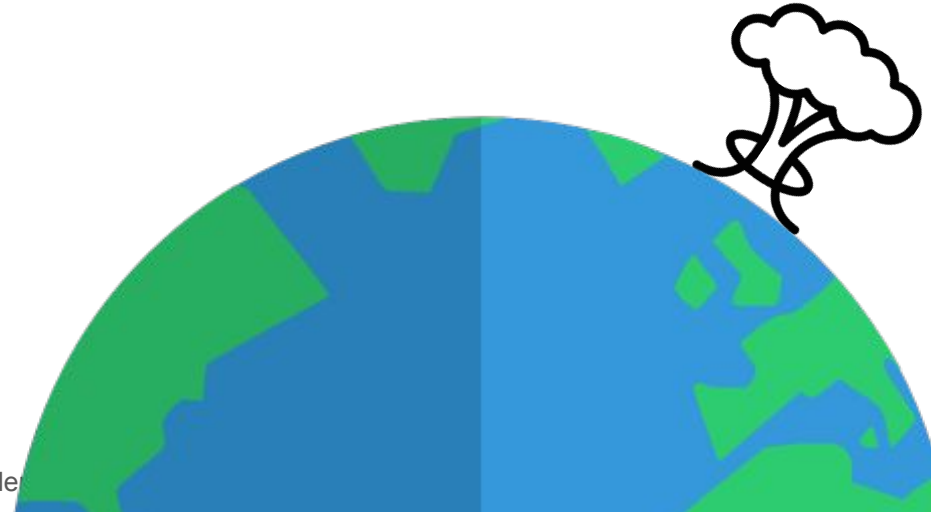
# Collision Handling

➔ Collision Detection

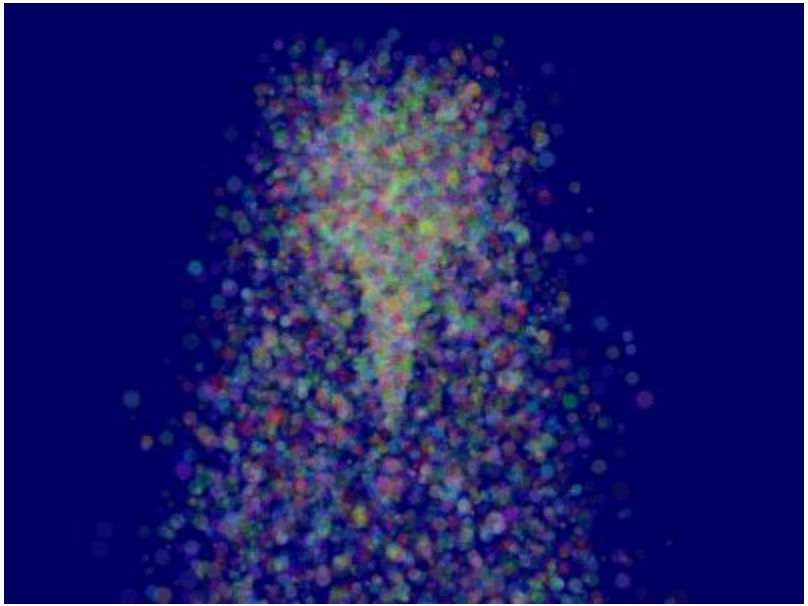
➔ Collision Determination

# Collision Handling

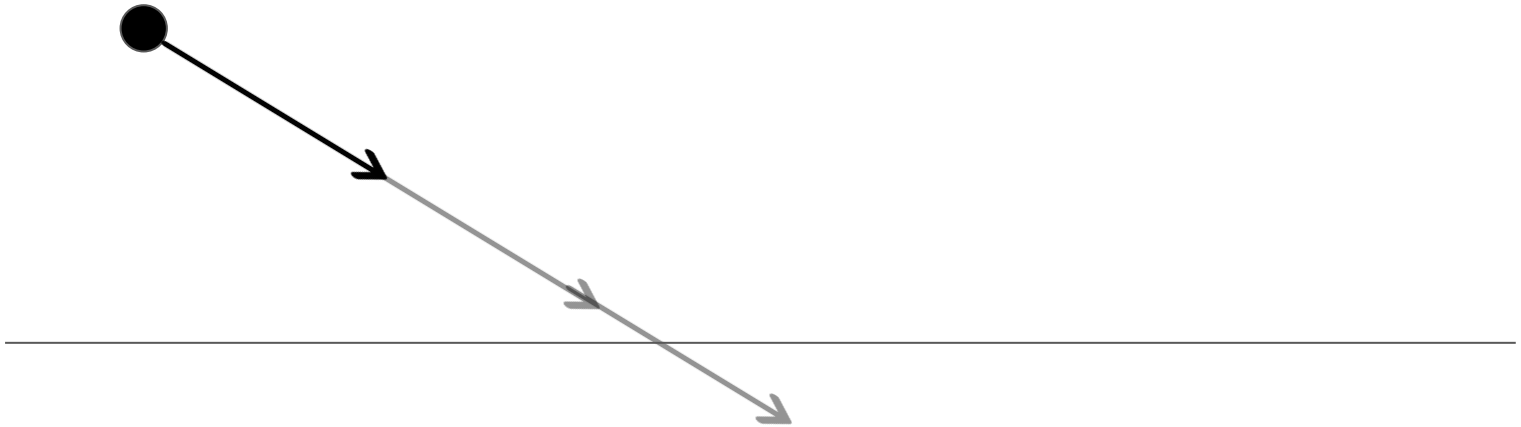
- ➔ Collision Detection
- ➔ Collision Determination
- ➔ Collision Response



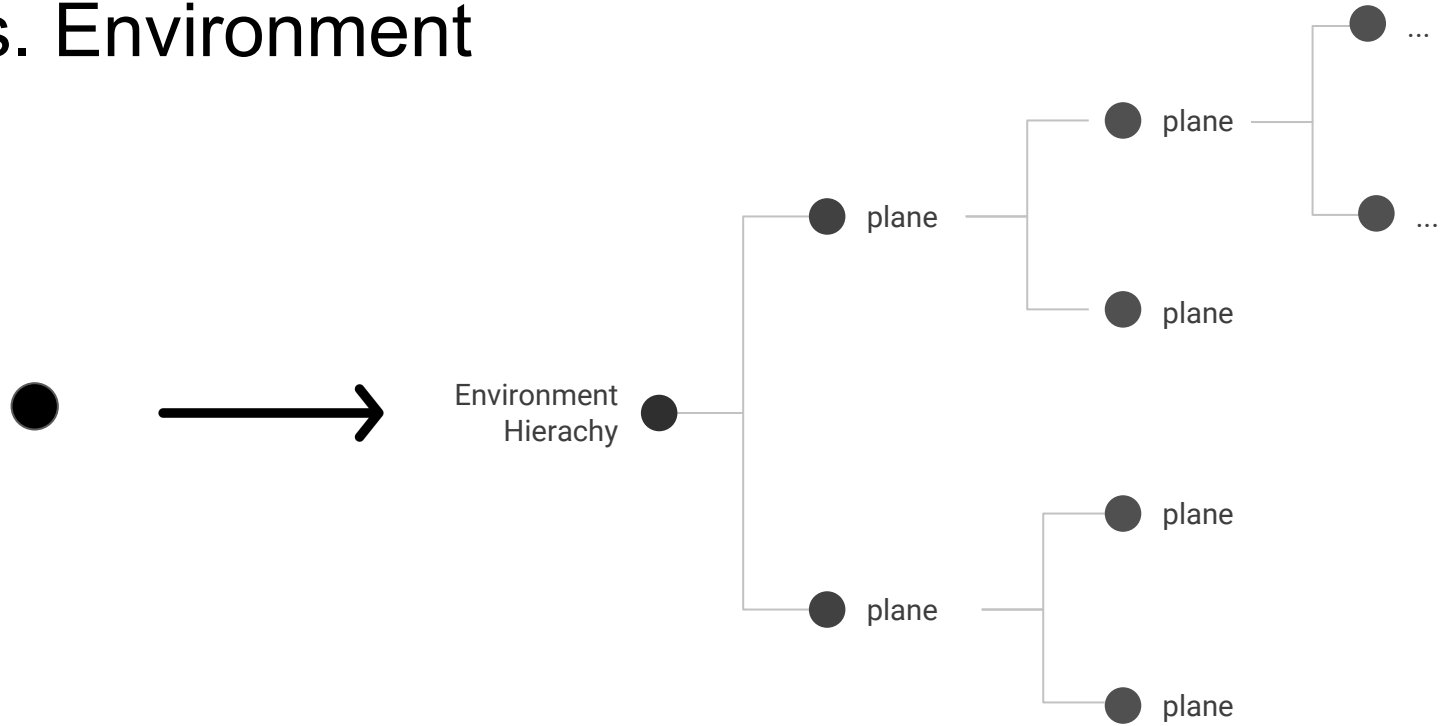
# Back to particles...



