USING GPUS FOR COLLISION DETECTION

AMD
Takahiro Harada

AGENDA

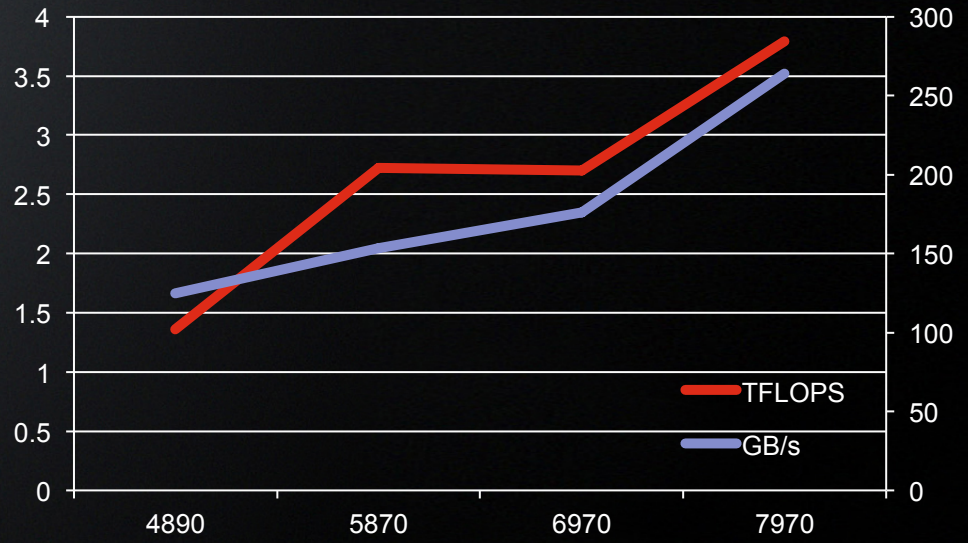
- GPU architecture updates
- Problem description
  - Why collision detection is different from other physics problems
- GPU collision detection
- Heterogeneous collision detection

GPU UPDATES

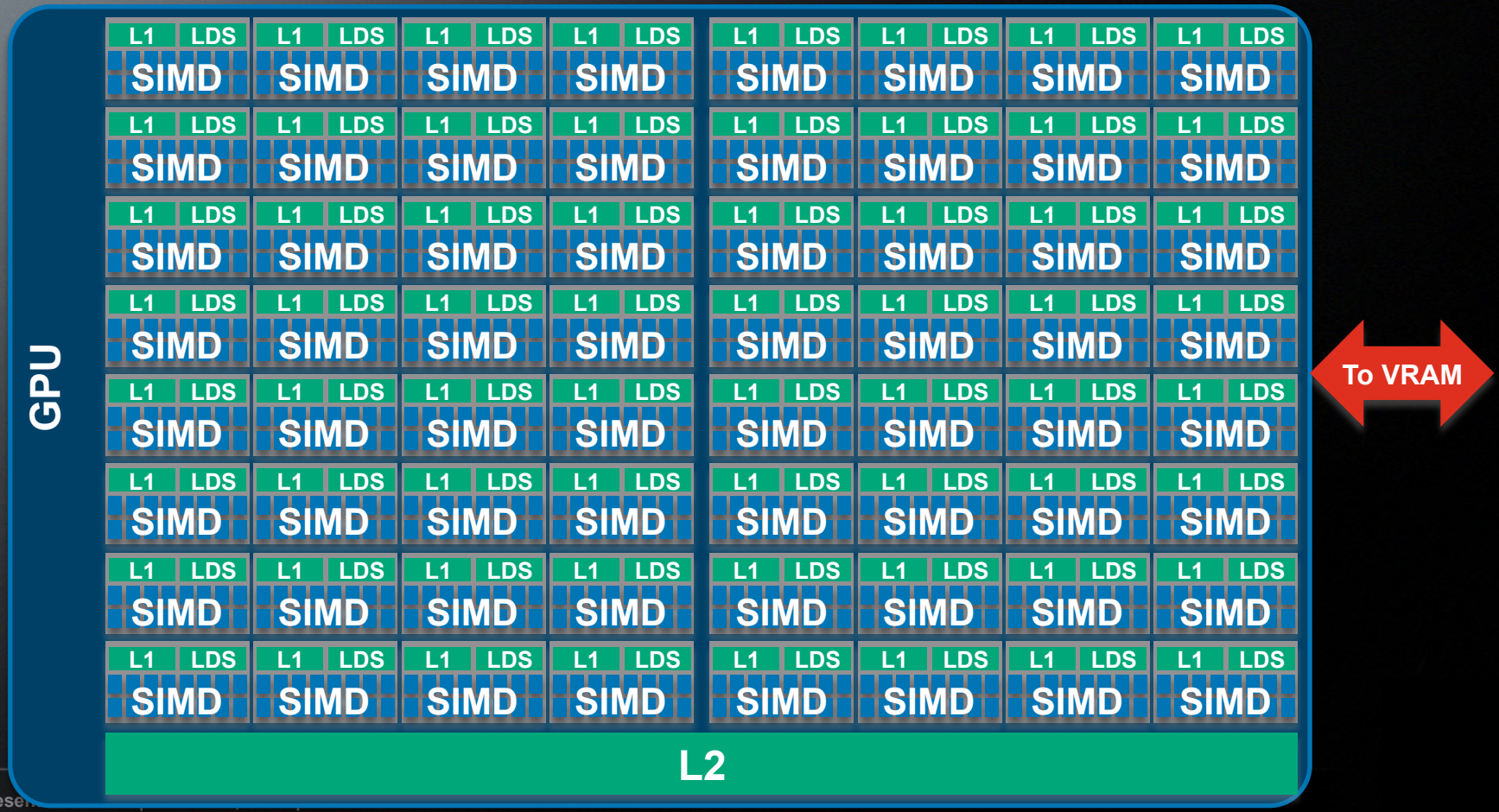
GPU PEAK PERFORMANCE

- High ALU performance (FLOPS)
- High memory bandwidth

- Radeon HD 7970
  - 3.79 TFLOPS
  - 264 GB/s
  - 32x4 SIMD engines
  - 64 wide SIMD



GPU ARCHITECTURE (AMD RADEON HD 7970)



PEAK PERFORMANCE IS HIGH, BUT

- GPU used to be only good at simple computations
- GPU is improving architecture

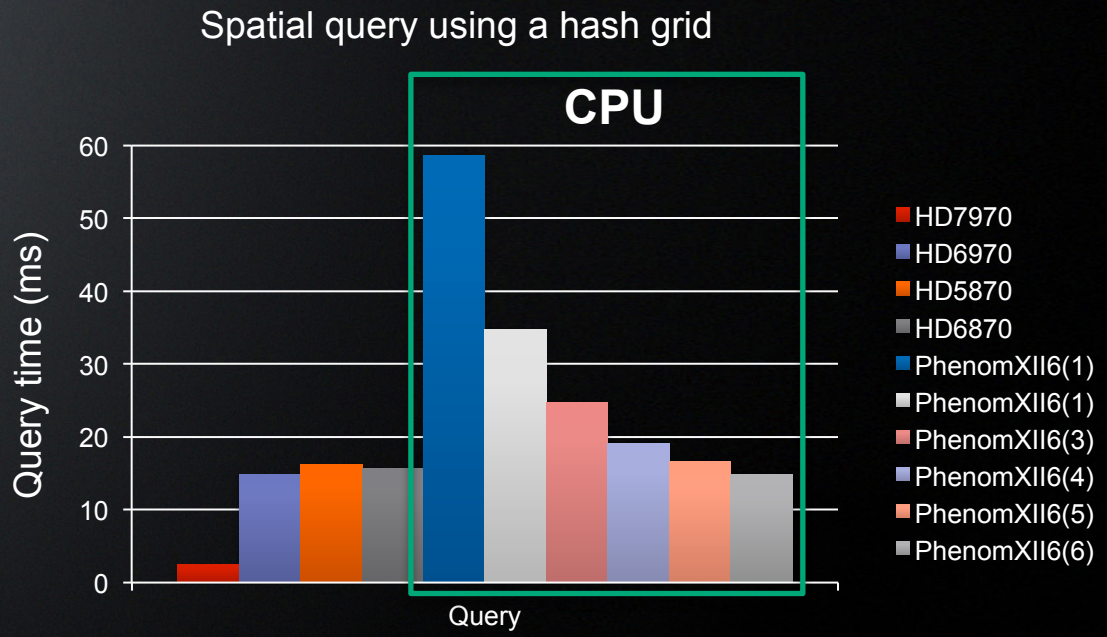
- GPU performs good only if used correctly
  - Divergence
- Prefer simpler algorithm and data structure
  - ⇔ Linked list traversal

EX) LINKED LIST TRAVERSAL

- Low ALU / Mem ratio
- Pointer chasing
- Traversal

```
Node* node = root;
do
{
  process( node );
  node = node->next;
}while( node )
```

- CPU is better for this

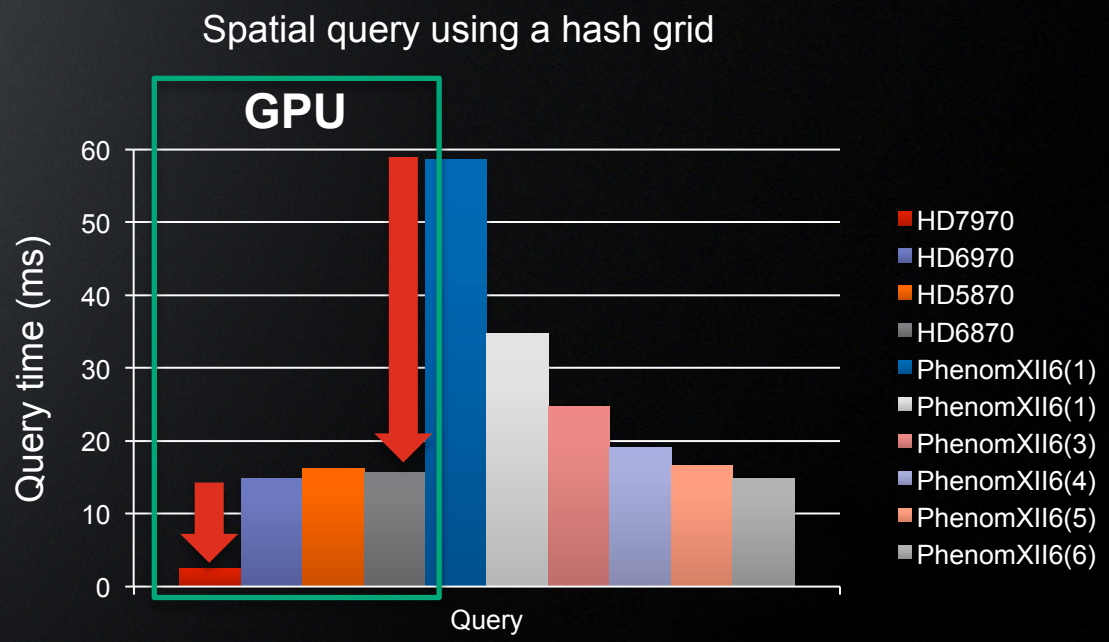


EX) LINKED LIST TRAVERSAL

- Low ALU / Mem ratio
- Pointer chasing
- Traversal

```
Node* node = root;
do
{
  process( node );
  node = node->next;
}while( node )
```

- CPU is better for this



REASON OF PERFORMANCE JUMP

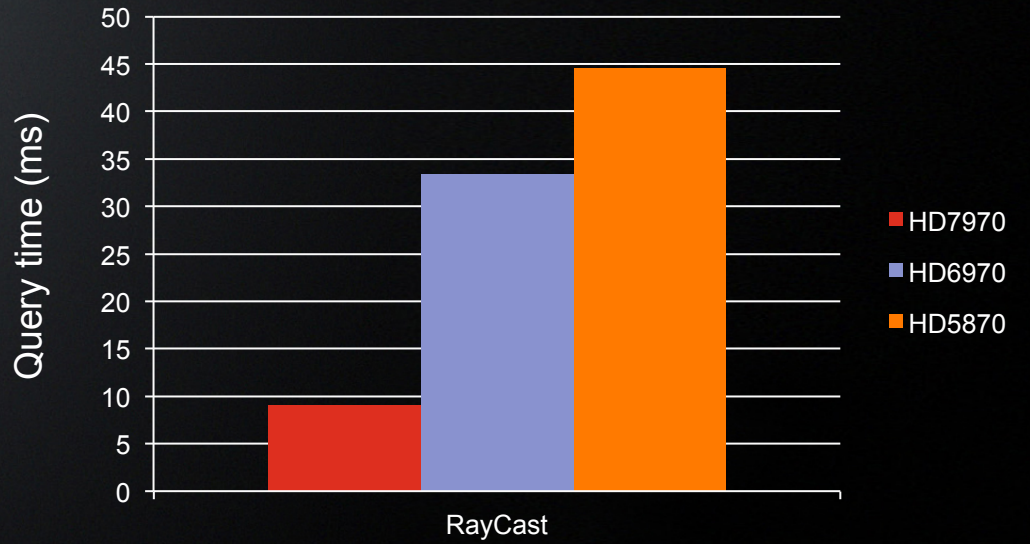
- Cannot explain only by raw performance increase
  - It is less than 2x.

