Physical Simulation on GPUs

Jim Van Verth

OpenGL Software Engineer
NVIDIA

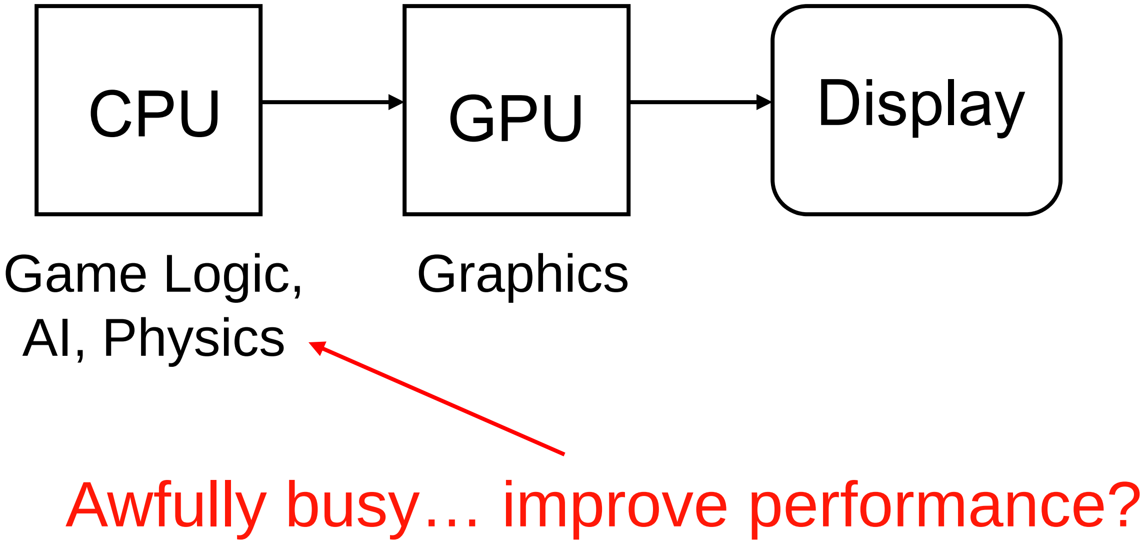
jvanverth@nvidia.com
www.nvidia.com
www.essentialmath.com

Physics on GPU

- Topics of discussion
  - Ways of parallelizing physics
  - Examples of GPU physics
  - CUDA and you

Parallelizing Physics

- What we’ve talked about so far



CPU

GPU

Display

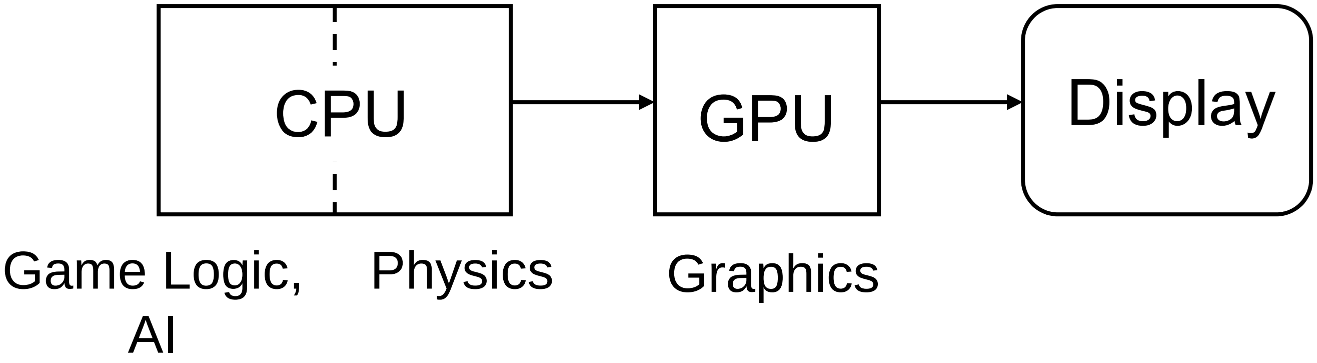
Game Logic, AI, Physics

Graphics

Awfully busy… improve performance?