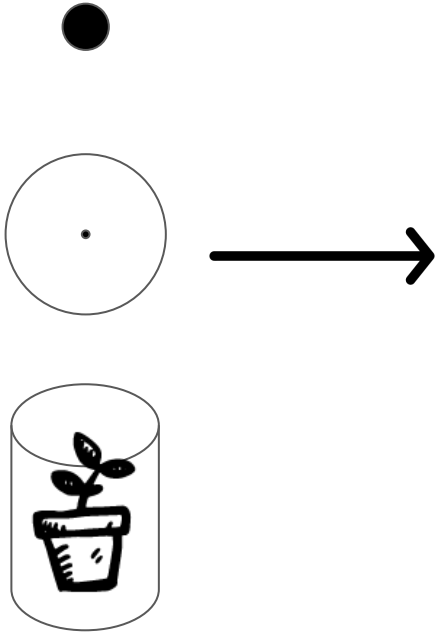
# Using a Particle's Velocity Vector

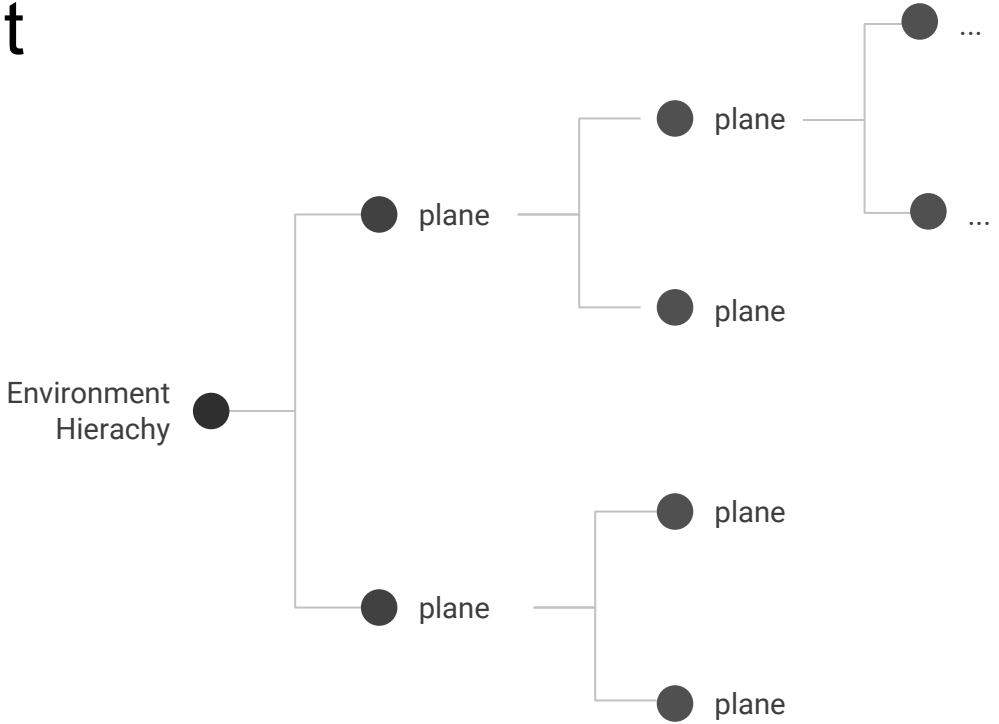


# Point vs. Environment



# Point vs. Environment

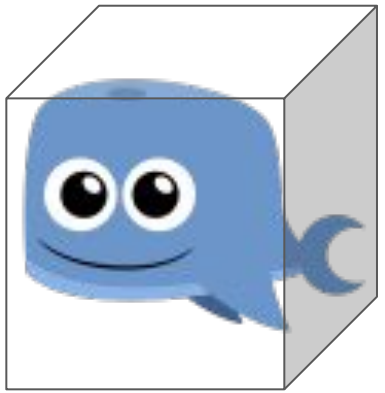




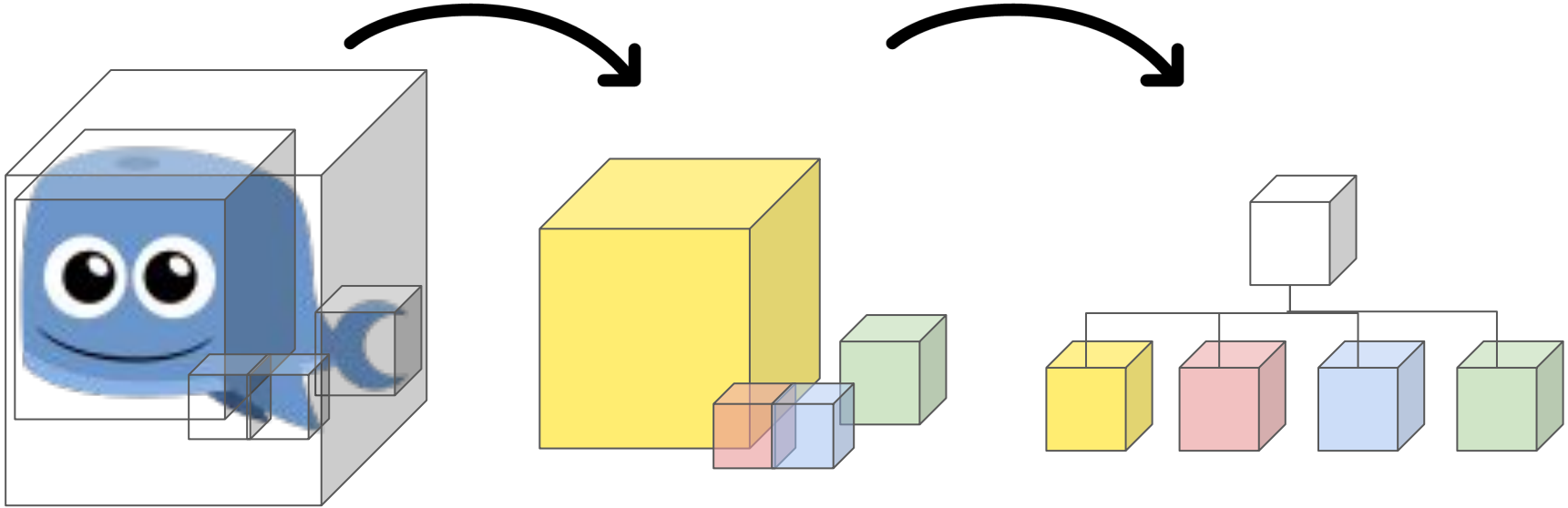
# Bounding Volumes

➔ Approximate complex objects with simple volumes

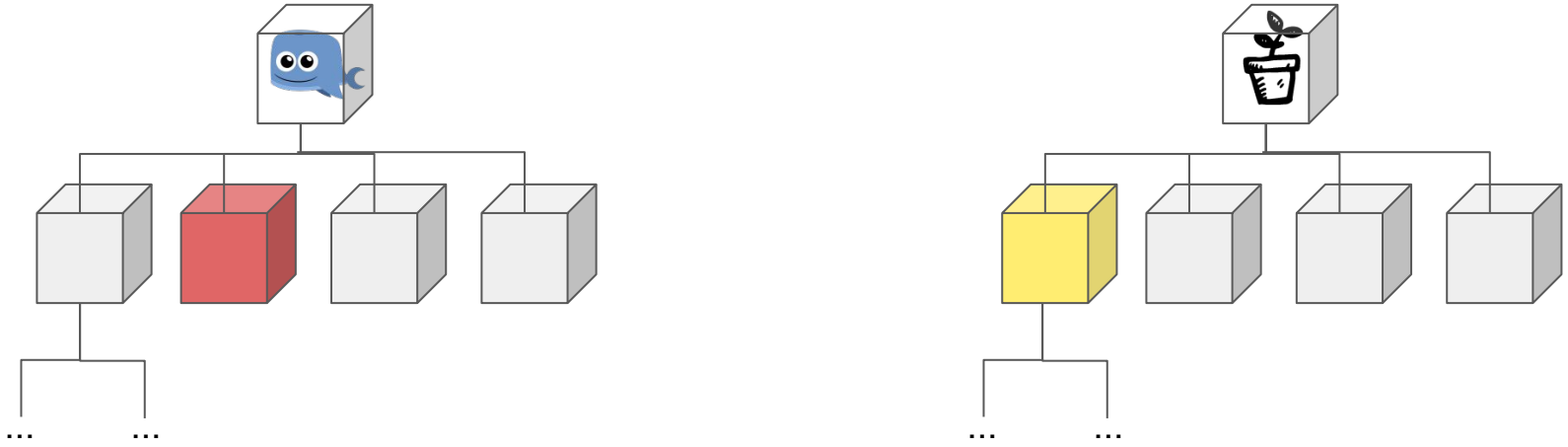
➔ Calculate collision of volumes instead of time-intensive collision of objects



# Bounding Volumes and Hierarchies



# Colliding Hierarchies



# But Scenarios get more complex...

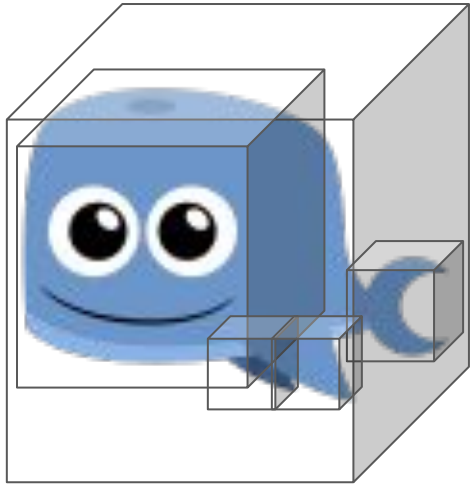
When objects in our scene are…

- ➔ Complex
- ➔ Moving
- ➔ Flexible
- ➔ Self-colliding
- ➔ Numerous
- ➔ ...