- Key architectural changes of HD7970
  - Non VLIW
    - Increases ALU efficiency
  - Improved memory system
    - RW cache
  - Faster atomic operations
    - On chip atomics

- Improves computation performance in general

LARGE LEAP IN PERFORMANCE

- Ray casting
  - Ray tracing
  - Visibility query
- GPU outperforms a multicore CPU
  - Test uses 6 cores



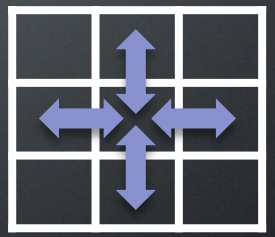
GPU EVOLUTION

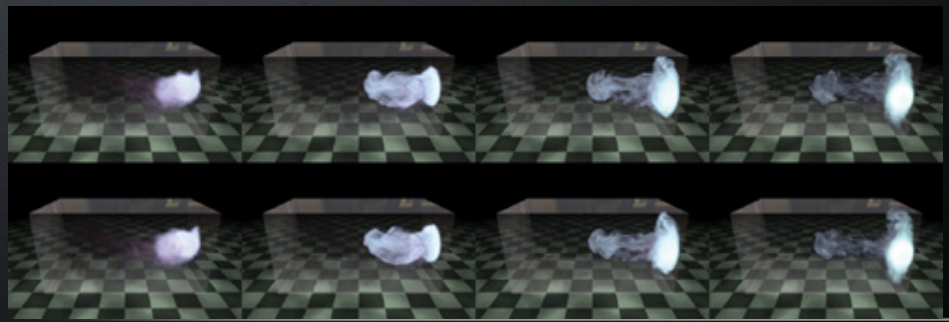
- GPU is not only adding more ALUs on a chip
- Architecture has been improved

GPU COLLISION DETECTION

COLLISION DETECTION WAS HARD FOR GPUS

- GPUs were used for physics simulation with fixed connectivity between entities
  - Grid based fluid simulation
    - Entities: Cells
    - Connections: Cell adjacency

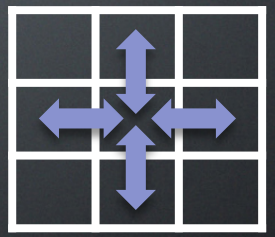


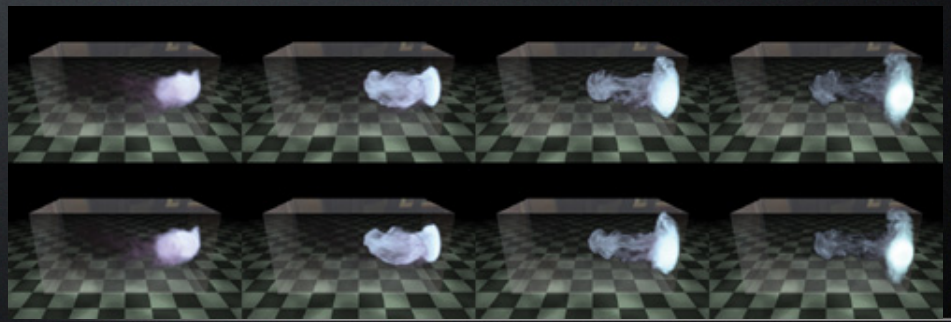


Crane et al., GPU Gems3 (2007)

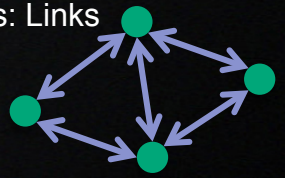
COLLISION DETECTION WAS HARD FOR GPUS

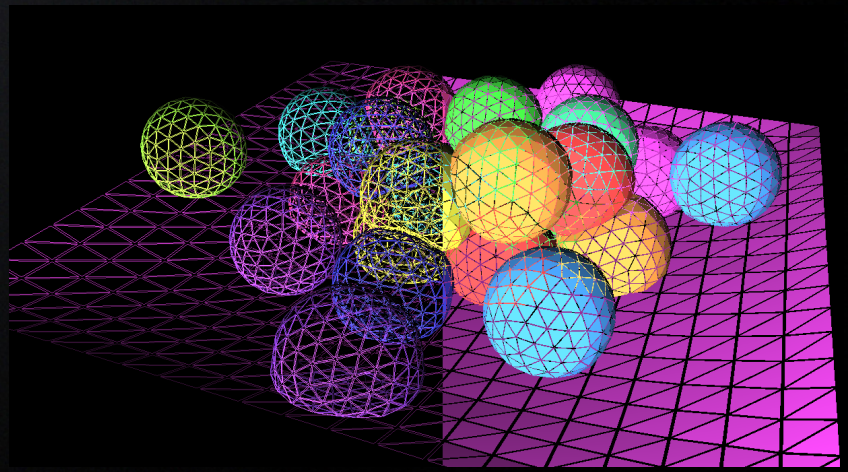
- GPUs were used for physics simulation with fixed connectivity between entities
  - Grid based fluid simulation
    - Entities: Cells
    - Connections: Cell adjacency





  - Cloth simulation
    - Entities: Vertices
    - Connections: Links

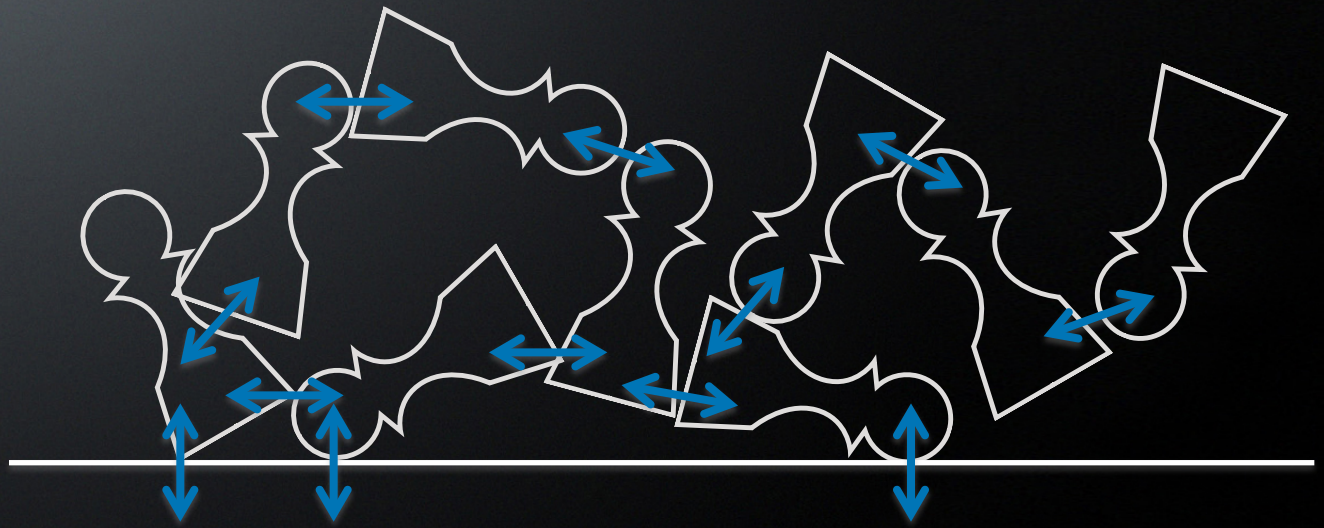




Crane et al., GPU Gems3 (2007)

COLLISION DETECTION

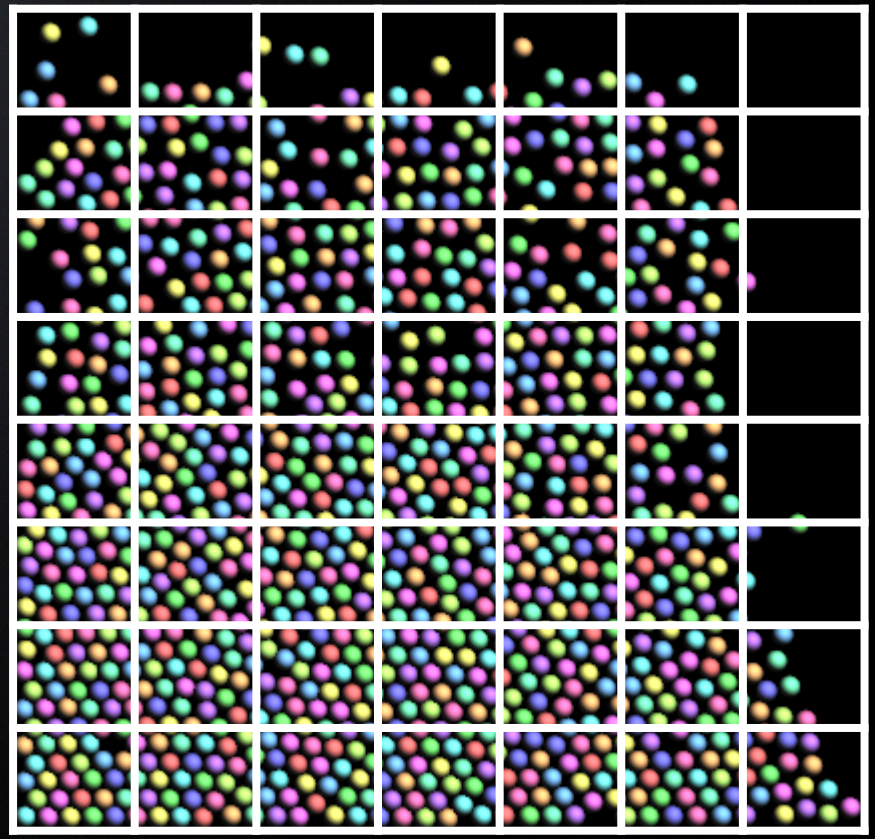
- Rigid body simulation <- Dynamic problems
  - Bodies move
  - Entities: Rigid bodies
  - Connections: Colliding pair



UNIFORM GRID

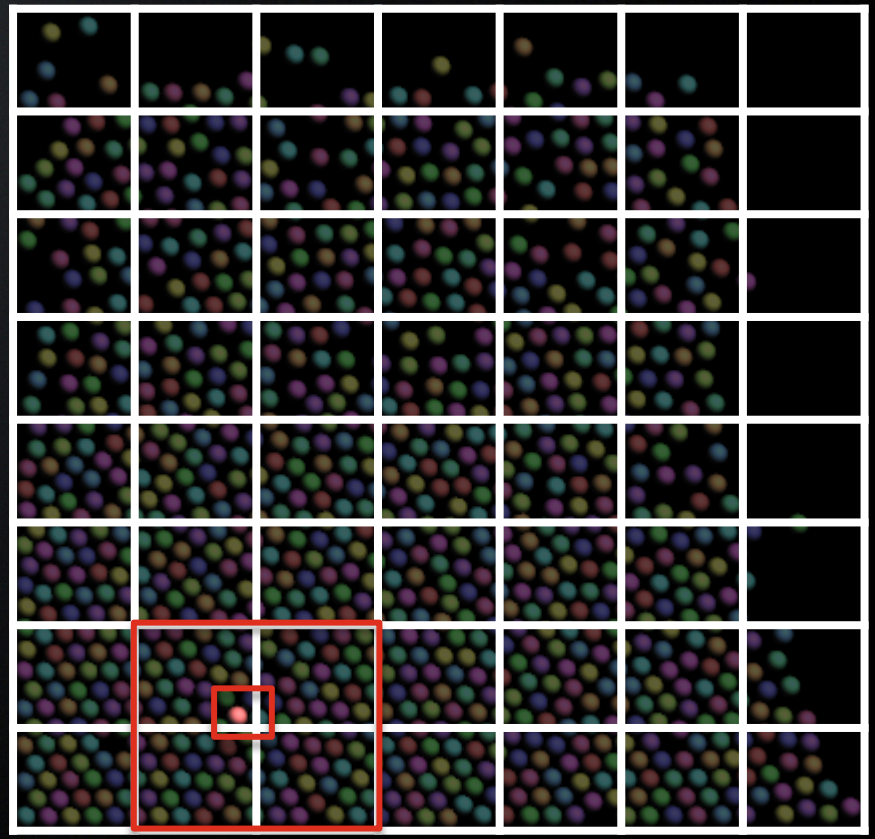
- Simple data structure
- Data is stored per cell