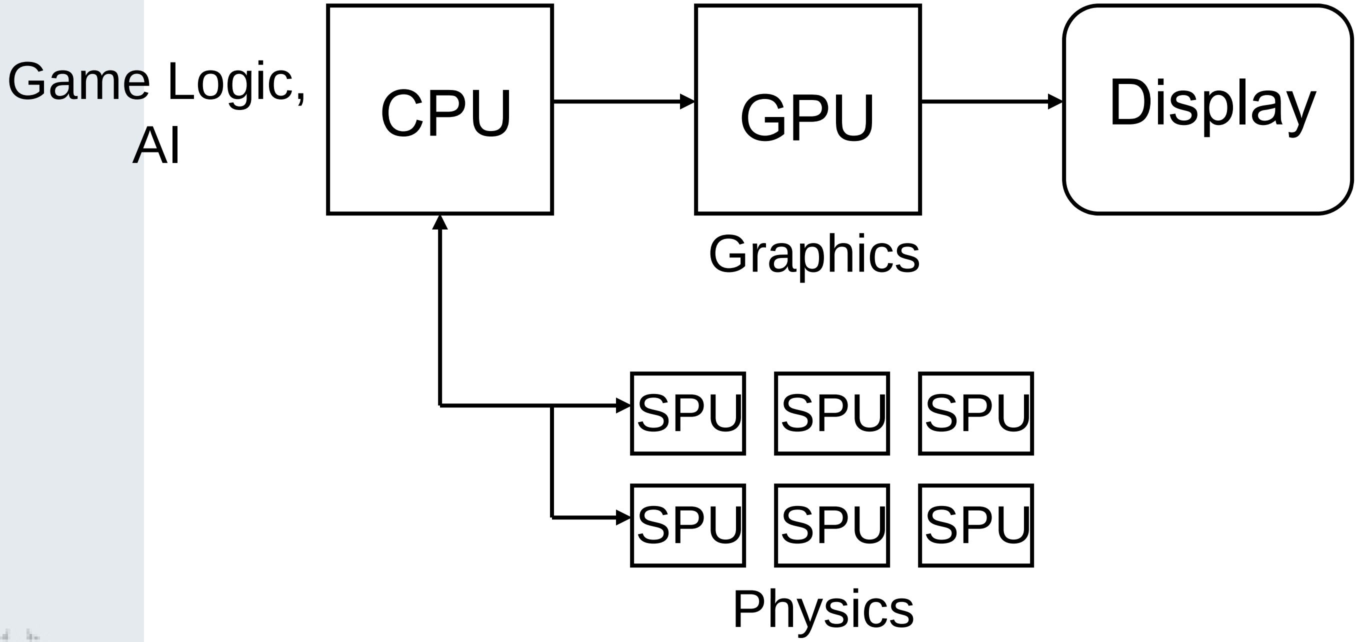
Parallelizing Physics

- Solution 1: Multicore CPU



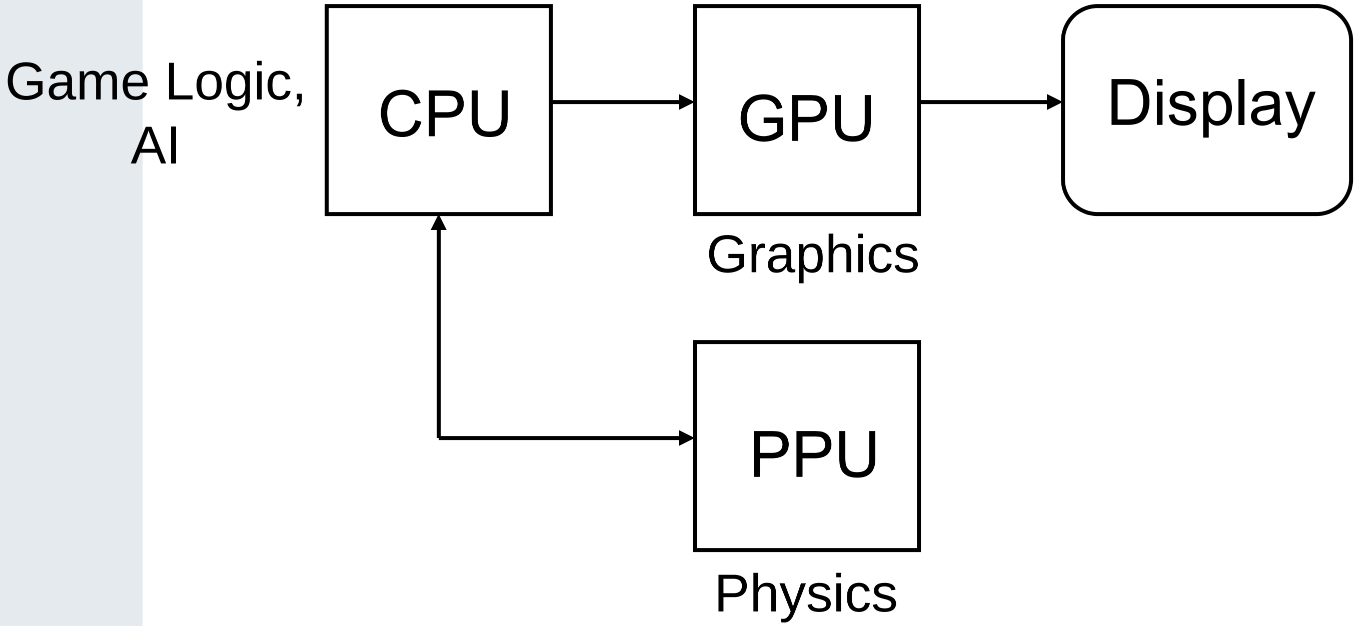
Parallelizing Physics

- Solution 2a: Cell processor



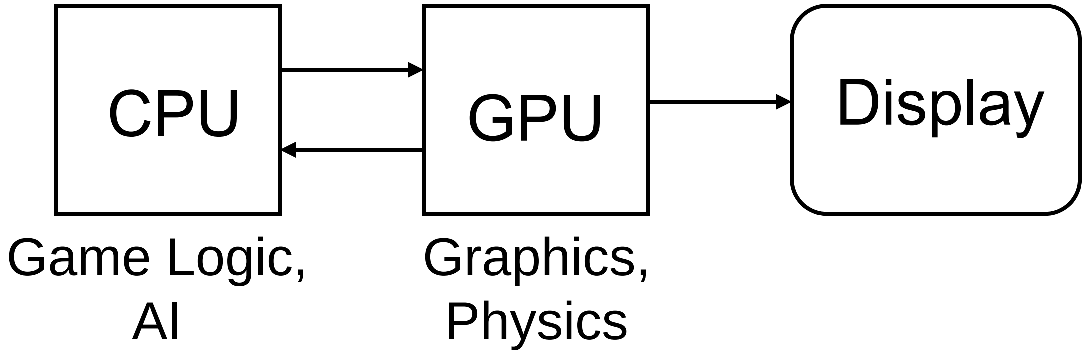
Parallelizing Physics

- Solution 2b: AGEIA processor



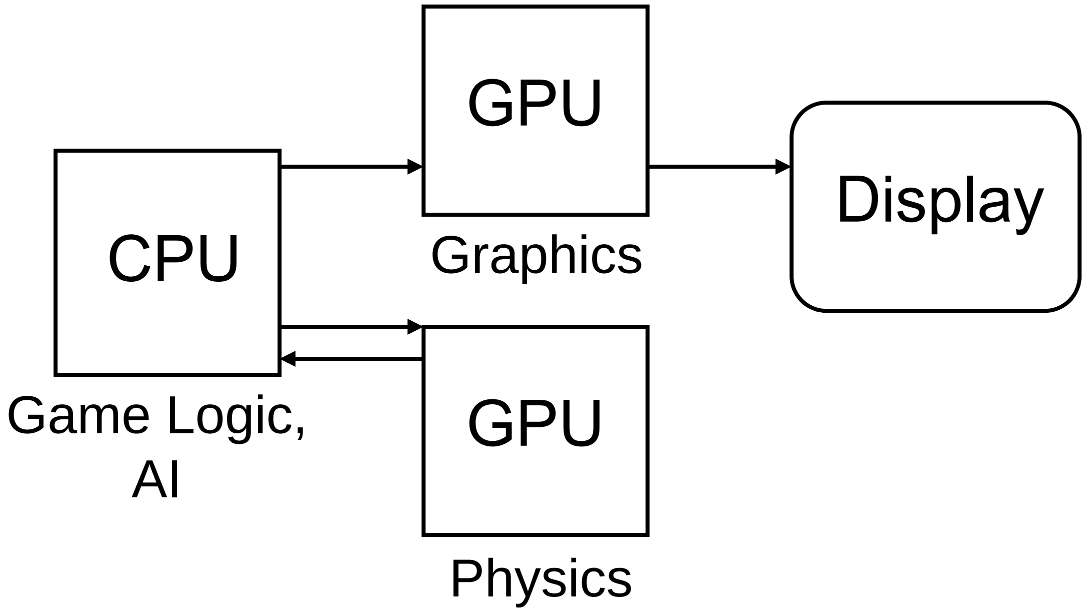
Parallelizing Physics

- Solution 3: Programmable GPU



Parallelizing Physics

- Solution 3b: SLI



GPU Computing

- Modern GPU has many independent processors:
  - GeForce 8800 GTX: 128 SPs
  - GeForce 8800 GT: 112 SPs
- Mostly processing power, not cache:
  - GeForce 8800 GTX: 300-400 Gflops
  - GeForce 8800 GT: 500 Gflops
- A lot of parallel power for physics!