# Space Partitioning

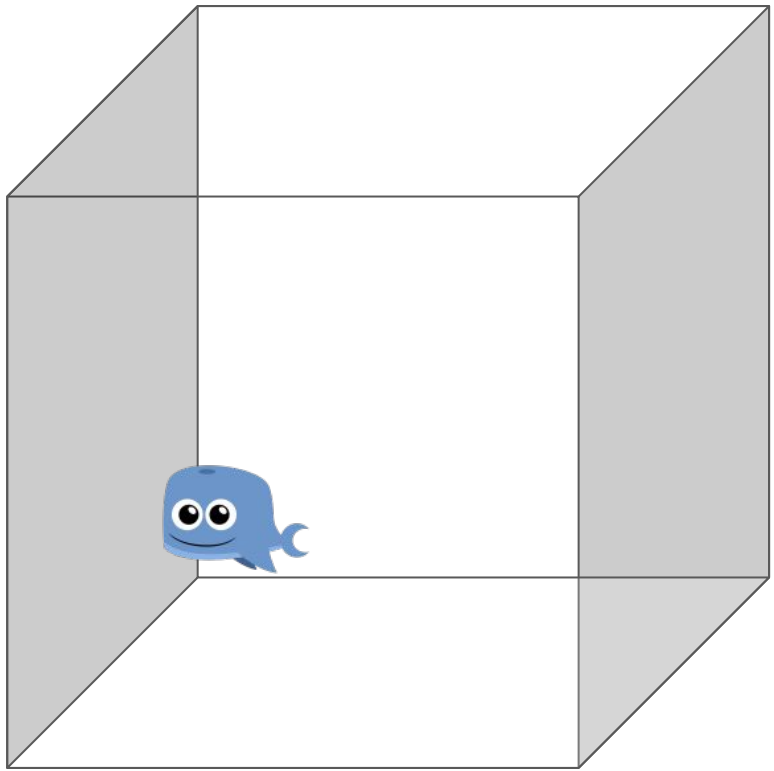
## Real-Time Rendering: Physics

# Remember Bounding Volumes?

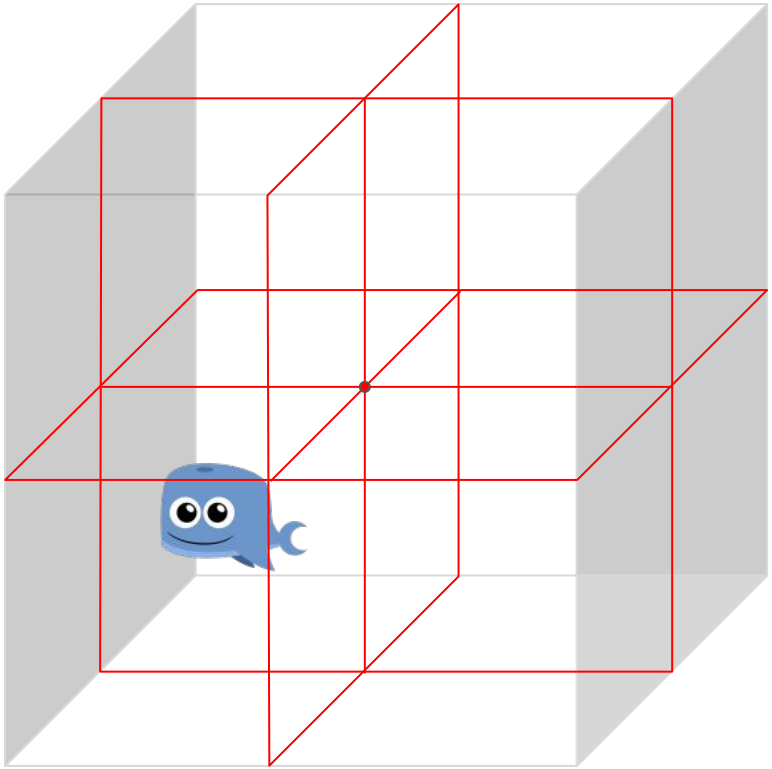


# Dividing Space

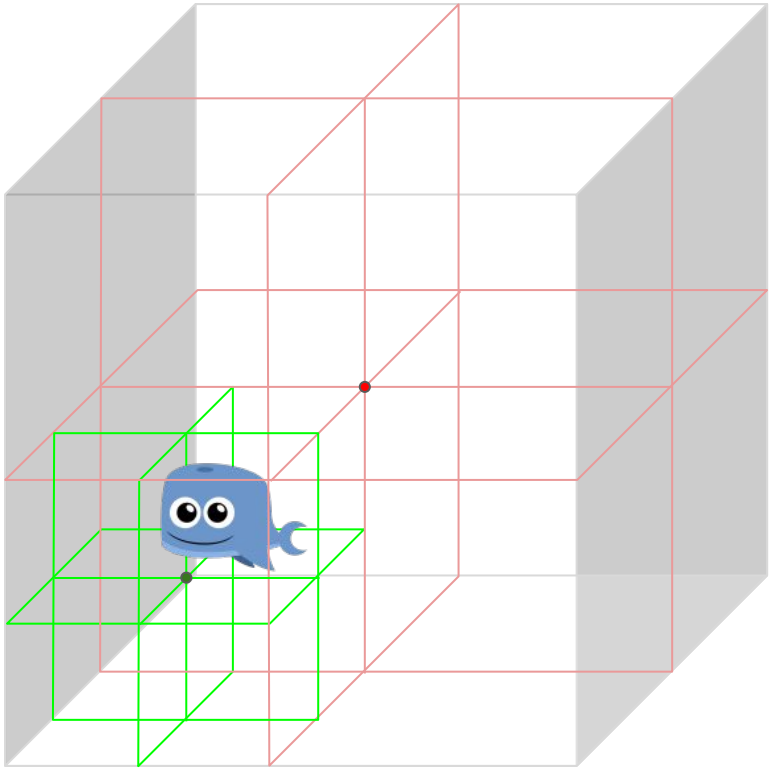
… so that objects in disjoint regions cannot collide