- Build
  - Overlay a grid
  - Find cell to which an object belongs
    - Uniform -> cell location can be calculated
  - Store reference to the cell



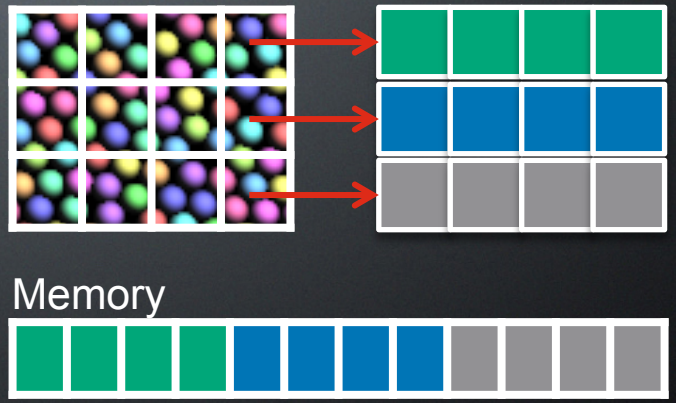
UNIFORM GRID

- Collision query
  - Find possible overlapping cells
  - Look up objects stored in the cell

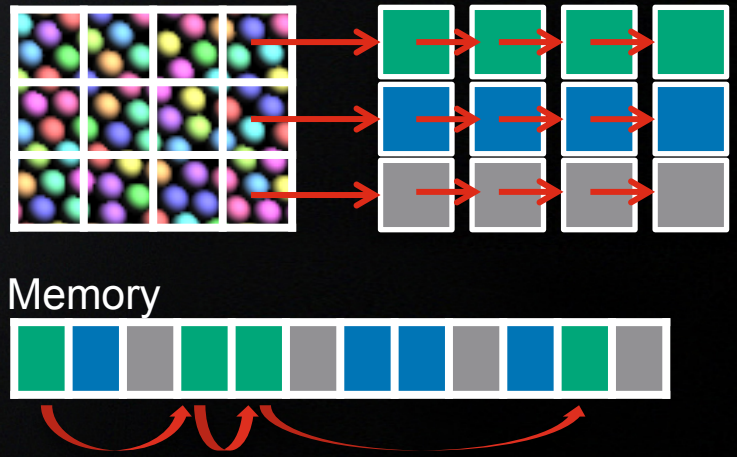


DATA STRUCTURE

- Linear array
  - An element stores data



- Linked list
  - An element stores data and link



UNIFORM GRID COLLISION DETECTION ON GPU

Algorithm

- Linear array
  - Fixed sized buffer
    - Each cell has a counter
    - Pros: simple
    - Cons: large memory, fixed entry

Enabler

- Pure graphics GPU
  - Tests in graphics pipeline

UNIFORM GRID COLLISION DETECTION ON GPU

Algorithm

- Linear array
  - Fixed sized buffer
    - Each cell has a counter
    - Pros: simple
    - Cons: large memory, fixed entry
  - Sort based
    - Sort by cell index
    - Pros: memory size
    - Cons: need a global sort

Enabler

- Pure graphics GPU
  - Tests in graphics pipeline

- On chip Local memory (LDS)
  - Fast sort (Radix sort)
  - Sort was slow (e.g., bitonic sort)

UNIFORM GRID COLLISION DETECTION ON GPU

Algorithm

- Linear array
  - Fixed sized buffer
    - Each cell has a counter
    - Pros: simple
    - Cons: large memory, fixed entry
  - Sort based
    - Sort by cell index
    - Pros: memory size
    - Cons: need a global sort
- Linked list
  - Each cell has a pointer to data
  - Pros: better for large problems
  - Cons: memory access pattern on traversal

Enabler

- Pure graphics GPU
  - Tests in graphics pipeline

- On chip Local memory (LDS)
  - Fast sort (Radix sort)
  - Sort was slow (e.g., bitonic sort)

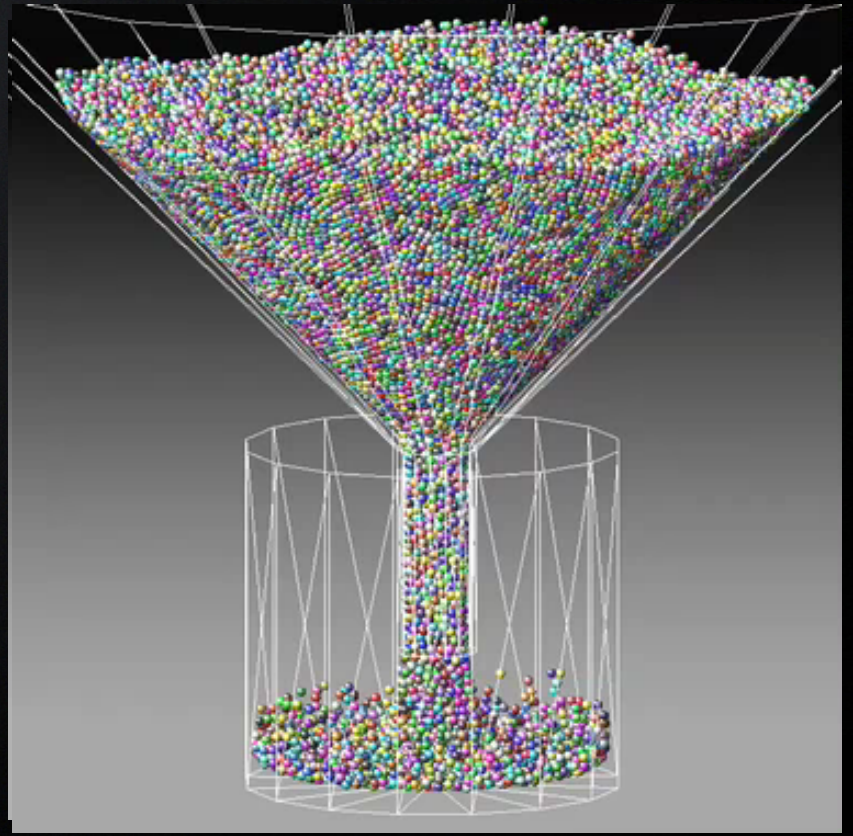
- Fast atomics
  - Intensive atomics was too expensive
  - Traversal was too slow

WHICH ALGORITHM IS THE BEST??

- Best algorithm depends on architecture
  - GPU or CPU
  - Different generation GPUs
- GPU is still changing architecture
  - Today's best algorithm isn't always the best tomorrow

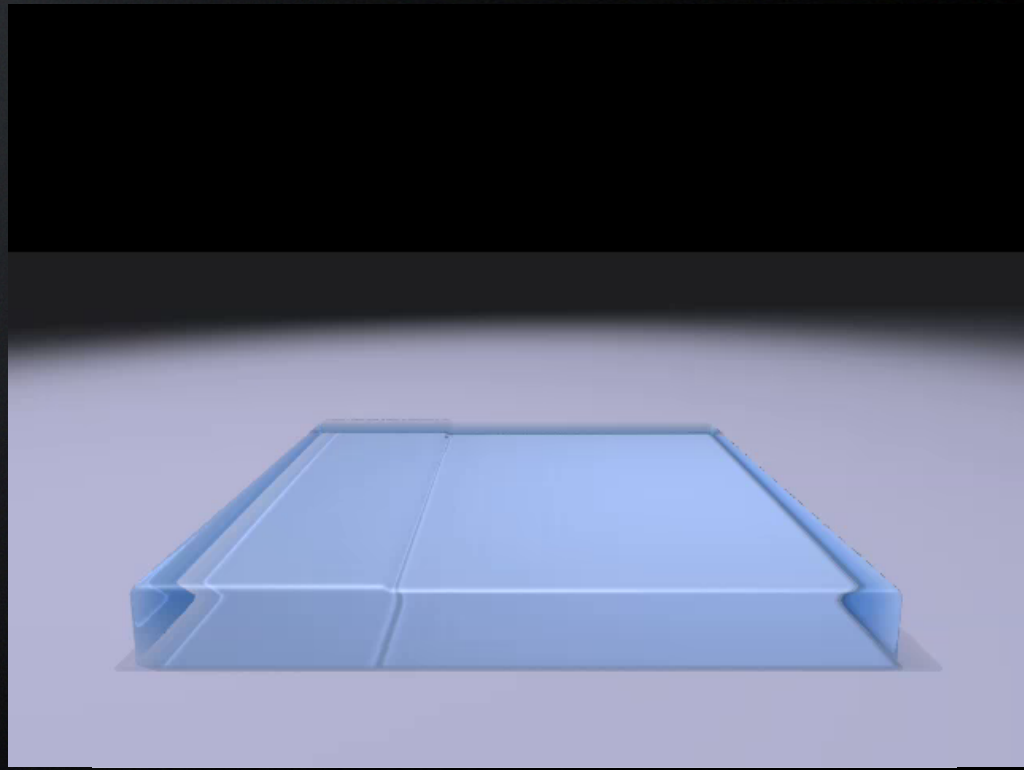
GRANULAR MATERIAL

- Distinct Element Method (DEM)
- Interaction force
  - Damped spring



FLUID SIMULATION

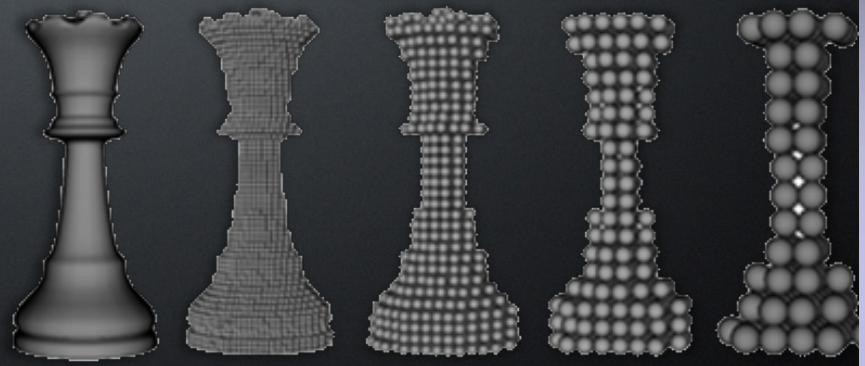
- Smoothed Particle Hydrodynamics
- Solving the Navier-Stokes equation on particles

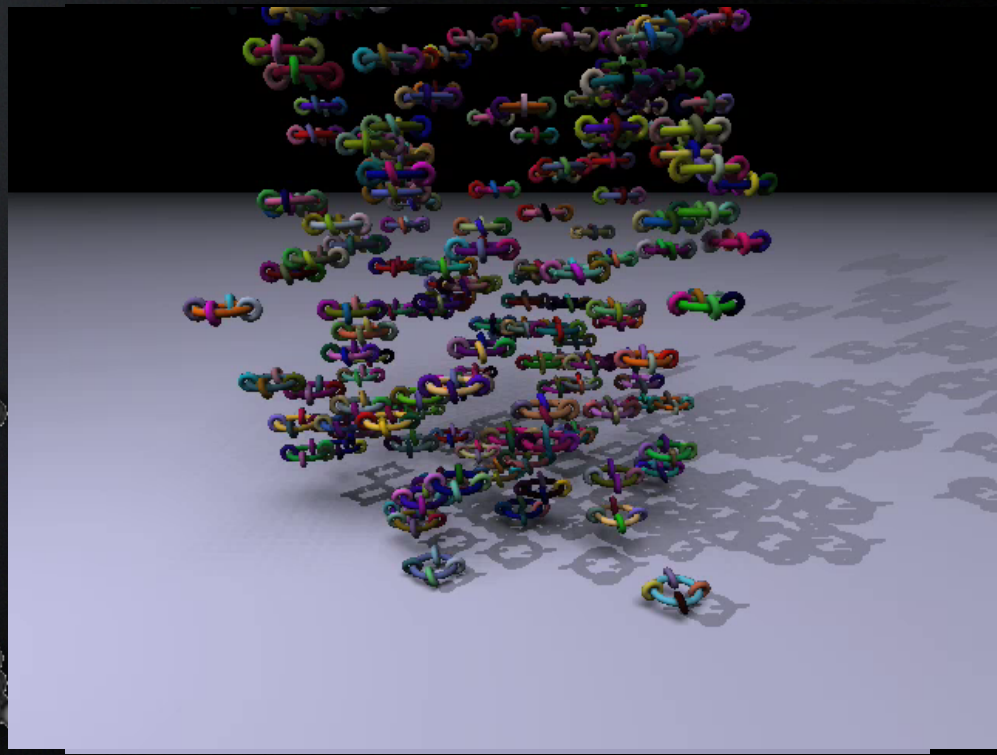


Harada et al., Smoothed Particle Hydrodynamics on GPUs, CGI (2007)