GPU Physics Example

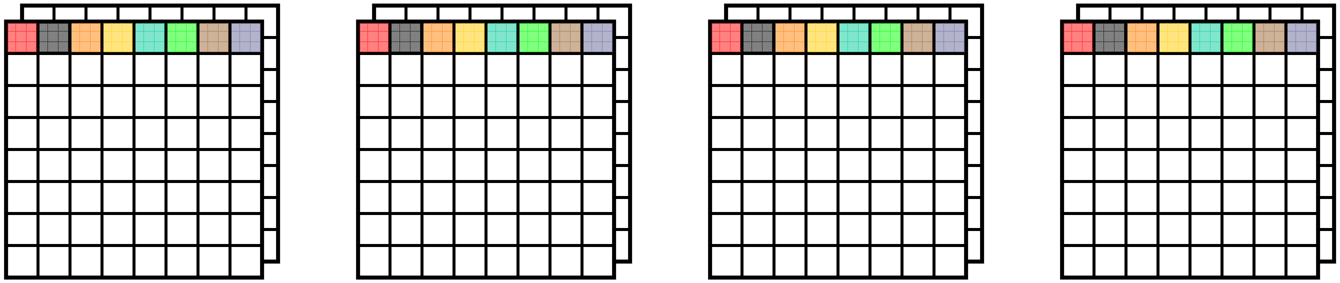
- From GPU Gems 3
- Takahiro Harada, “Real-time Rigid Body Simulation on GPUs”
- Simple physics engine, all running on GPU

GPU Physics Example

- Idea: GPU is good at:
  - Many similar computations
  - Simple data
- So:
  - Particles for collision representation
  - Grid for collision detection
  - Simple collision response

Object Representation

- Global object data in texture pairs



Position

Orientation

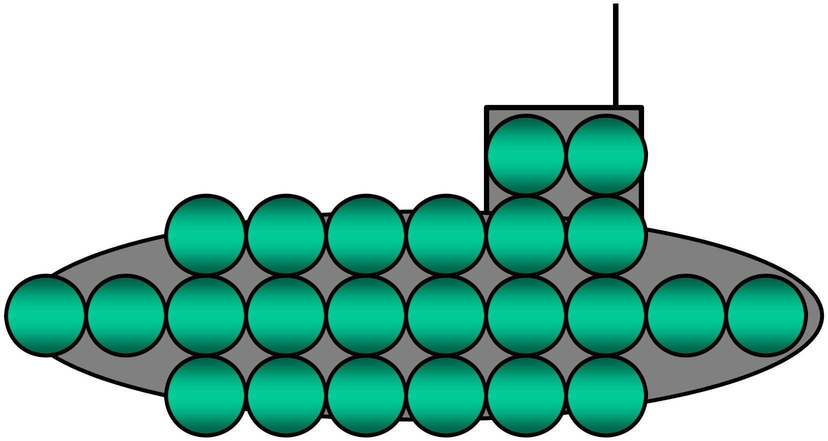
Linear Momentum

Angular Momentum

- Alternate frame to frame

Object Representation

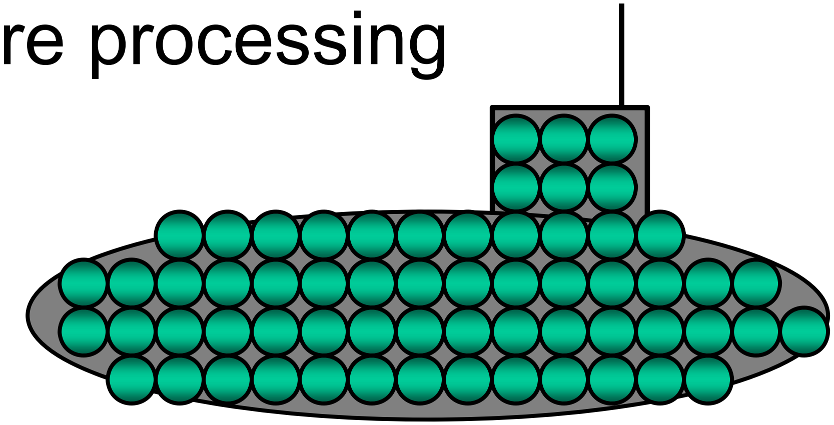
- Collision rep: Solid (or shell) of particles