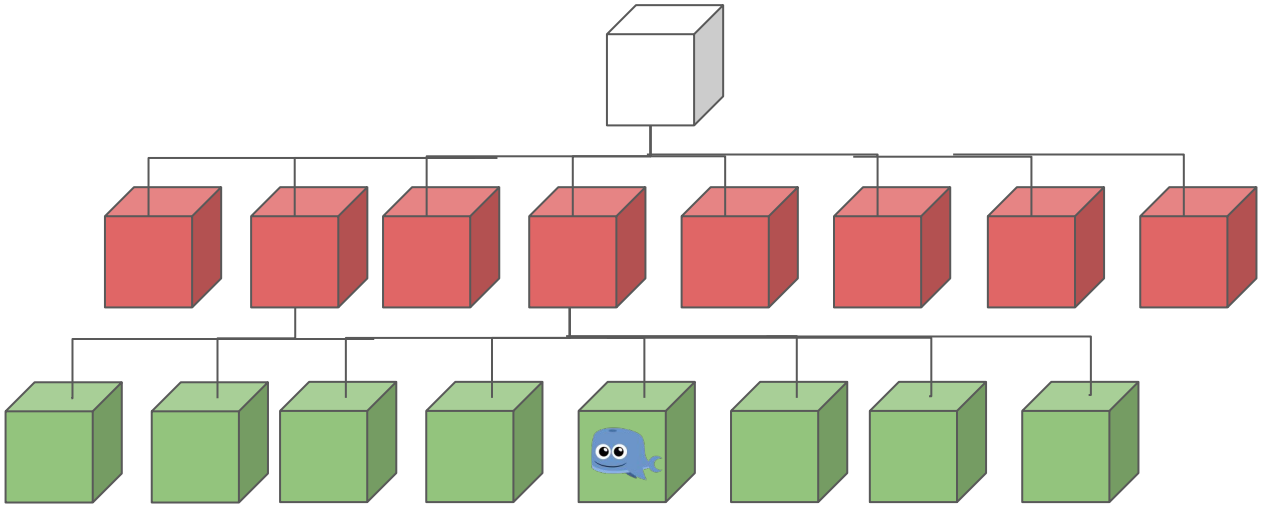
# Dividing Space

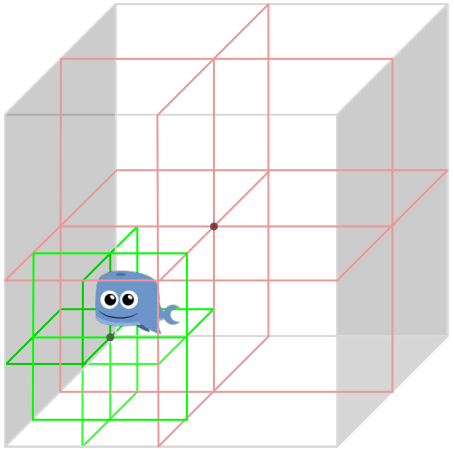


# Dividing Space

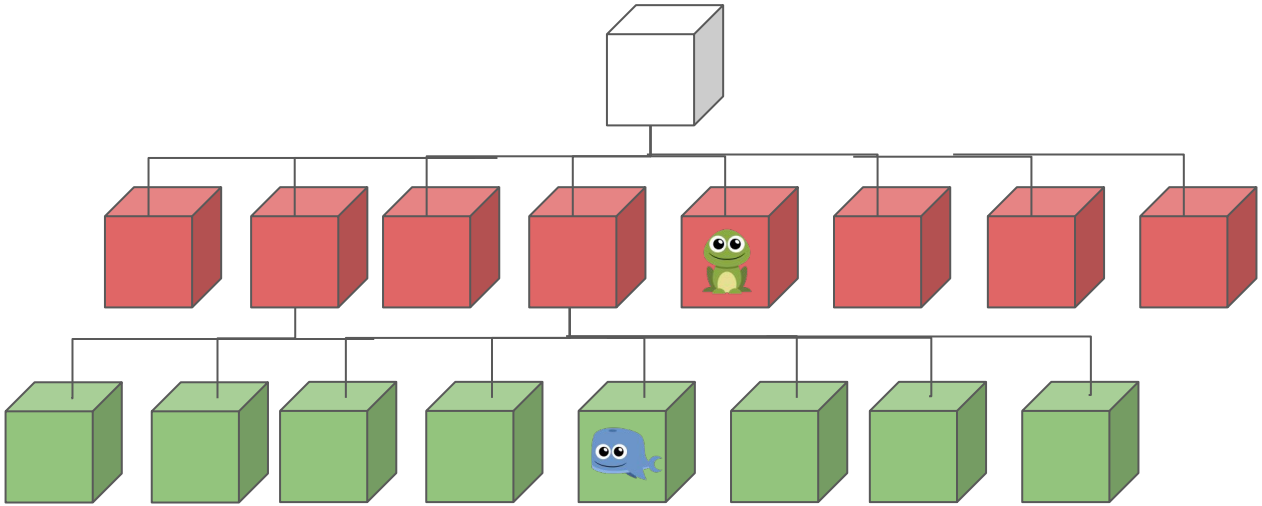


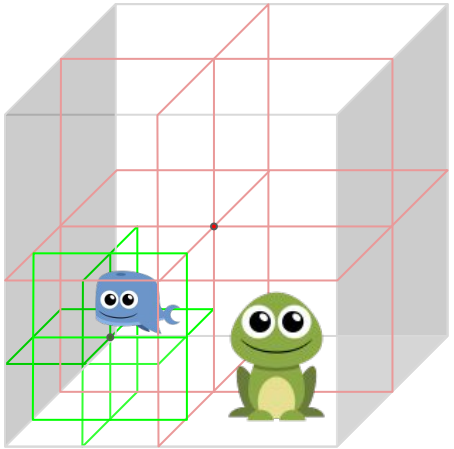
# Dividing Space - Octree



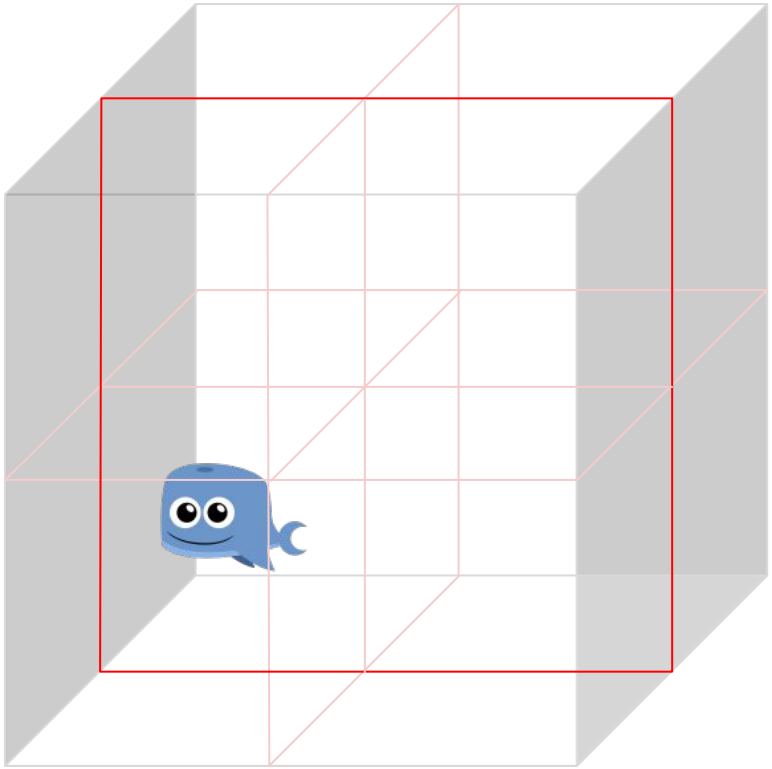


# Dividing Space - Octree

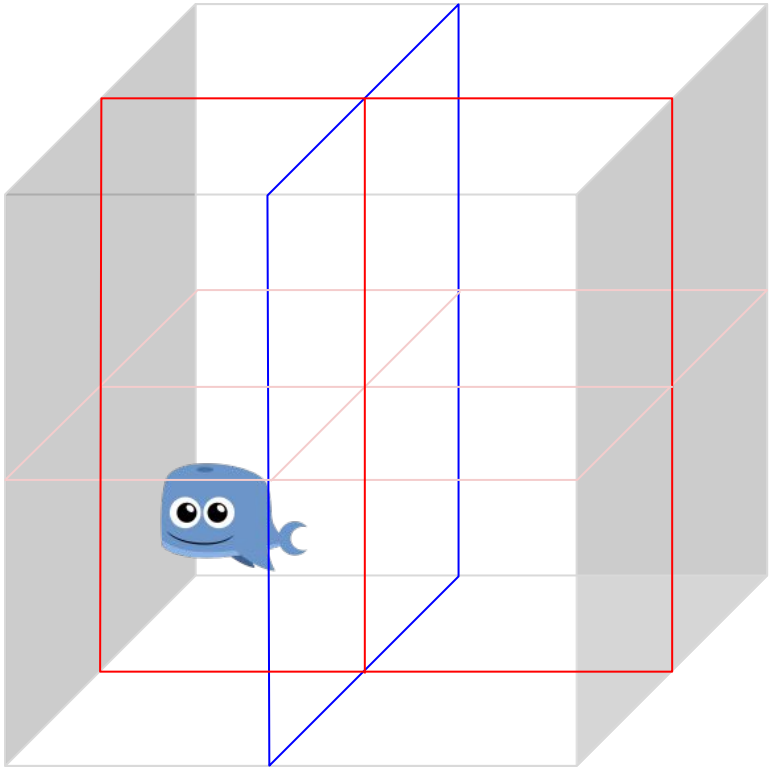




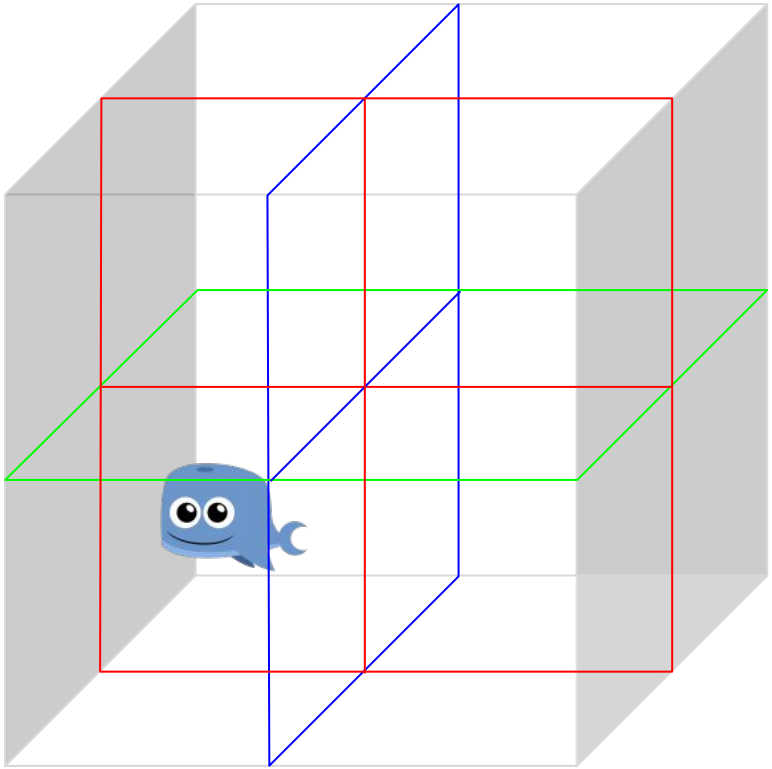
# Dividing Space



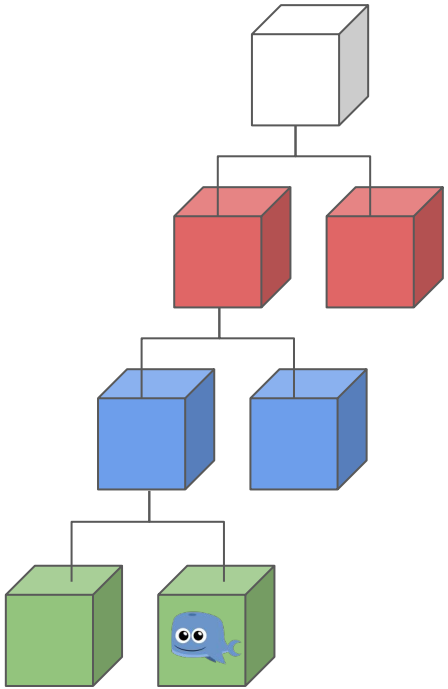
# Dividing Space

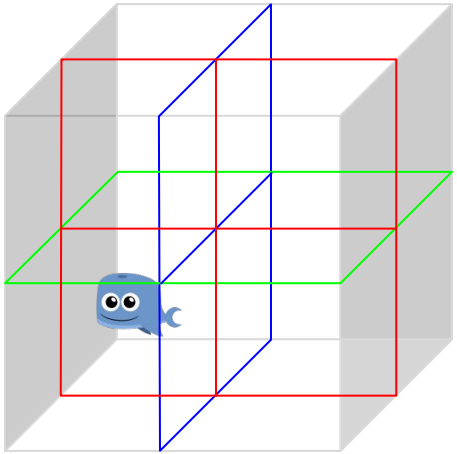


# Dividing Space

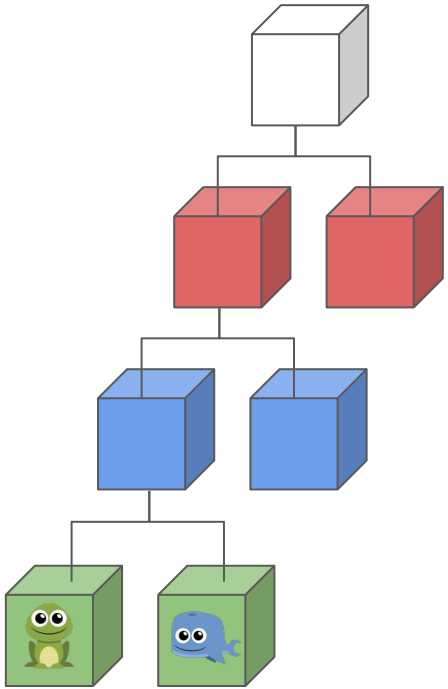


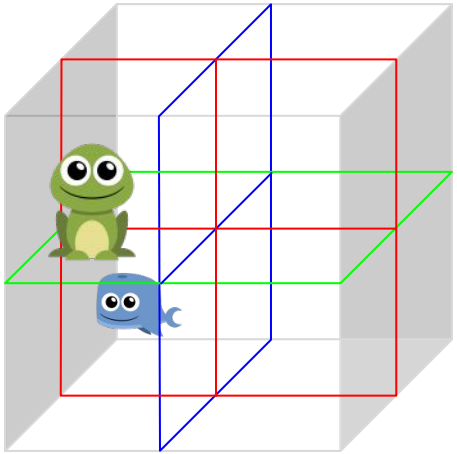
# Dividing Space - 3D-Tree (kD-Tree)