RIGID BODY SIMULATION

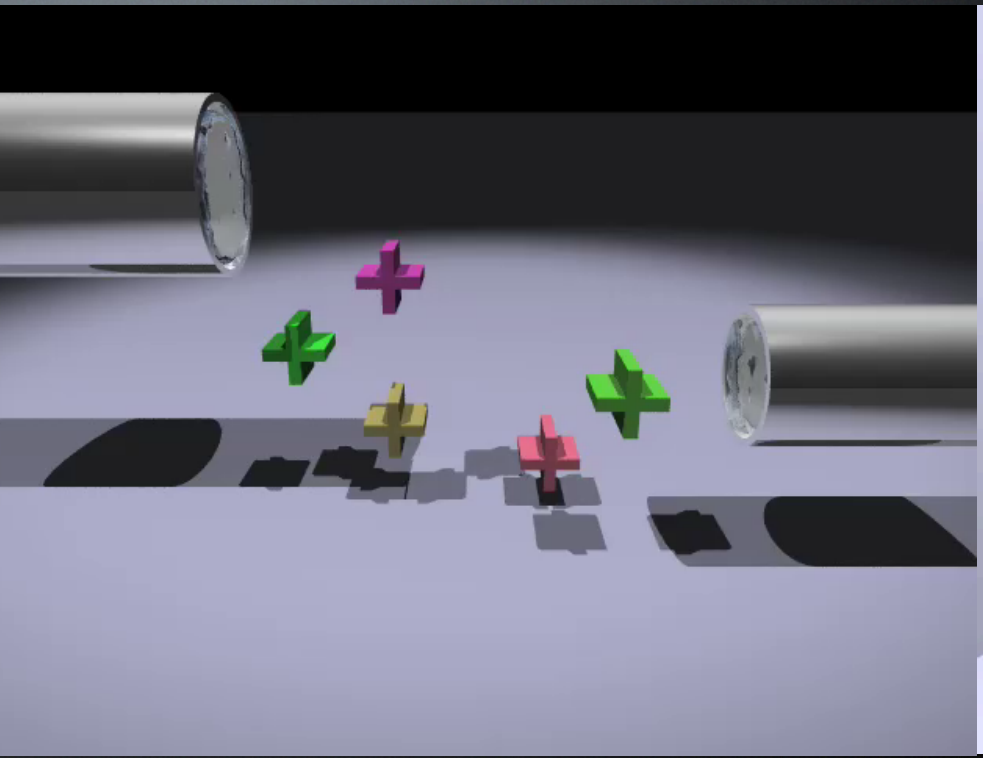
- Represent a rigid body with a set of particles
- Rigid body collision = Particle collision





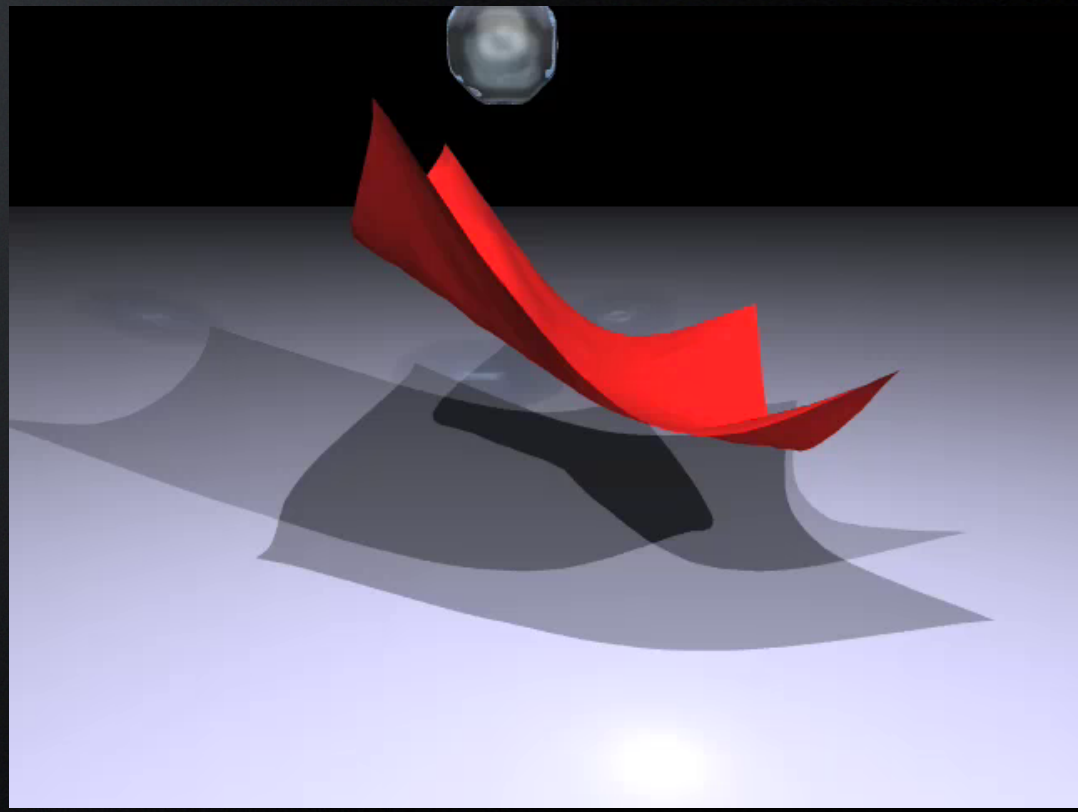
Harada et al., Real-time Rigid Body Simulation on GPUs, GPU Gems3 (2007)

COUPLING RIGID BODIES + FLUID



CLOTH COLLISION DETECTION

- Particle v.s. triangle mesh collision
- Multiple uniform grids
  - 1st grid: particles
  - 2nd grid: triangles



Harada et al., Real-time Fluid Simulation Coupled with Cloth, TPCG (2007)

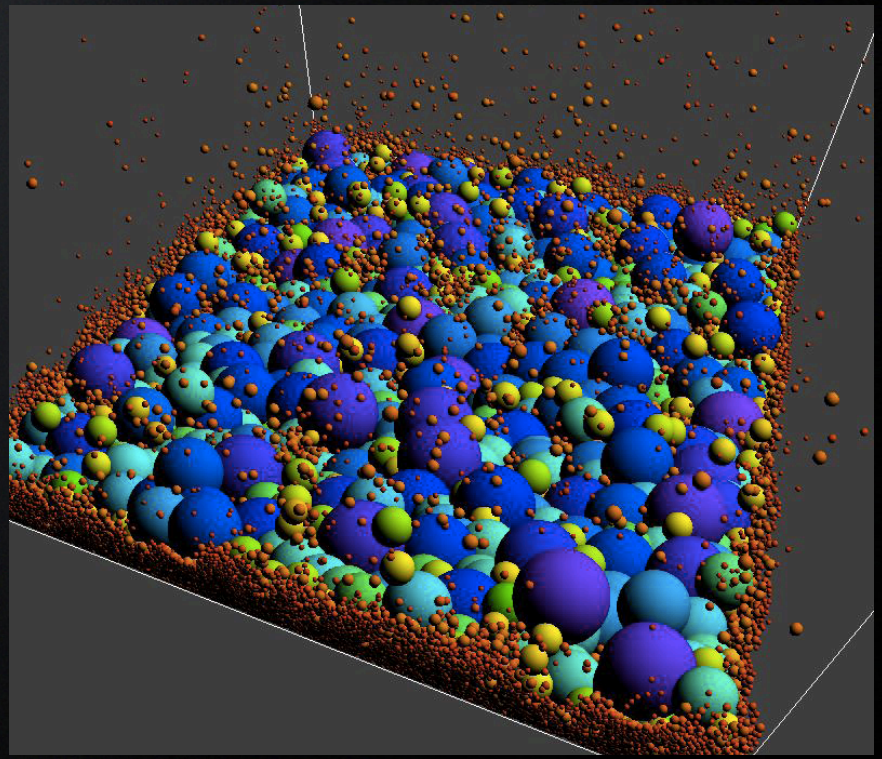
OTHER DATA STRUCTURE

- Grid
- Sweep & prune
  - Sweep
- Bounding volume hierarchy
  - Build

- Rigid body simulation
  - Broad-phase collision detection
  - Narrow-phase collision detection

BROAD-PHASE COLLISION DETECTION

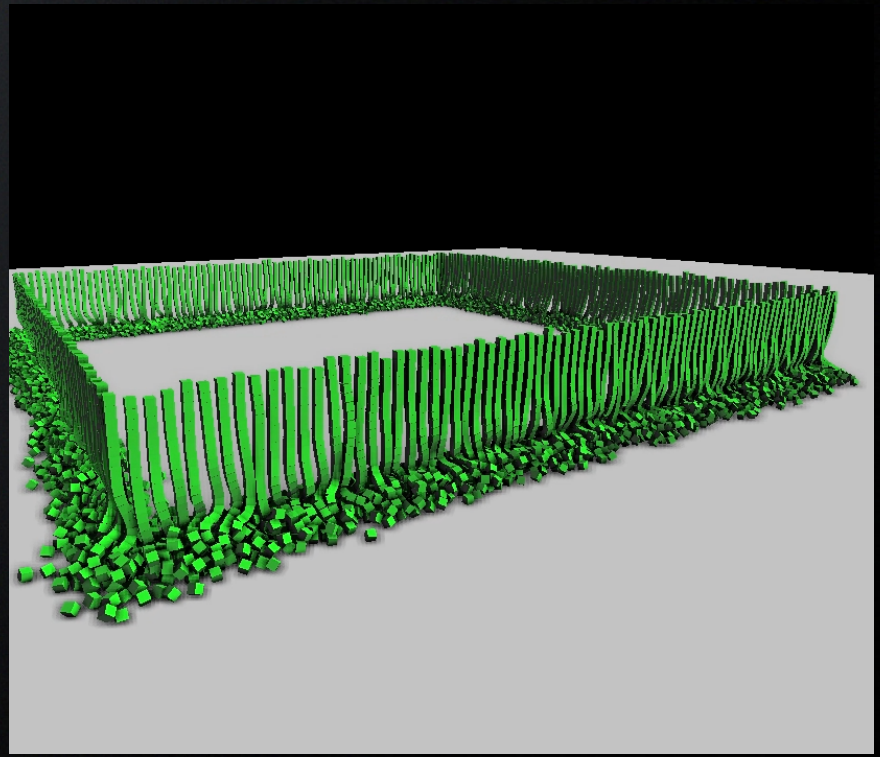
- Sweep & prune
  - Sort end points
  - Sweep sorted list
- Optimizations
  - Split sweep of a large object
  - Workspace subdivision
  - Cell subdivision
- Detail
  - Young J. Kim's presentation



Liu et al., Real-time Collision Culling of a Million Bodies on Graphics Processing Units, TOG(2010)