- Store as
  - Fixed radius
  - Displacement from center of mass

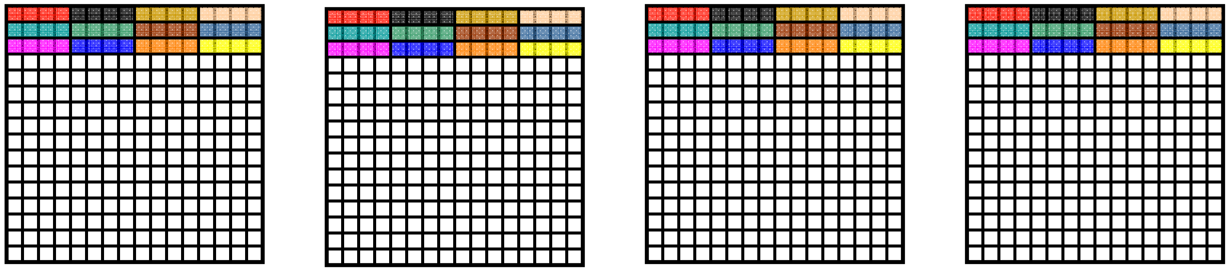
Object Representation

- Smaller particles == better fit
- But more processing



Object Representation

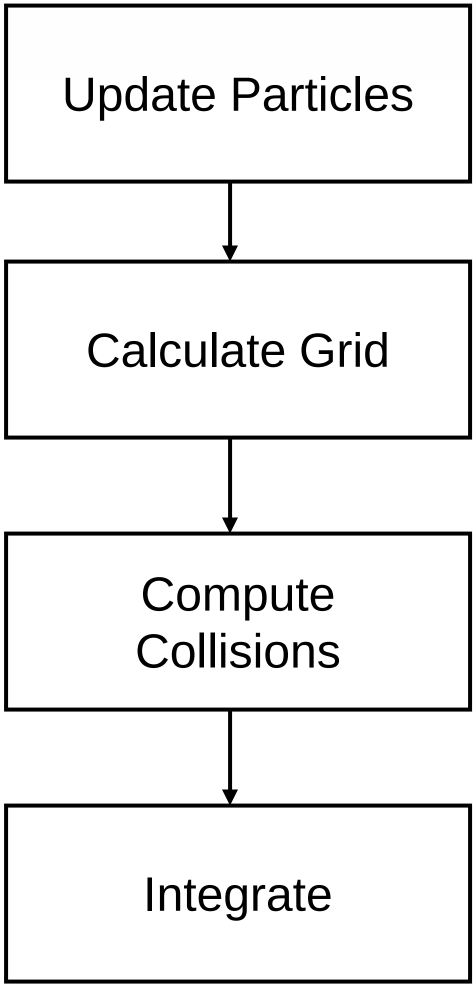
- Particle data stored in texture and three rendertargets



Displacement Position Velocity Force

- Update position, velocity each frame from global object data
- Update force from collisions