# Dividing Space - 3D-Tree (kD-Tree)





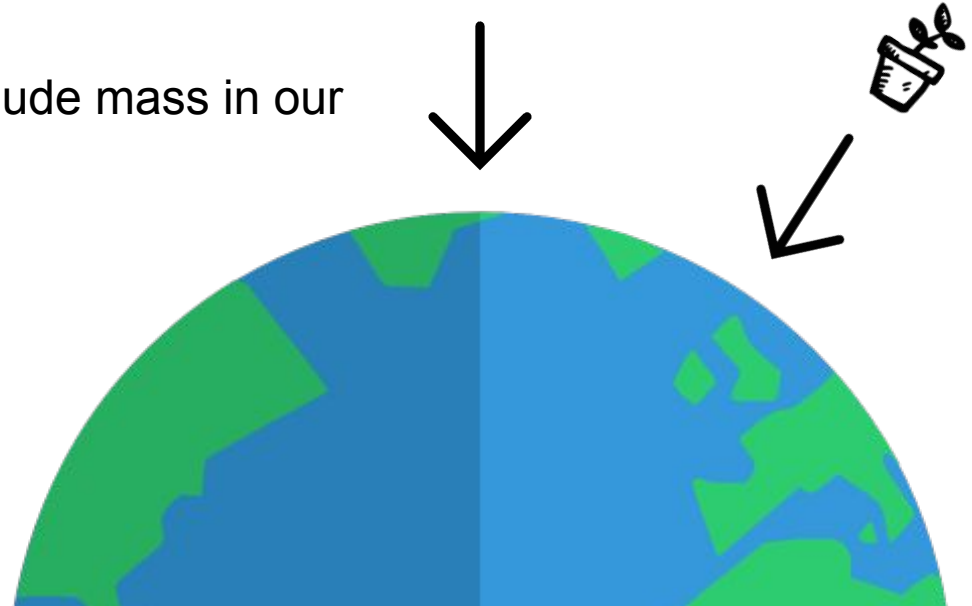
# Body Physics - Forces

Real-Time Rendering: Physics

# Recap Gravity

- ➔ Constant acceleration
- ➔ Force increases with mass

- ➔ Now we include mass in our System!



# Some (Basic) Formulas...

Force, acceleration and mass:

*Acceleration of mass*

```
f = m * a
```

Momentum, velocity and mass:

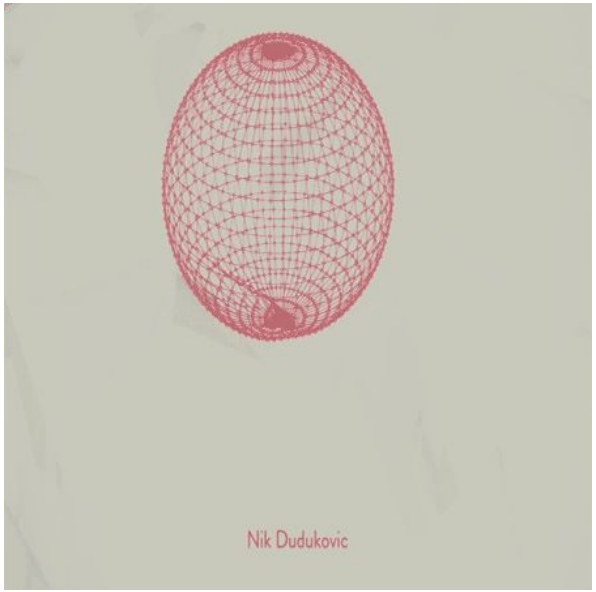
*Energy of moving mass*

```
P = m * v
```

➔ Calculate forces, velocity and acceleration to update an object's position

# Internal vs. External Forces

external




External & internal

# Rigid Bodies (external forces)

Real-Time Rendering: Physics

# Simulated Forces

- Gravity
- Friction
- Normal Force
- Collisions
- Impulsive Forces
- Persistent Forces
- Angular Forces
- ...

# Simulated Forces

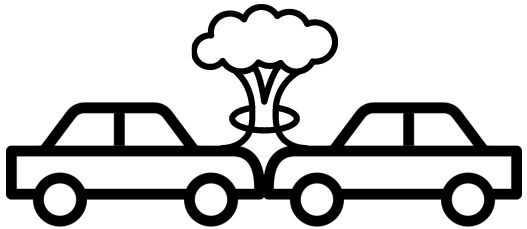
- ➔ Gravity
- ➔ Friction
- ➔ Normal Force
- ➔ Collisions
- ➔ Impulsive Forces
- ➔ Persistent Forces
- ➔ Angular Forces
- ➔ ...

# Simulated Forces

- ➔ Gravity
- ➔ Friction
- ➔ Normal Force
- ➔ Collisions
- ➔ Impulsive Forces
- ➔ Persistent Forces
- ➔ Angular Forces
- ➔ ...

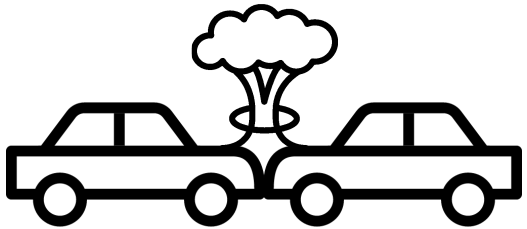
# Simulated Forces

- ➔ Gravity
- ➔ Friction
- ➔ Normal Force
- ➔ Collisions
- ➔ Impulsive Forces
- ➔ Persistent Forces
- ➔ Angular Forces
- ➔ ...



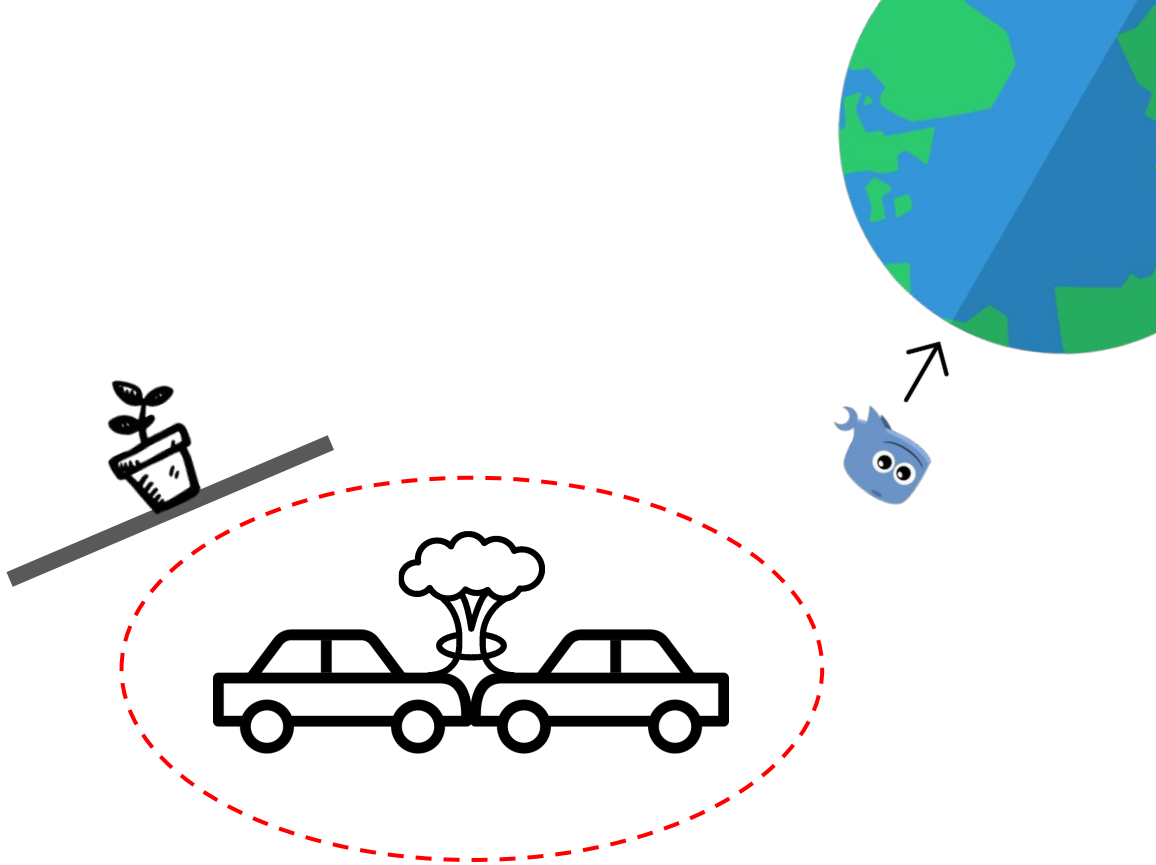
# Simulated Forces

- ➔ Gravity
- ➔ Friction
- ➔ Normal Force
- ➔ Collisions
- ➔ Impulsive Forces
- ➔ Persistent Forces
- ➔ Angular Forces
- ➔ ...



# Simulated Forces

- ➔ Gravity
- ➔ Friction
- ➔ Normal Force
- ➔ **Collisions**
- ➔ Impulsive Forces
- ➔ Persistent Forces
- ➔ Angular Forces
- ➔ ...