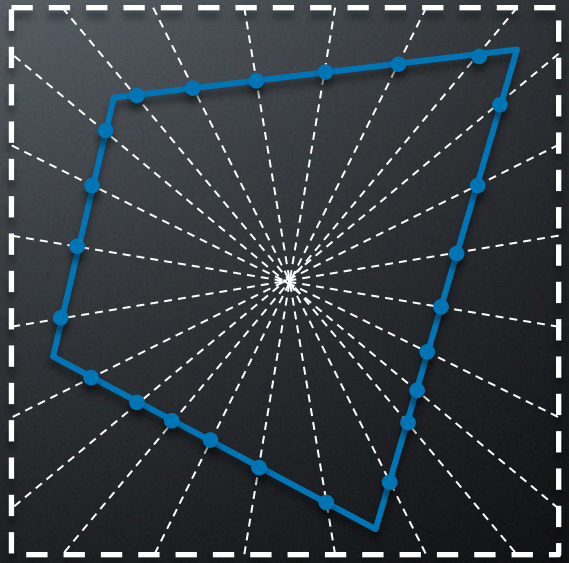
NARROW-PHASE COLLISION DETECTION

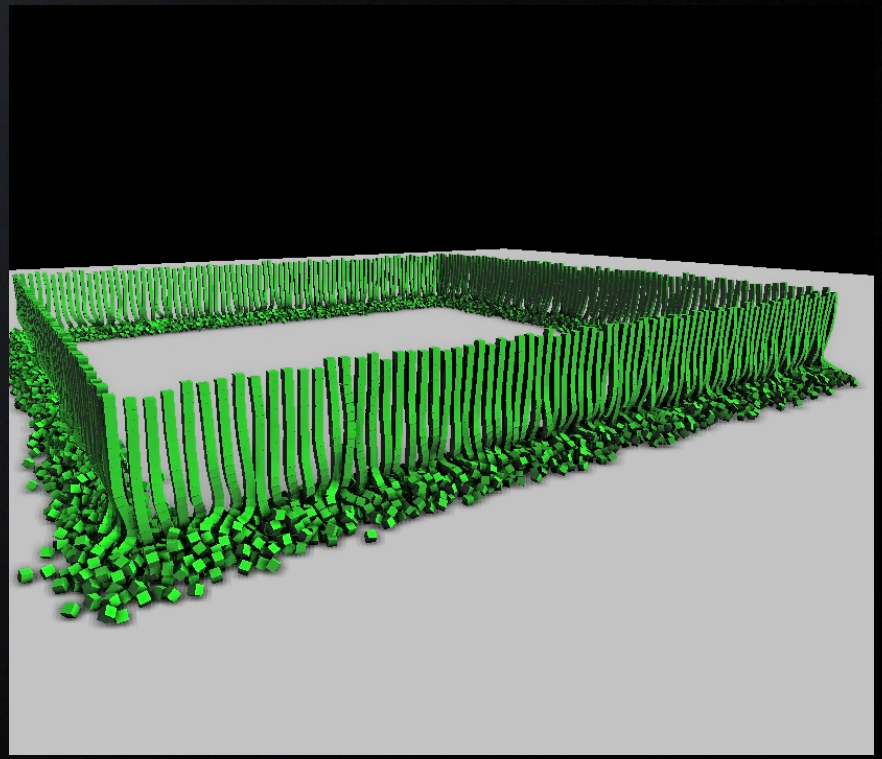
- Variety of computation
  - Many shape combinations -> divergence
    - Box-box, box-sphere, box-convex, …
  - Sequential algorithm
    - GJK



NARROW-PHASE COLLISION DETECTION

- Unified shape representation
- Parallelize per pair, parallelize in a pair





NARROW-PHASE COLLISION DETECTION

- Distance Query

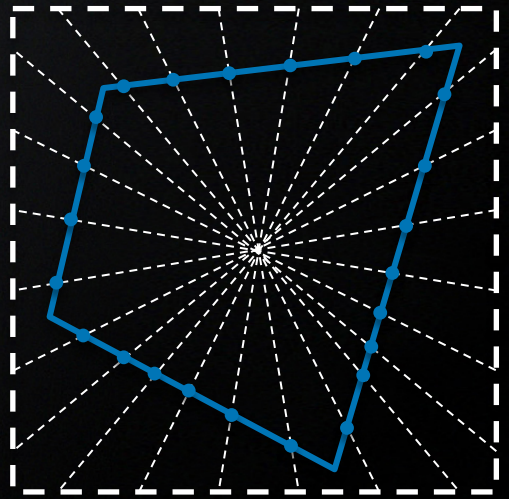
```
Parallel For All Surface Points
  d = Calculate distance
  if( d < surface height )
        Append Contact Point
```

- Contact point reduction

```
p0 = Parallel Reduce(Direction0)
p1 = Parallel Reduce(Direction1)
p2 = Parallel Reduce(Direction2)
p3 = Parallel Reduce(Direction3)
```

- Export

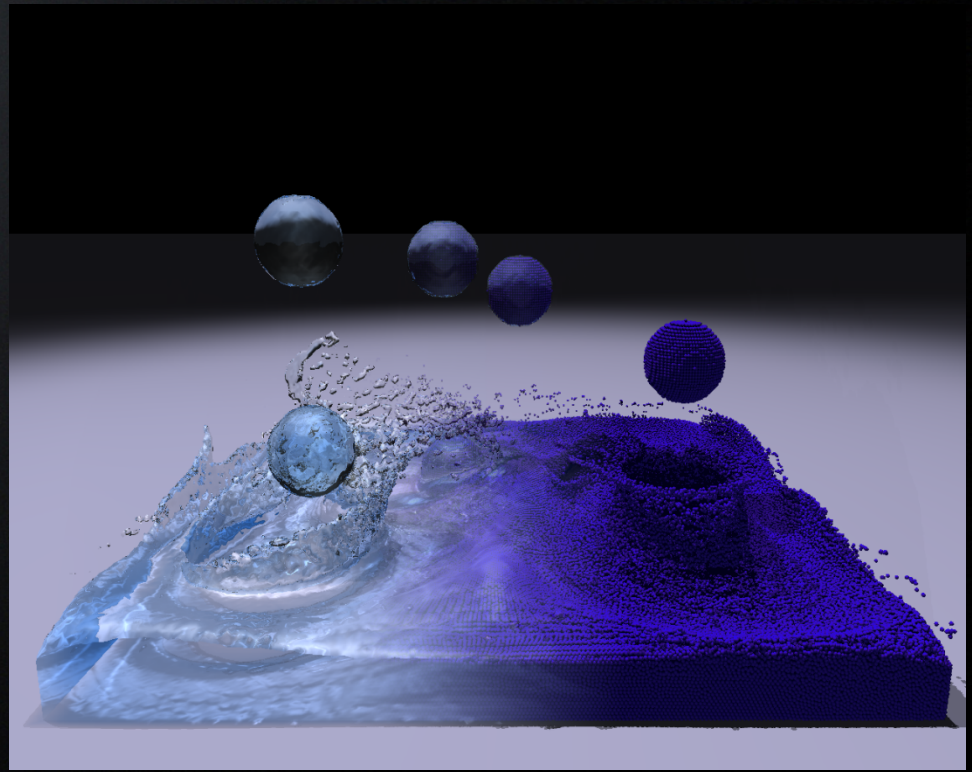
```
Append contact(p0, p1, p2, p3)
```



HETEROGENEOUS COLLISION DETECTION

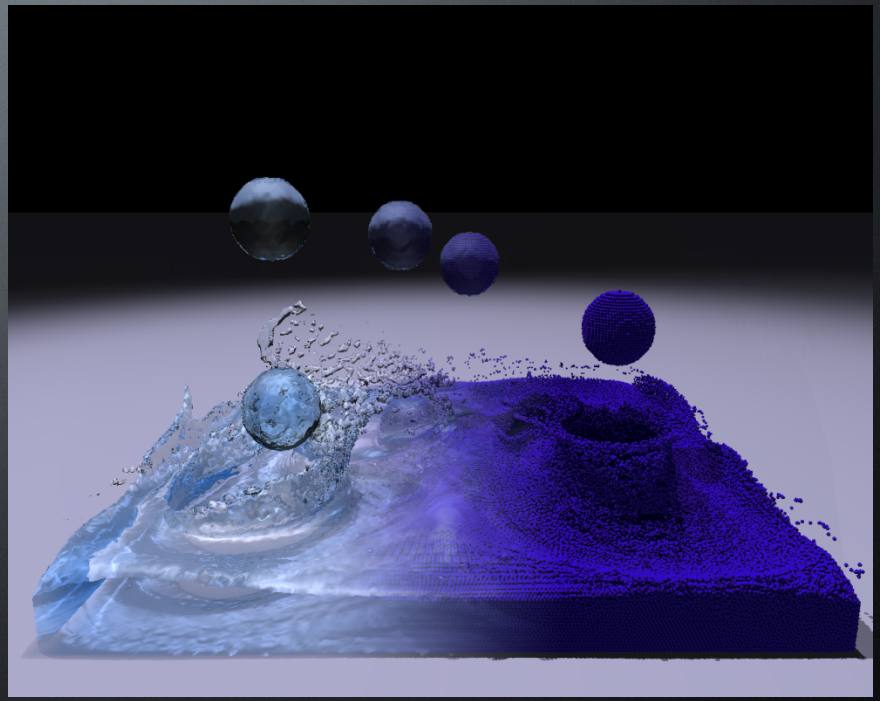
PARTICLE BASED SIMULATION ON THE GPU

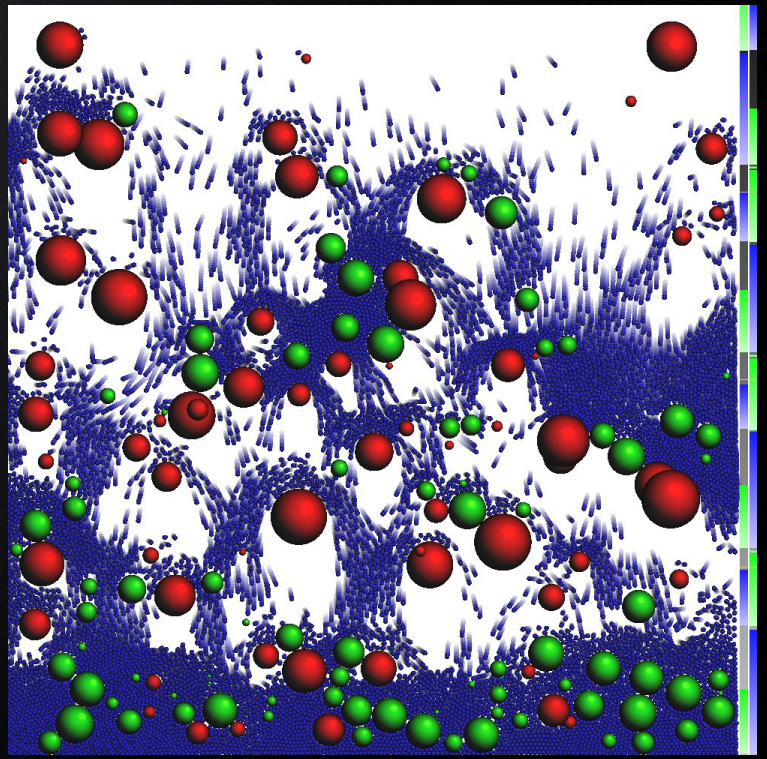
- Large number of particles
- Particles with identical size
  - Work granularity is almost the same
  - Good for the wide SIMD architecture



Harada et al. 2007

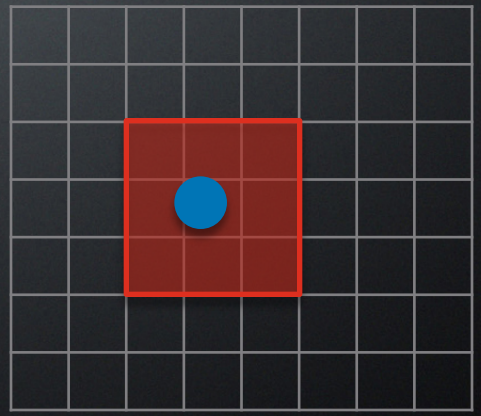
DIFFERENT PARTICLE SIMULATIONS

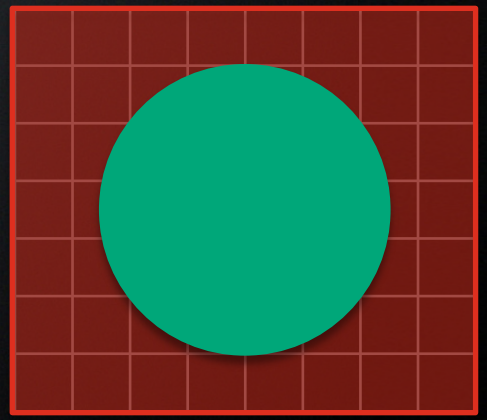


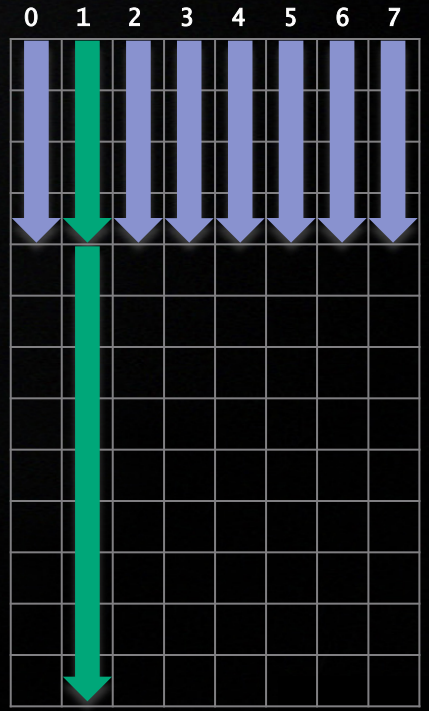


MIXED PARTICLE SIMULATION

- Not only small particles
- Difficulty for GPUs
  - Large particles interact with small particles
  - Large-large collision