Pipeline

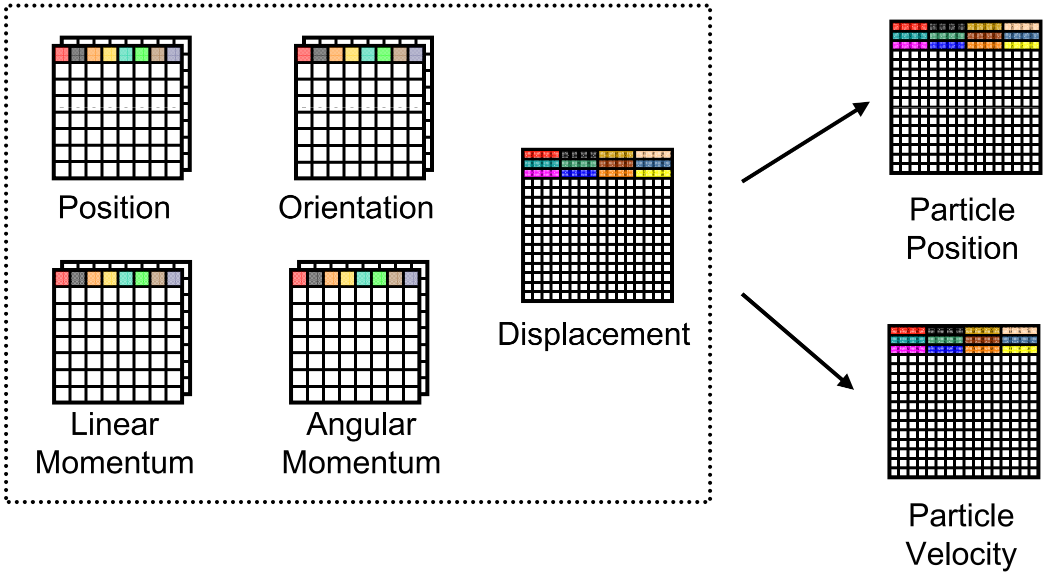

Update Particles
Calculate Grid
Compute Collisions
Integrate

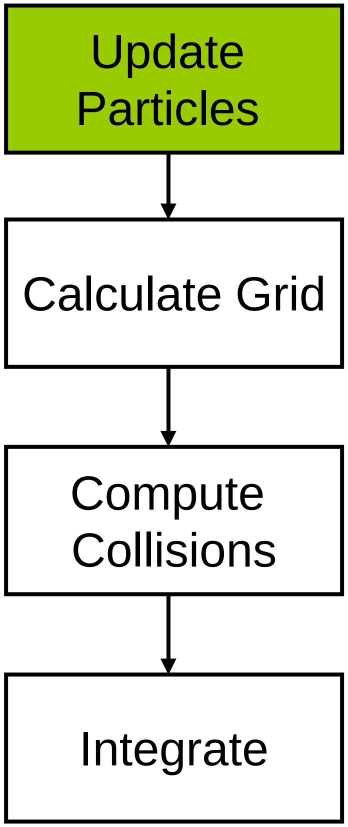
Update Particles

- For each object do:

  Iterate through all particles

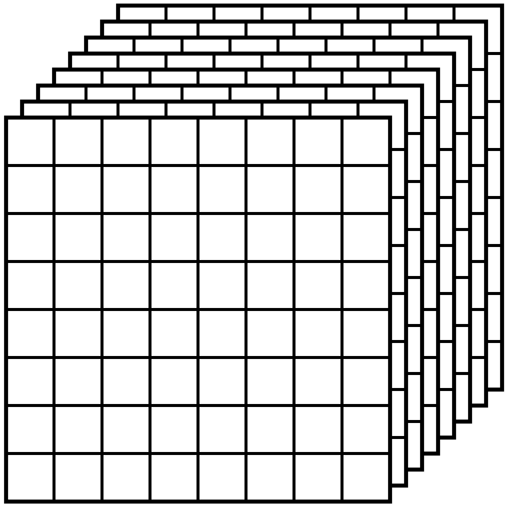
  Update particle position, velocity



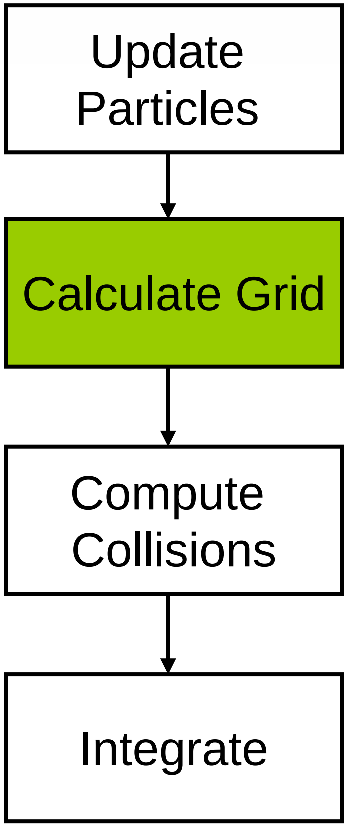


Grid Representation

- Stored as slabs within 2D rendertarget



- Voxel stored as texel
- Four particle indices per texel



Update Particles

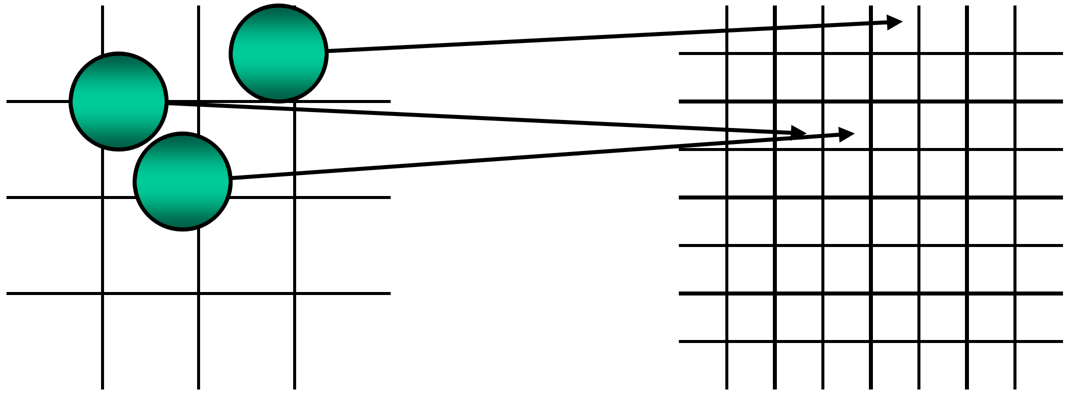
Calculate Grid

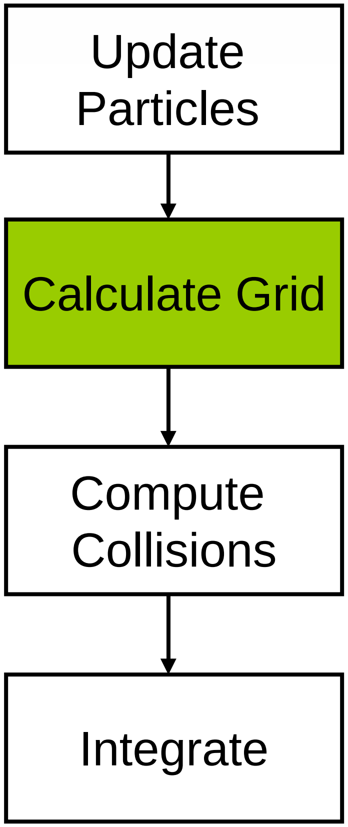
Compute Collisions

Integrate

Grid Creation

- For each particle do
  - Compute grid index
  - Write particle index to appropriate component at that location



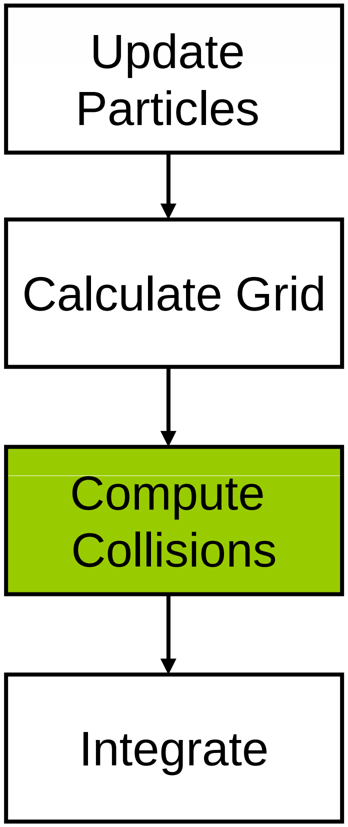


Collision Resolution

- For each voxel do

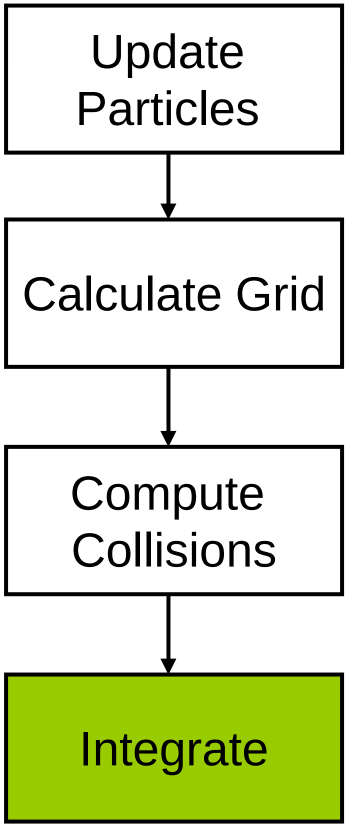
  For each particle in voxel do

  - Compute force based on particles in this and 27 neighboring voxels
  - Regardless of collision!
    - Spring force
    - Damping from relative vel.
    - Tangential force



Integration

- Compute new linear and angular momenta based on collision (and other) forces
  - Force/torque on rigid body is weighted sum of forces from each particle
- Compute new position and orientation from momenta