# Collisions

Non-elastic
Objects continue with same velocity

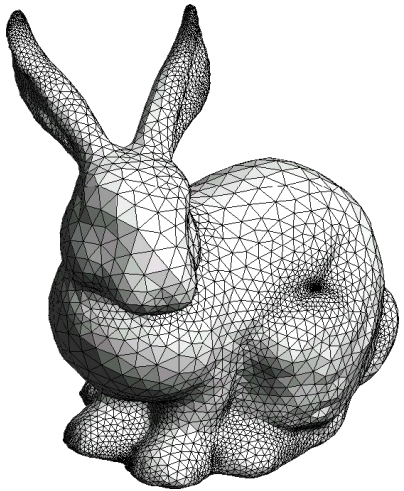
**vs.**

Elastic
Objects bounce in different directions

# Soft Bodies (internal forces)

# Deformable Objects

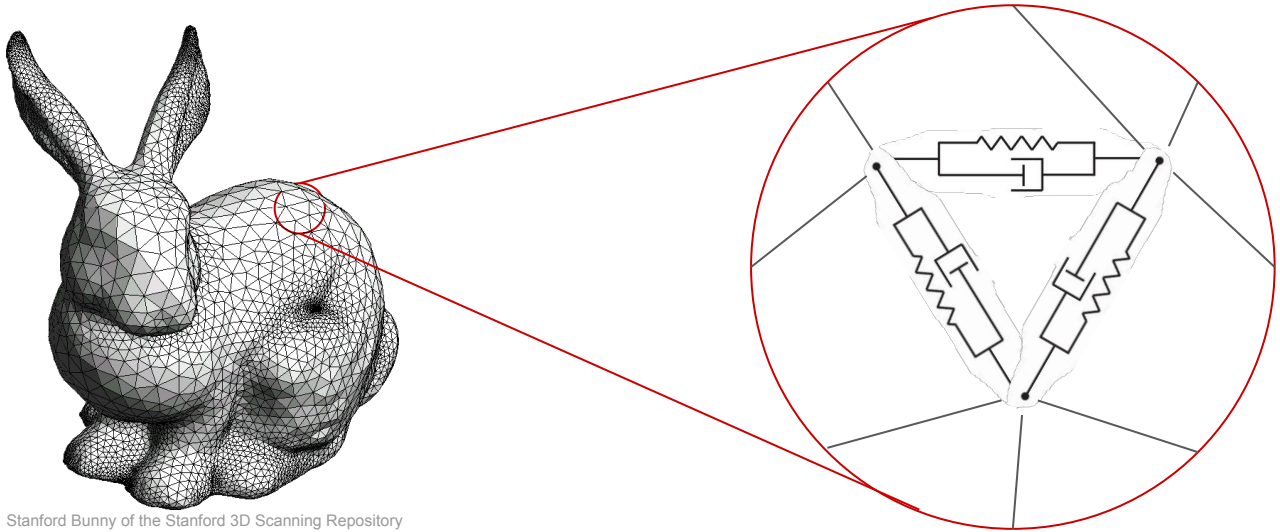
➔ Vertices have mass and can change position



Stanford Bunny of the Stanford 3D Scanning Repository

# Deformable Objects

- ➔ Vertices have mass and can change position
- ➔ Internal forces are simulated with spring - damper - systems

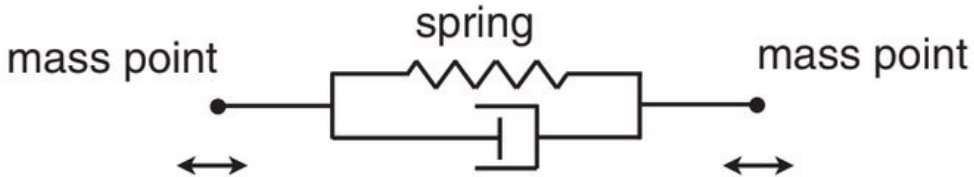


Stanford Bunny of the Stanford 3D Scanning Repository

# Spring-Damper-System

➔ (virtual) springs apply force to return to their *rest length*

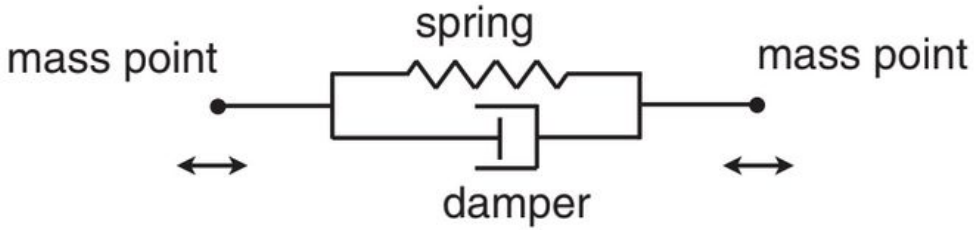
$$f = f_r = k*(l_{rest} - l)$$



# Spring-Damper-System

- ➔ (virtual) springs apply force to return to their *rest length*
- ➔ Damper reduces jerky behavior

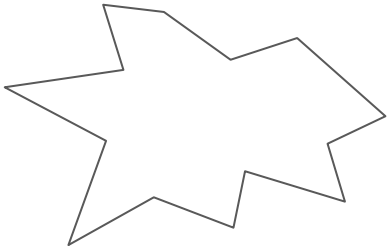
$$f = f_r + f_d = k*(l_{rest} - l) + c*v$$

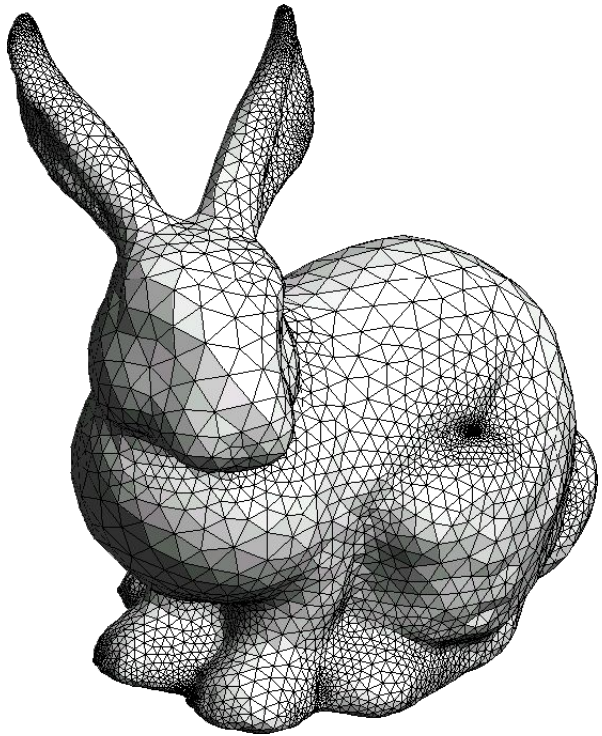



Destructible Objects

# Boolean Destruction

➔ Subtract one mesh from another

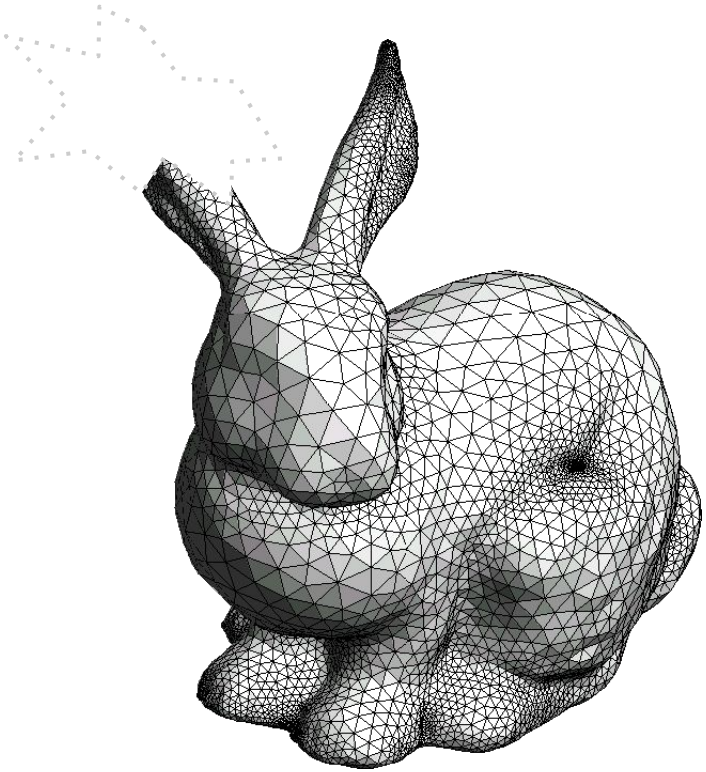




Stanford Bunny of the Stanford 3D Scanning Repository

# Boolean Destruction

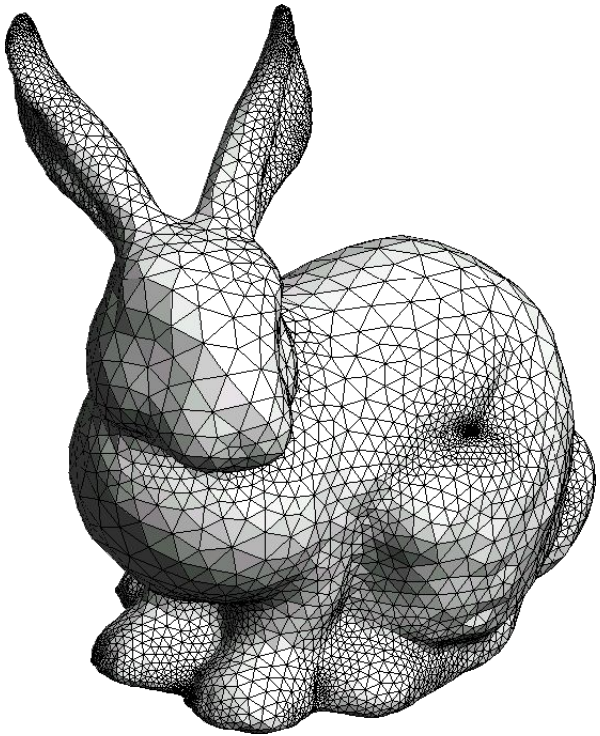
- ➔ Subtract one mesh from another
- ➔ E.g. holes of impacts