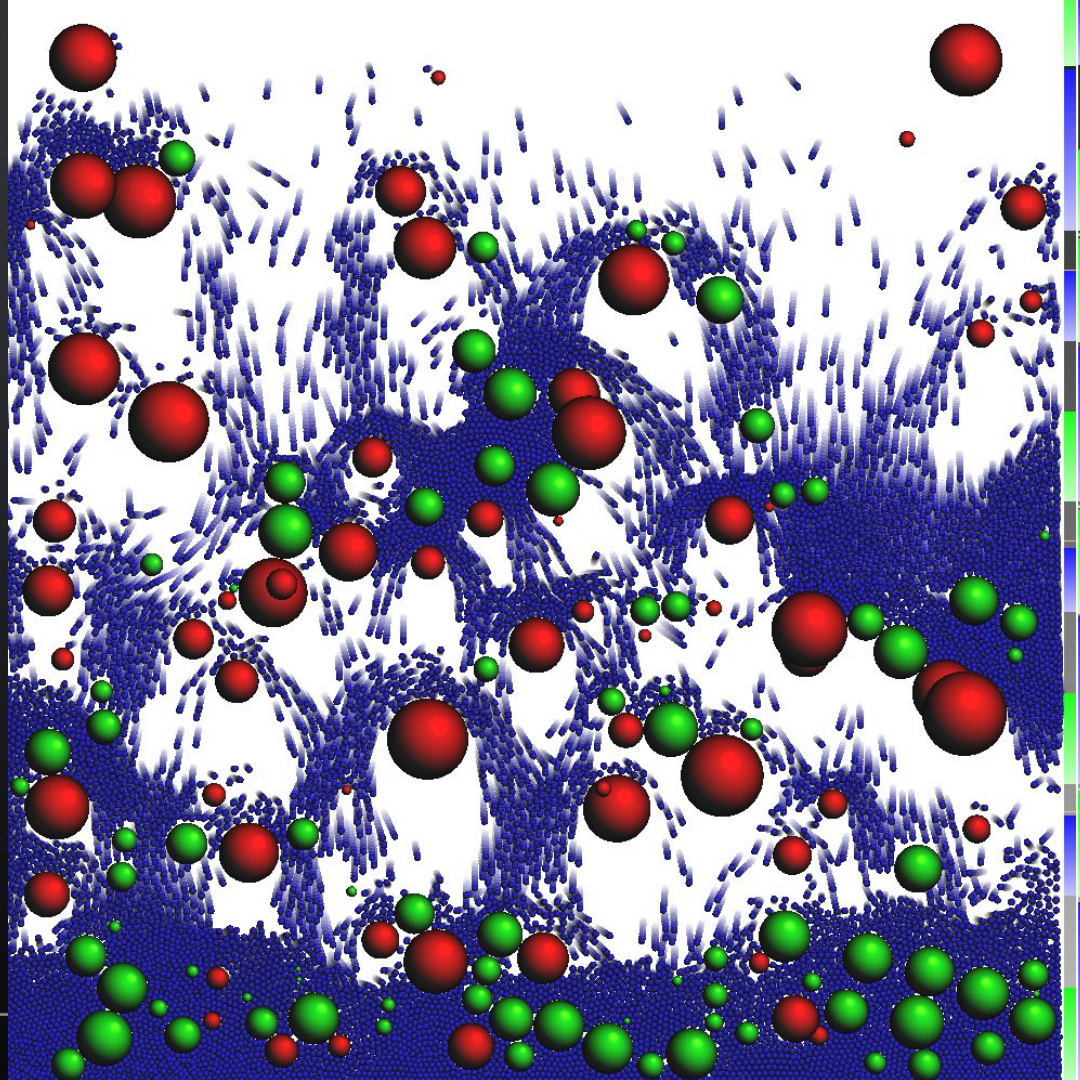






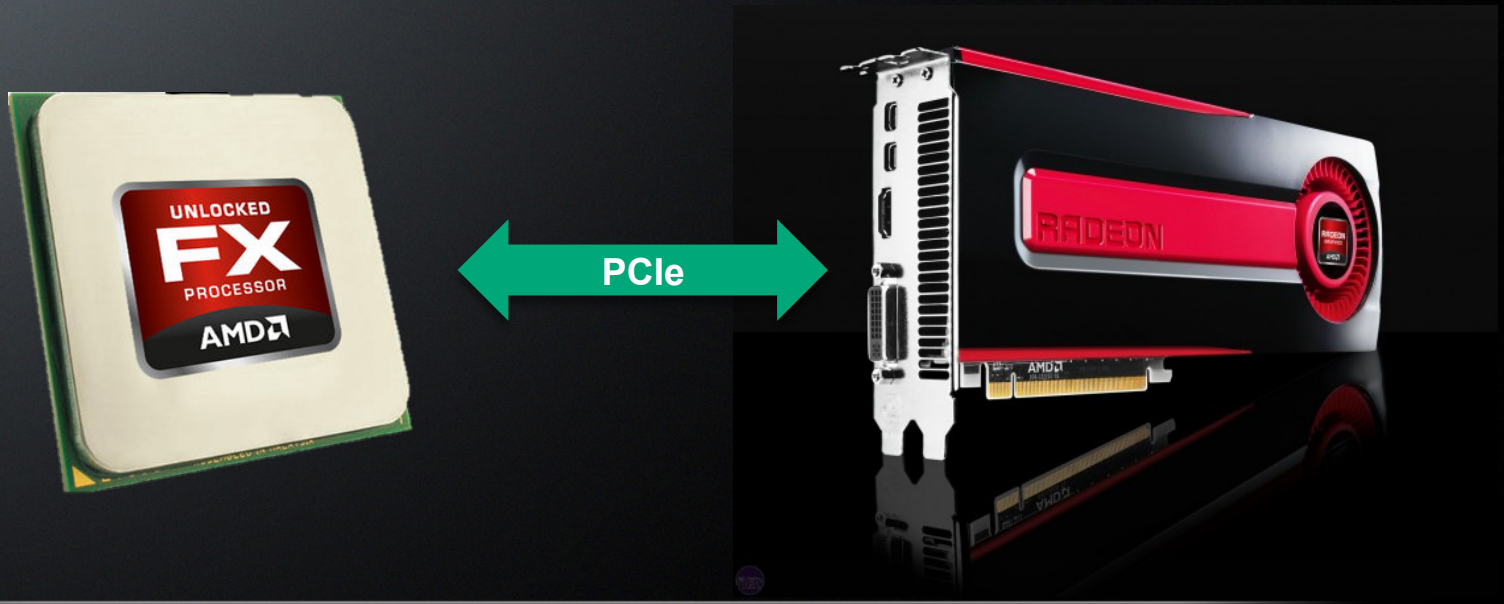
CHALLENGE

- Non uniform work granularity
  - Small-small(SS) collision
    - Uniform, GPU
  - Large-large(LL) collision
    - Non Uniform, CPU
  - Large-small(LS) collision
    - Non Uniform, CPU
- Can use a CPU and a discrete GPU
  - PCIe data transfer bottleneck



SOLVING NON-UNIFORMITY

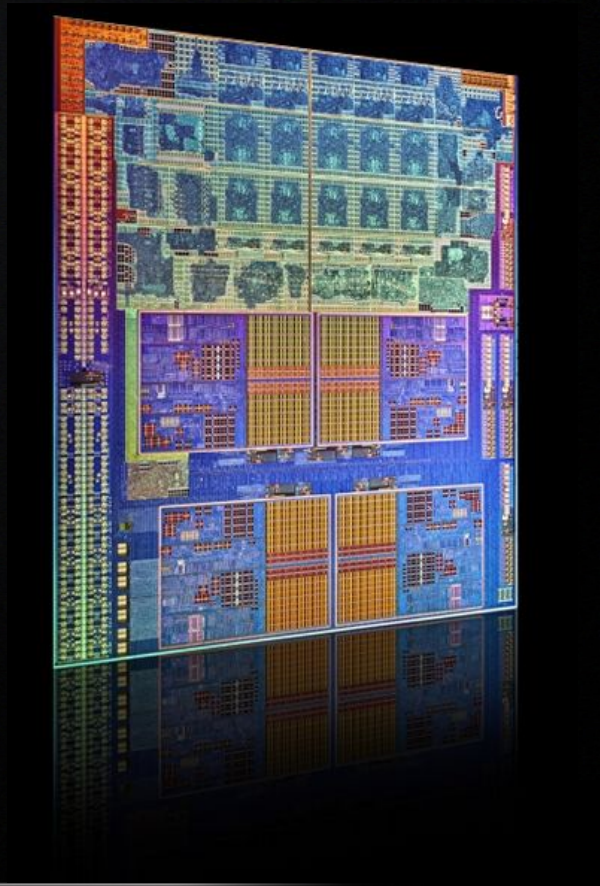
- If we can use CPU and GPU, problem is solved
- PCI Express data transfer kills the performance



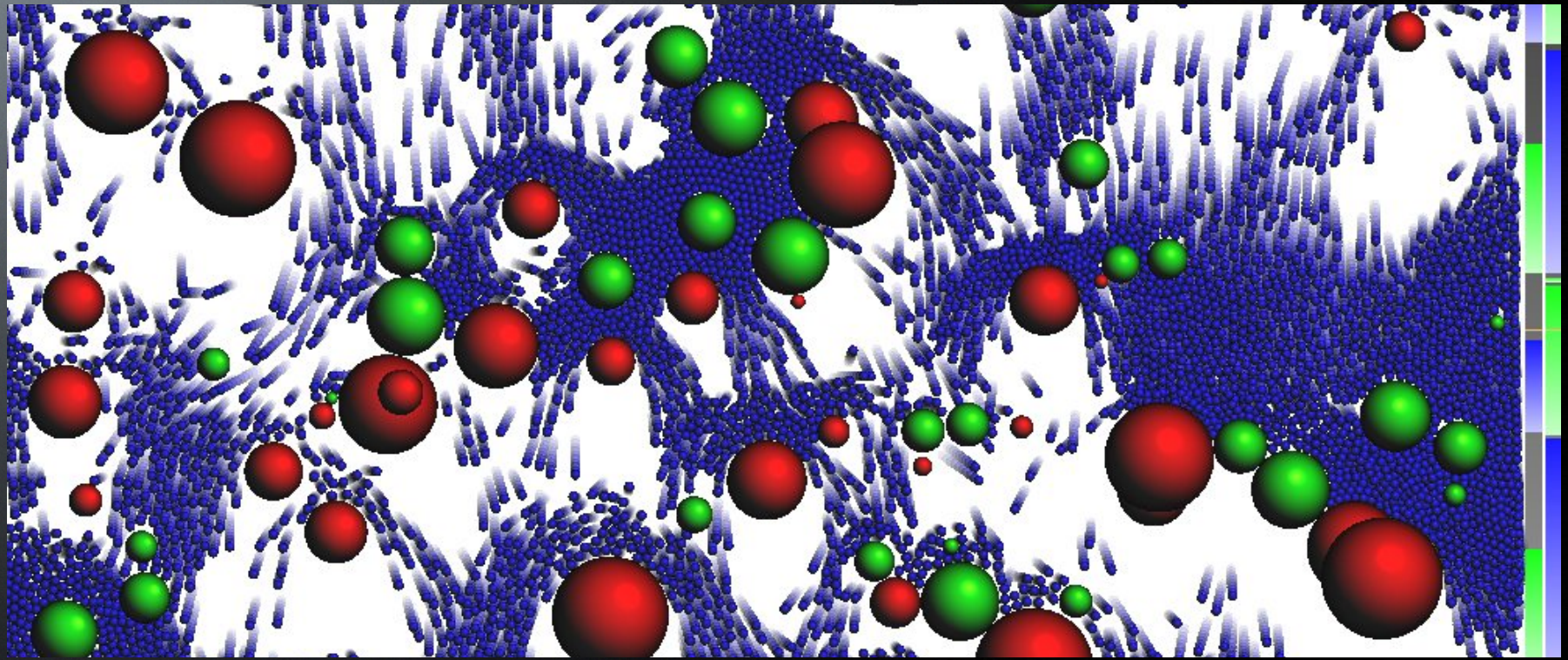
HETEROGENEOUS ARCHITECTURE

- CPU + GPU on the same die
- AMD Fusion (APU), Intel Sandybridge
- No PCI Express bus
- Memory sharing
- Able to dispatch a job to the GPU without data transfer
  - Serial work: dispatch to the CPU
  - Parallel work: dispatch to the GPU