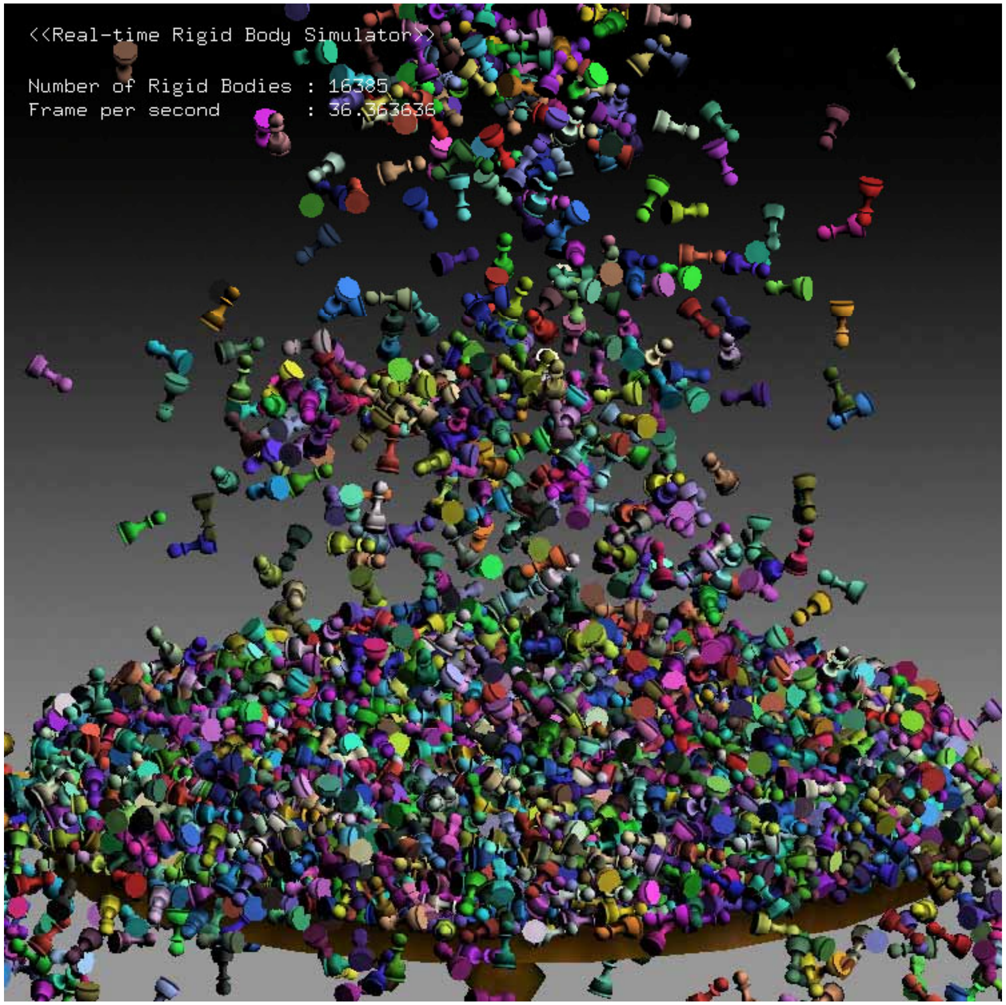
Update Particles

Calculate Grid

Compute Collisions

Integrate

Demo



```
<<Real-time Rigid Body Simulator>>

Number of Rigid Bodies : 16385
Frame per second       : 36.363636
```

Other approaches

- Simon Green’s particles
- Nyland, Harris and Prins: N-body sim.
- Parallelize one piece:
  - Ex. Broad Phase (from GPU Gems 3)
- Do smaller problem
  - Ex. Fluid dynamics (Hellfire: London)

GPU Computing

- How to program?
- In the past had to use Cg, GLSL, HLSL
- Problems:
  - Requires specialized shader knowledge
  - Data is often texture or rendertarget
  - Can't "scatter" data easily

CUDA

- Solution is CUDA
- Stands for Compute Unified Device Architecture
- Extensions on C/C++
- Interoperable with D3D and OpenGL
- www.nvidia.com/cuda
- Use it!

CUDA

- Updating our example:
  - Instead of Cg, use standard C++ w/CUDA extensions
  - Instead of textures or rendertargets, just use CUDA arrays
  - Instead of vertex shader, use scatter operation

References

Takahiro Harada. “Real-Time Rigid Body Simulation on GPUs.” In *GPU Gems 3*, Hubert Nguyen, ed., Addison-Wesley, 2007.

Lars Nyland, Mark Harris, Jan Prins. “Fast N-Body Simulation with CUDA.” In *GPU Gems 3*, Hubert Nguyen, ed., Addison-Wesley, 2007.

Scott Le Grand. “Broad-Phase Collision Detection with CUDA.” In *GPU Gems 3*, Hubert Nguyen, ed., Addison-Wesley, 2007.

Keenan Crane, Ignacio Llamas, Sarah Tariq. “Real-Time Simulation and Rendering of 3D Fluids.” In *GPU Gems 3*, Hubert Nguyen, ed., Addison-Wesley, 2007.

Simon Green. “CUDA Particles” NVIDIA whitepaper, November, 2007.

NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 1.1, November 2007.