Stanford Bunny of the Stanford 3D Scanning Repository

# Fracturing

➔ Fragments are pre-computed by the physics engine

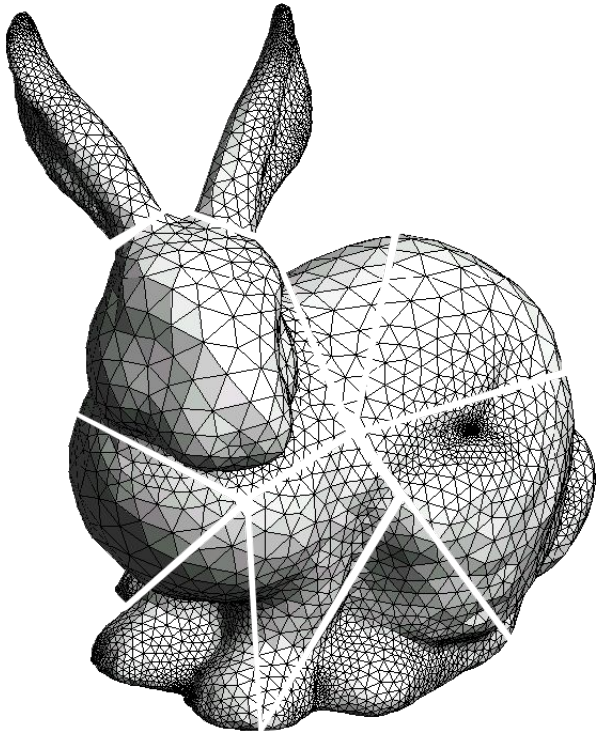


Stanford Bunny of the Stanford 3D Scanning Repository

# Fracturing

- ➔ Fragments are pre-computed by the physics engine
- ➔ When certain requirements are met, whole rigid objects is substituted with fragments
- ➔ Fragments then behave like independent objects

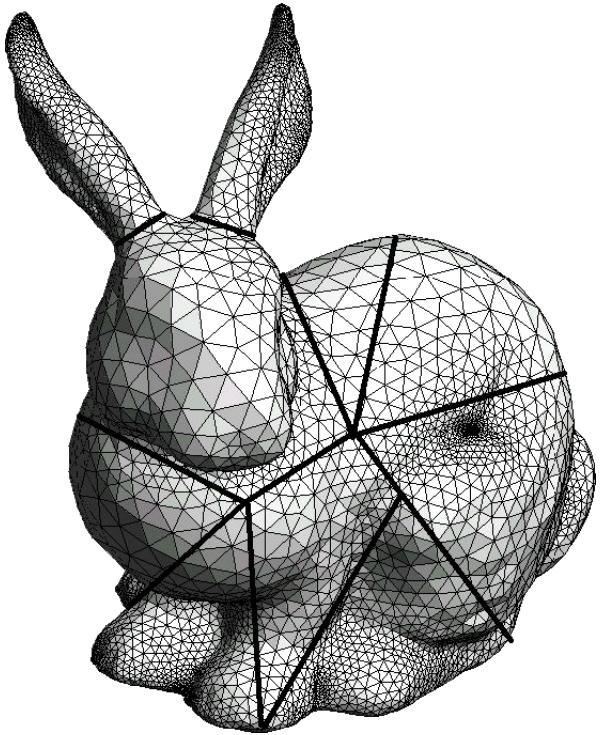
- ➔ Various methods for generating fragments



Stanford Bunny of the Stanford 3D Scanning Repository

# Slicing

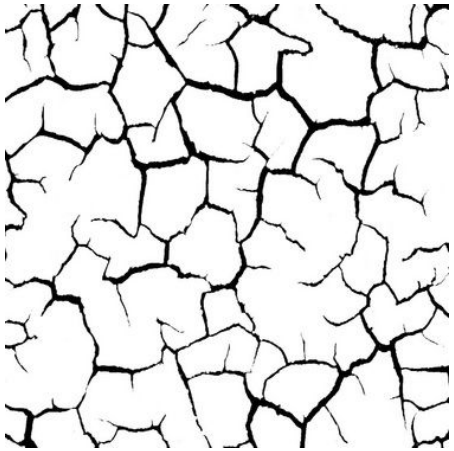
- ➔ Recursively split object
- ➔ Borders may be  manipulated by various parameters

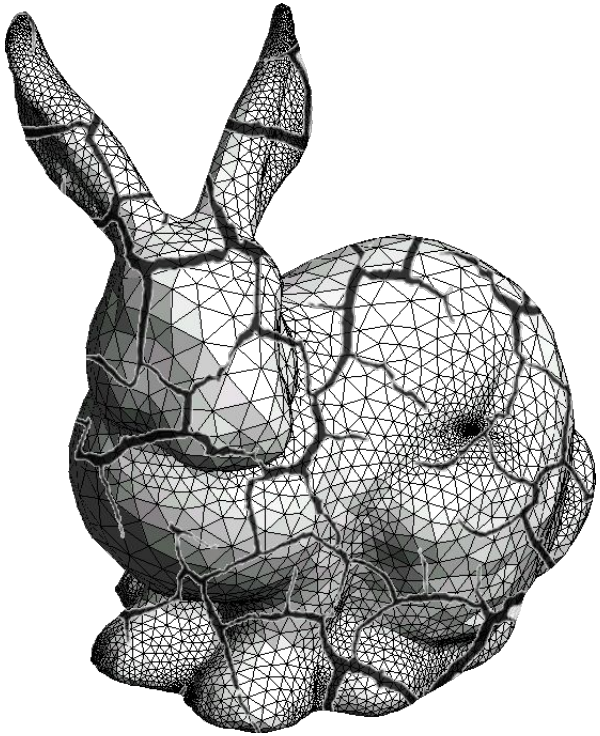


Stanford Bunny of the Stanford 3D Scanning Repository

# Fracturing Map

- ➔ User provides fracture map which is then projected onto geometry
- ➔ User can fully control the fracture design

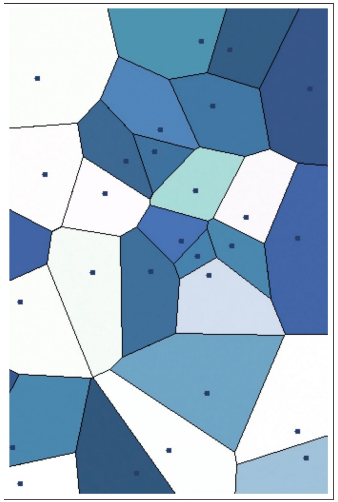


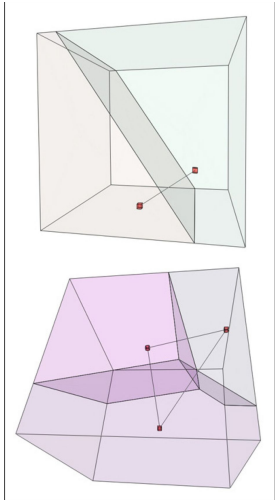


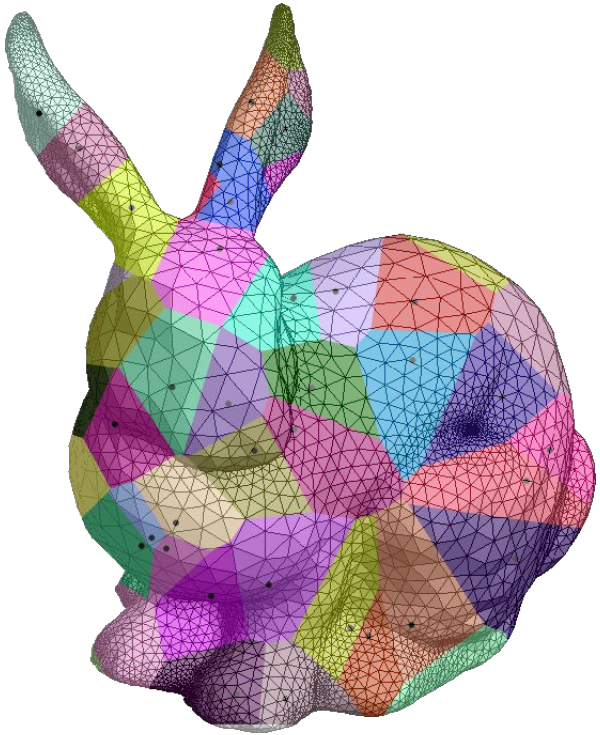
Stanford Bunny of the Stanford 3D Scanning Repository

# 3D Voronoi Diagram

- ➔ Fragments are created by calculating a 3D Voronoi structure.
- ➔ Efficient while providing believable results