- Can use this for a non-uniform simulation



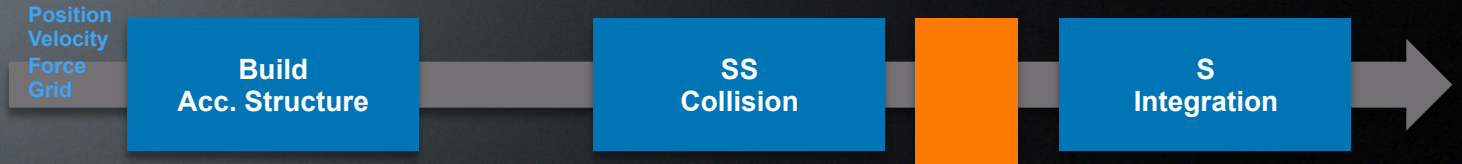
DIFFERENT PARTICLE SIMULATIONS



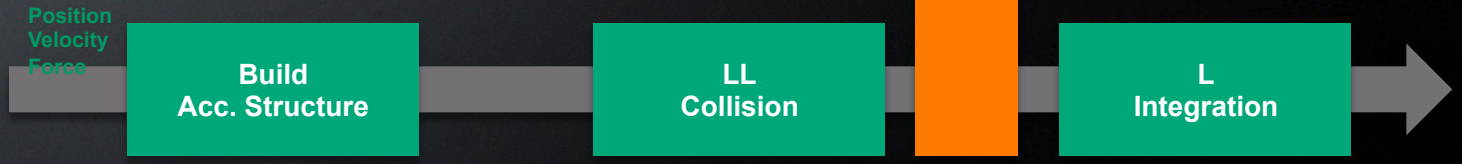
METHOD

TWO SIMULATIONS

- Small particles

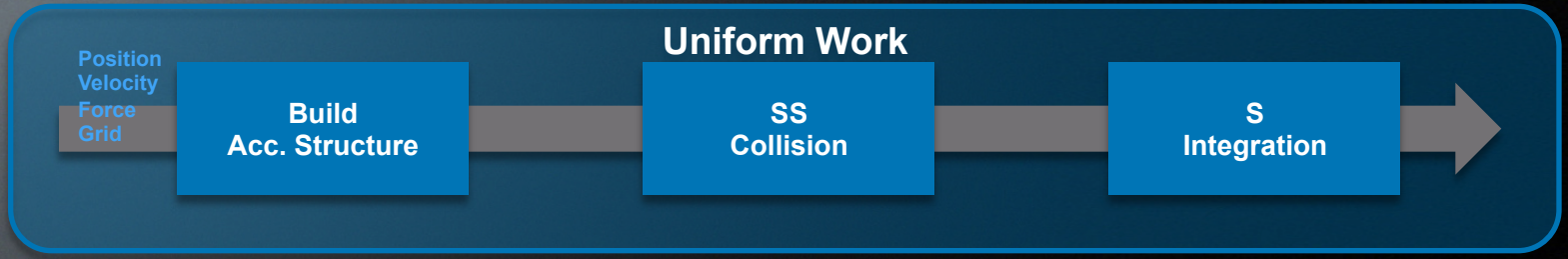


- Large particles

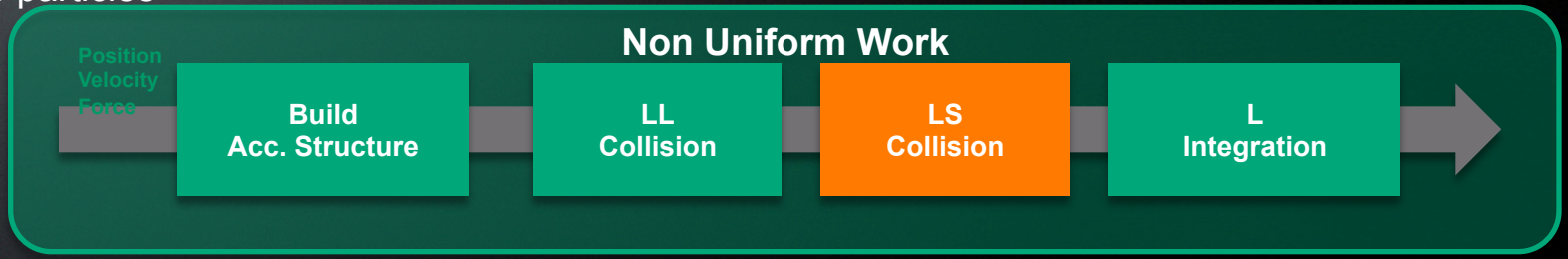


CLASSIFY BY WORK GRANULARITY

- Small particles

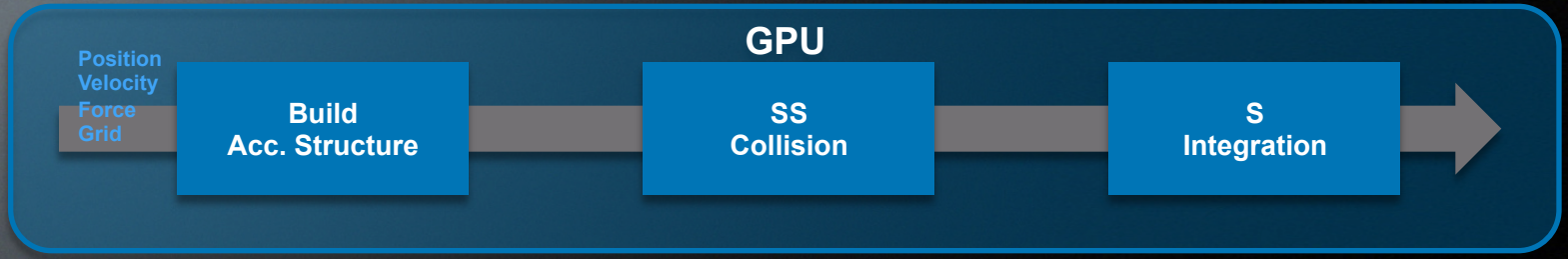


- Large particles

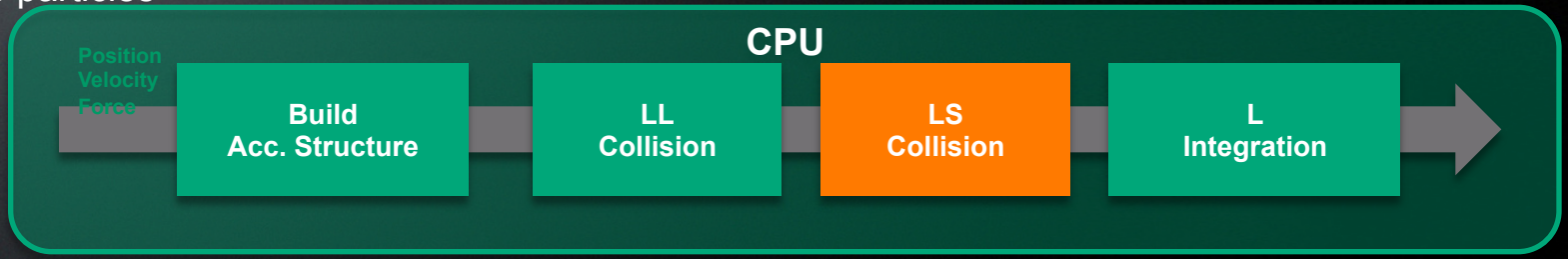


CLASSIFY BY WORK GRANULARITY, ASSIGN PROCESSOR

- Small particles

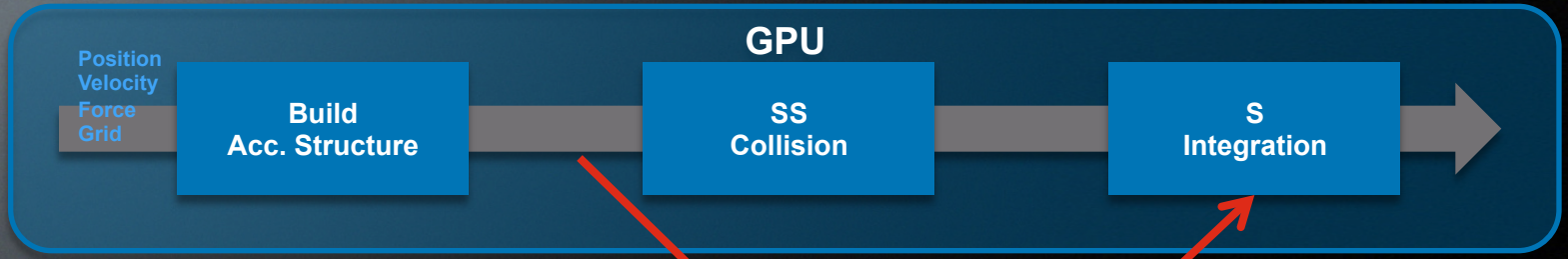


- Large particles

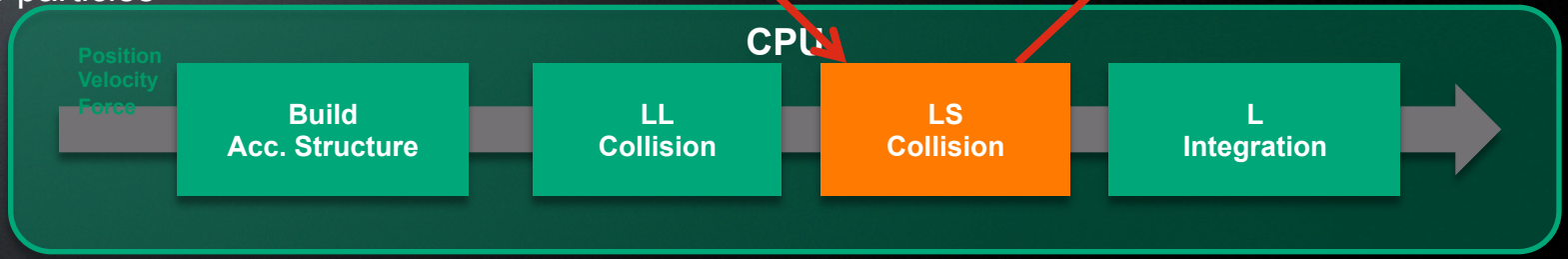


DATA SHARING

- Small particles



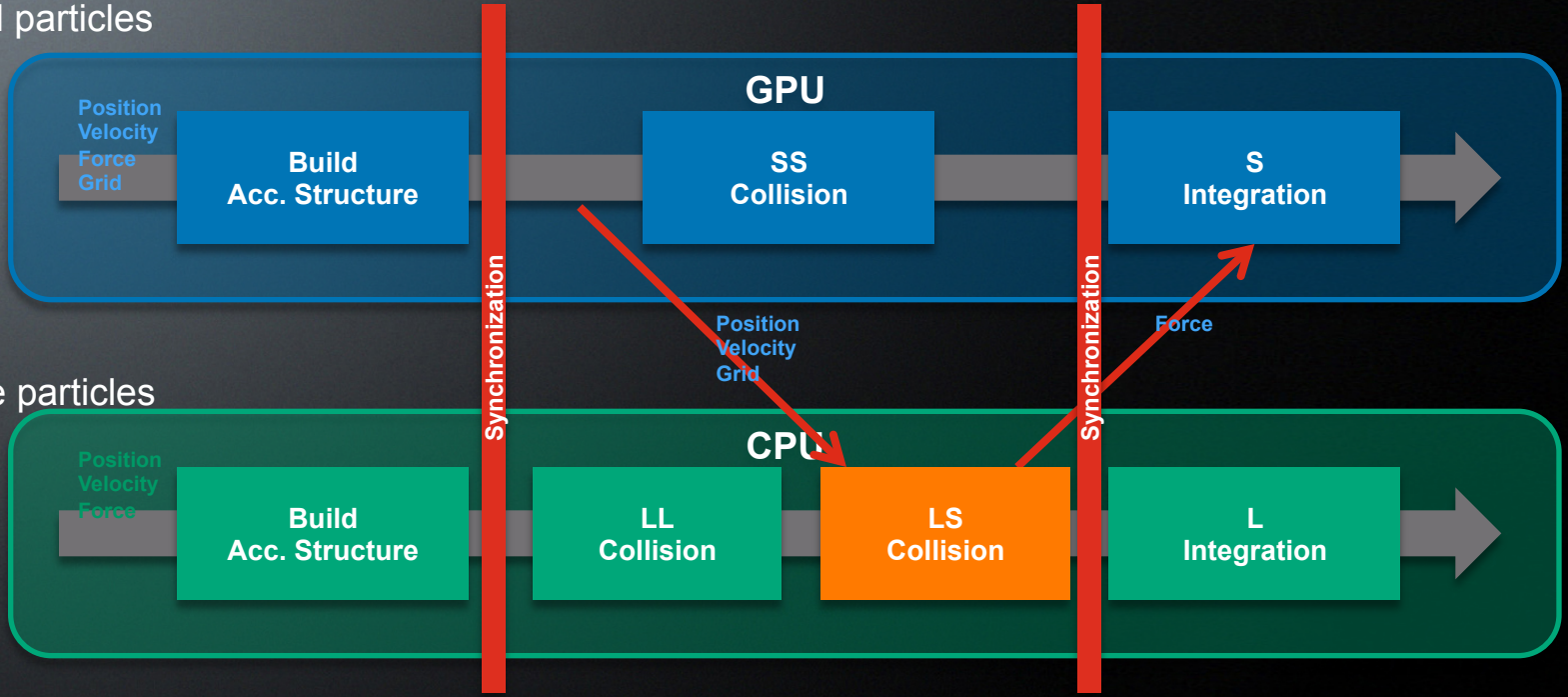
- Large particles



- Grid, small particle data has to be shared with the CPU for LS collision
  - Allocated as zero copy buffer

SYNCHRONIZATION

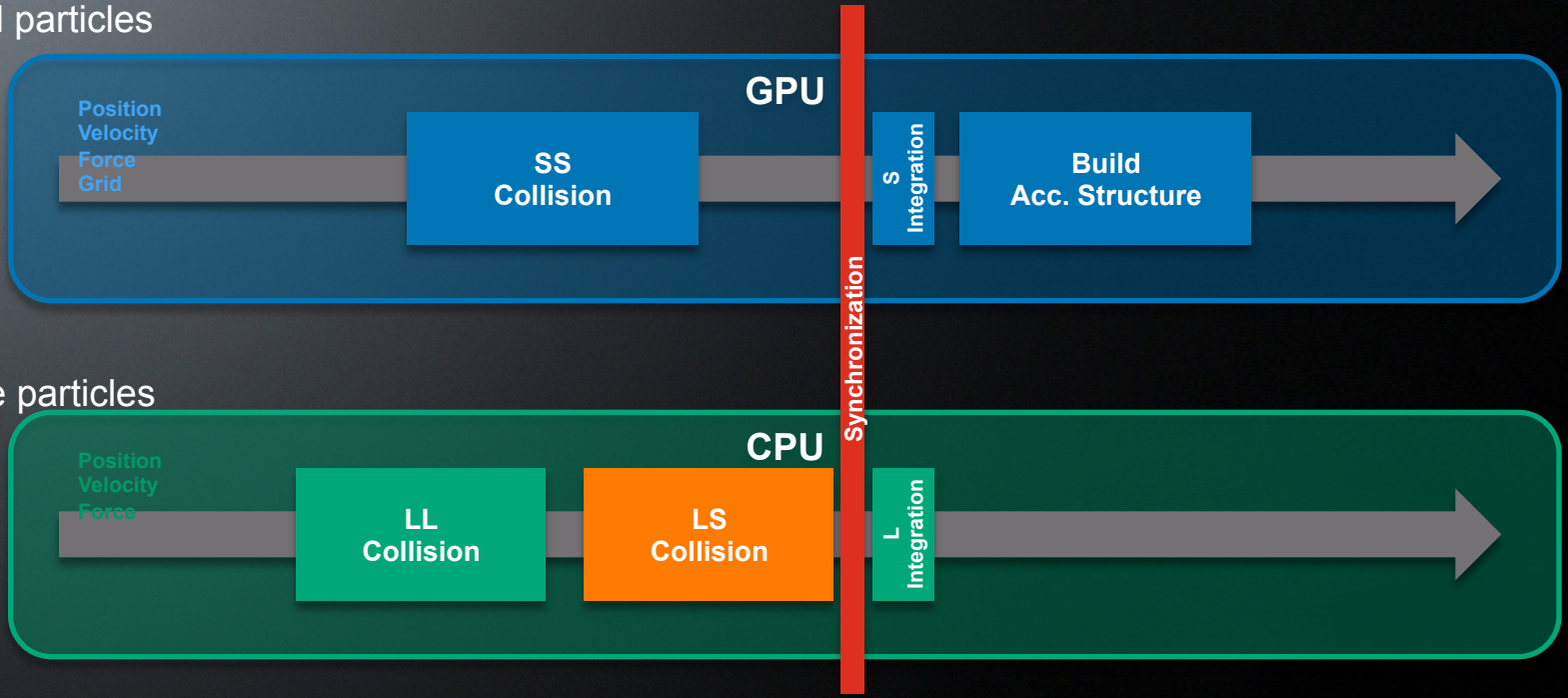
- Small particles
- Large particles



- Grid, small particle data has to be shared with the CPU for LS collision
  - Allocated as zero copy buffer

VISUALIZING WORKLOADS

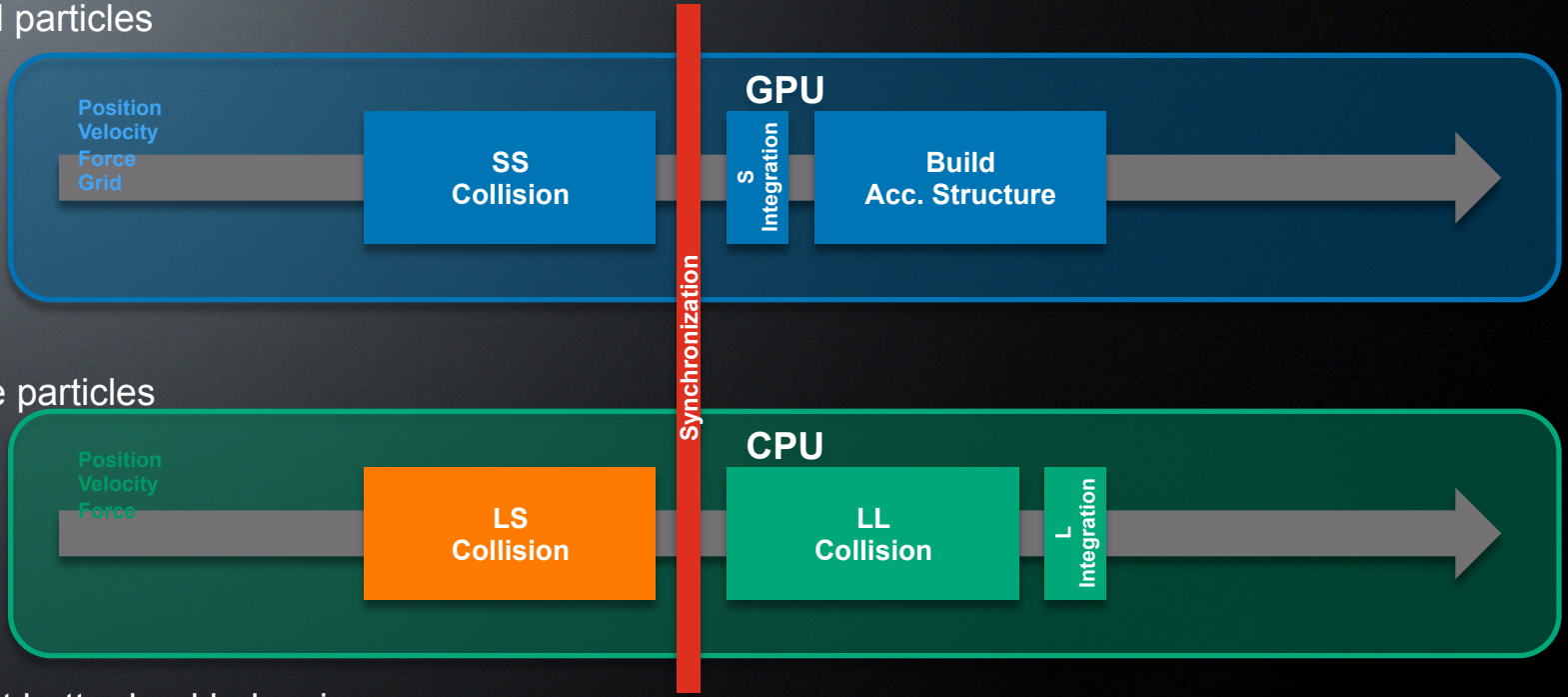
- Small particles



- Large particles
- Grid construction can be moved at the end of the pipeline
  - Unbalanced workload

LOAD BALANCING

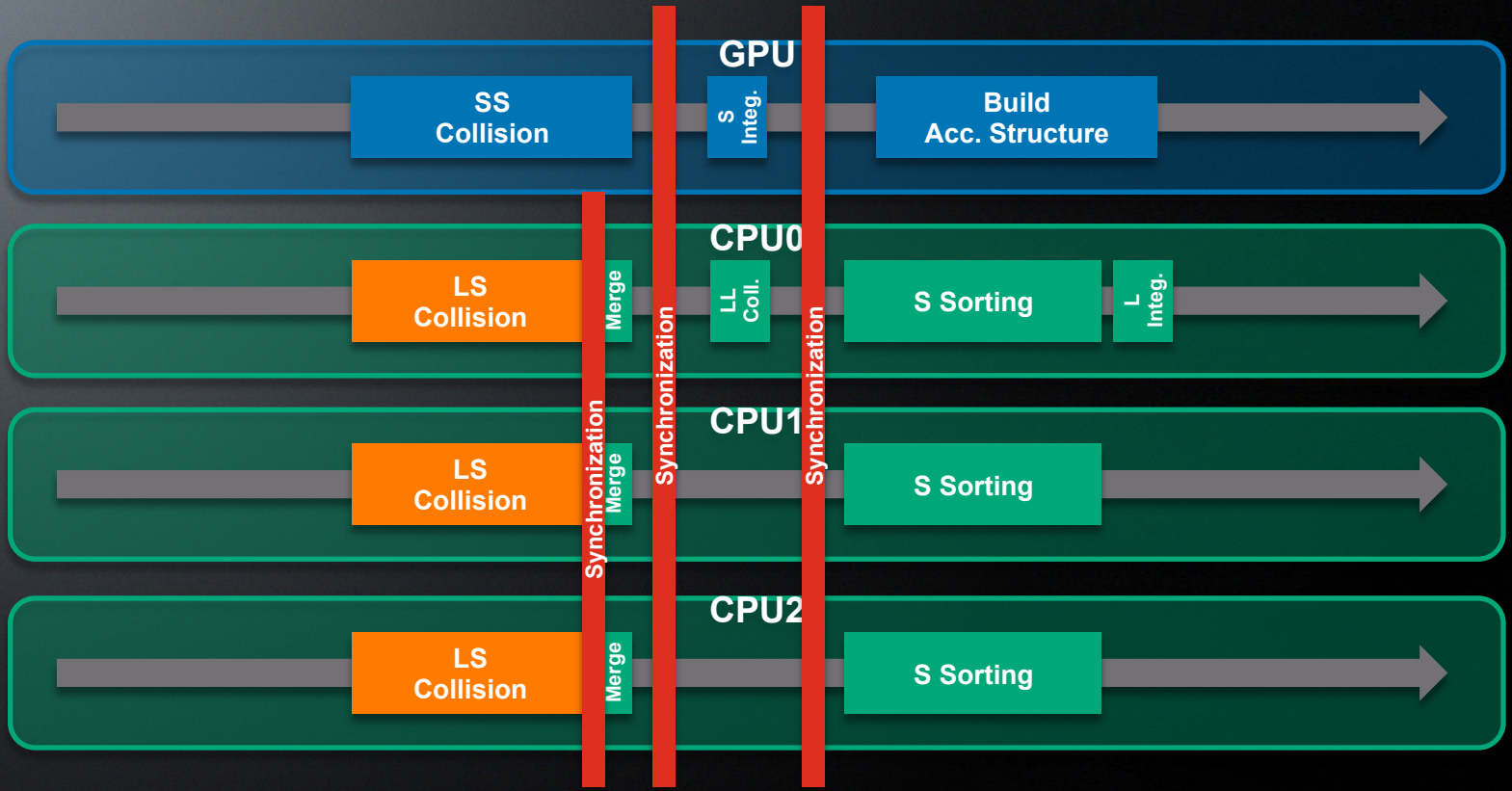
- Small particles



- Large particles
- To get better load balancing
  - The sync is for passing the force buffer filled by the CPU to the GPU
  - Move the LL collision after the sync

MULTI THREADING (4 THREADS)

OPTIMIZATION2: IMPROVING SMALL-SMALL COLLISION



Harada, Heterogeneous Particle-Based Simulation, SIG ASIA Talk(2011)

aphics 2012

May 13-18

OPEAN ASSOCIATION FOR COMPUTER GRAPHICS

CPU Work

GPU Work

QUESTIONS?

PROBLEM SOLVED??

- Uniform grid can solve a lot of problems
- However, uniform grid is not perfect solution
- More sophisticated collision detection is necessary

- Rigid body simulation
  - 2 step collision detection
  - Broad-phase collision detection
    - Grid, Sweep & prune, Tree
  - Narrow-phase collision detection
    - SAT, GJK, approximated solutions