Stanford Bunny of the Stanford 3D Scanning Repository

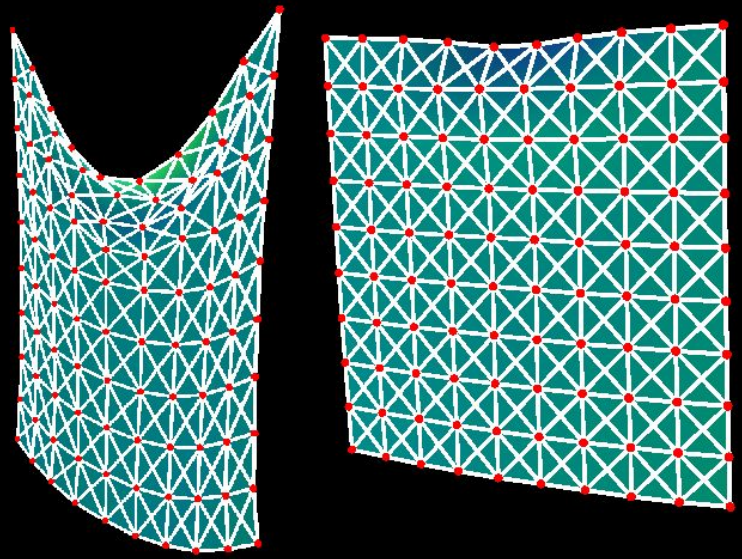

Cloth Simulation

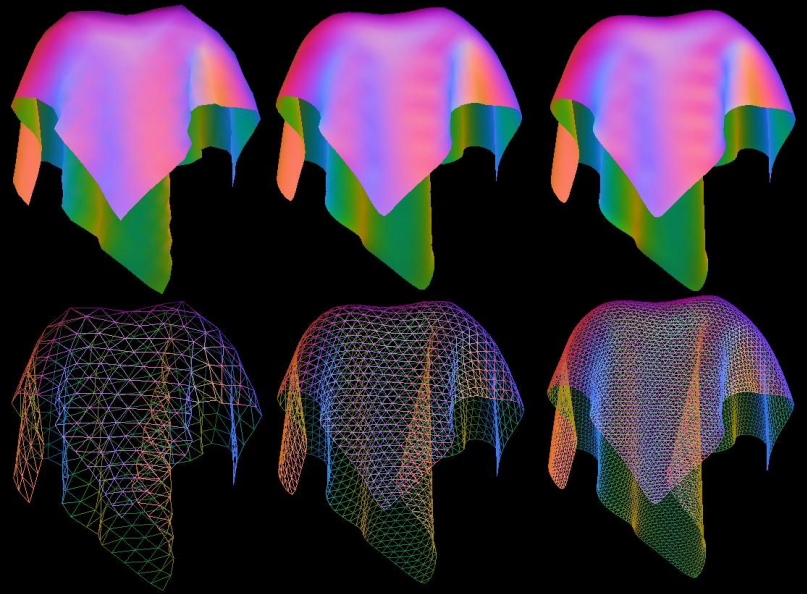
Real-time Cloth Rendering with Fiber-level Detail

# Looks vs. Behaviour

Real-time Cloth Rendering with Fiber-level Detail

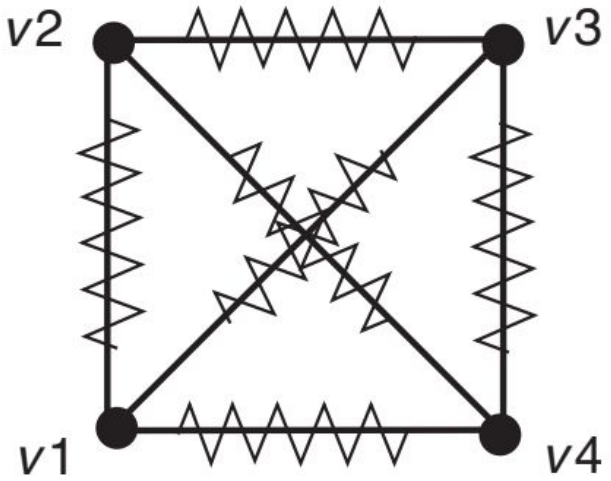
# Simulation with Spring-Mass-System



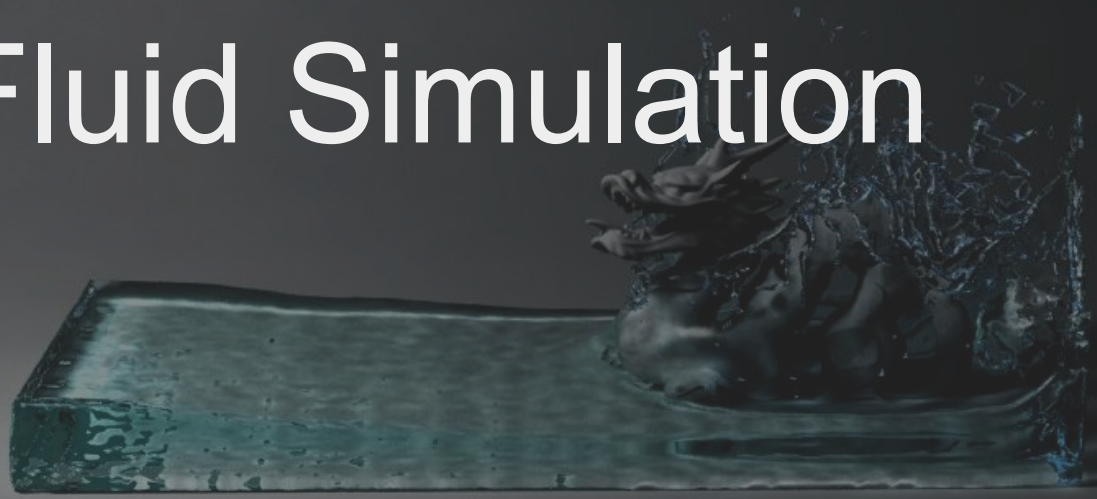


# Simulation with Spring-Mass-System

- ➔ Piece of cloth is 2D deformable mesh
- ➔ Collision handling must include self-collision
- ➔ ‘Accurate’ cloth-object-collision: Test triangles of mesh for collision
- ➔ Cloth-cloth-collision: Test & restrict distances of vertices

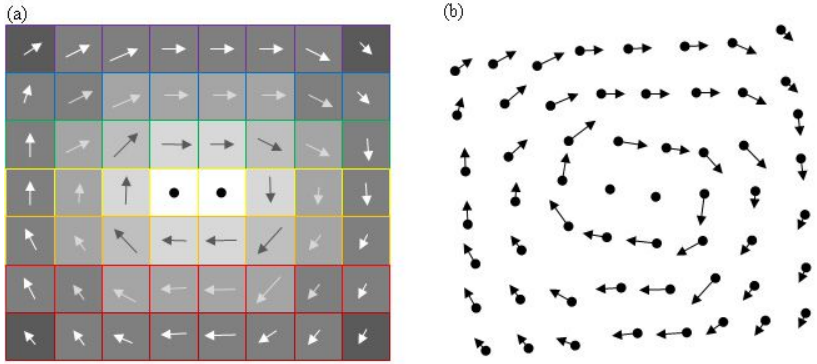


# Fluid Simulation



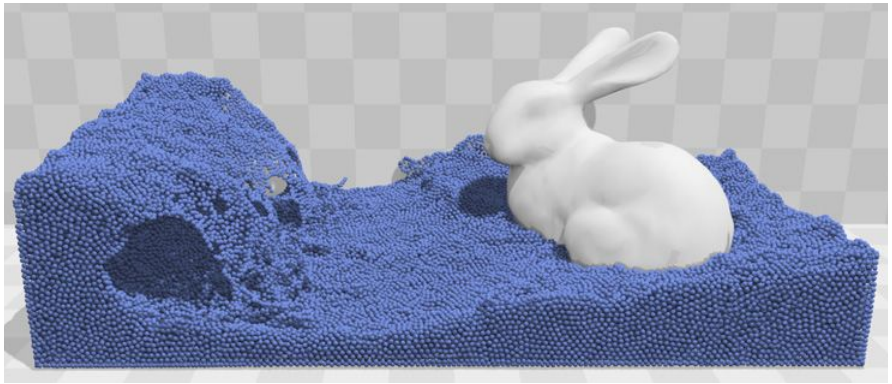
# Approaches to Fluid Simulation

Field-Based




https://software.intel.com/en-us/articles/fluid-simulation-for-video-games-part-1

Particle-Based

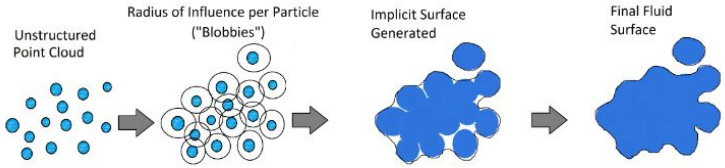


http://www.cs.cornell.edu/courses/cs5643/2015sp/a1PositionBasedFluids/index.html

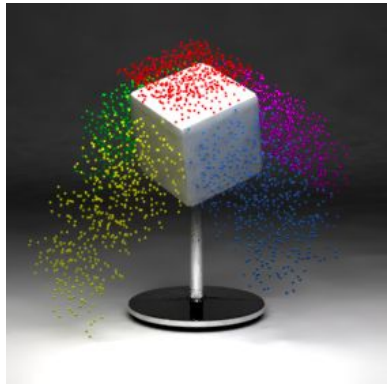
# Remember Particle Systems?

- ➔ Water consists of numerous (almost) free moving particles (atoms)
- ➔ Simulation consists of big particle systems with parameters leading to water-like behaviour

https://www.youtube.com/watch?v=DhNt_A3k4B4




http://users.encs.concordia.ca/~grogono/Graphics/fluid-5.pdf



By Halixi72 at the English language Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=58485548

# Sources:

ani_ch7 (Physically based Animation)
rtr_ch17 (collision_detection)
3dm_ch12 (mechanics_2)
“Approaches to destruction effects in real-time computer graphics” - )R. Hettich)
Real-time Cloth Rendering with Fiber-level Detail - (Kui Wu and Cem Yuksel)

khanacademy.org/partner-content/pixar
opengl-tutorial.org/assets/images/tuto-particules/
what-when-how.com/advanced-methods-in-computer-graphics/

Iconarchive.com
onlinewebfonts.com
giphy.com