Accelerating Rigid Body Simulation on Today’s GPUs

Takahiro Harada
takahiro.harada@amd.com
RIGID BODY SIMULATION PIPELINE

- Broad phase collision detection
  - Quick check using bounding volumes
- Narrow phase collision detection
  - Detailed check using geometry
- Solve
**WHY USE GPU?**

- Large scale simulation requires a lot of bodies
  - Destruction
- Increase the cost of a simulation

- GPU has high-
  - Peak performance
  - Memory bandwidth

- GPU is different from CPU
GPU RIGID BODY TECHNIQUES

- Broad phase
- Narrow phase
- Solve

GPU Graphics

GPU Compute
GPU RIGID BODY TECHNIQUES

- Rahul Sathe, Rigid Body Collision Detection on the GPU, Sig Poster(2006)
Takahiro Harada et al., Smoothed Particle Hydrodynamics on GPUs, Proc. of CGI(2007)
GPU RIGID BODY TECHNIQUES

- Takahiro Harada, Real-time Rigid Body Simulation on GPUs, GPU Gems 3 (2007)

**Broad phase**
- SPH
  - Harada (2007)
  - Uniform Grid (Atomics)

**Narrow phase**
- Cube map
  - Sathe (2006)
- Cube map
- Particles

**Solve**
- Penalty

**GPU Graphics**
**GPU Compute**
### GPU RIGID BODY TECHNIQUES

<table>
<thead>
<tr>
<th>Broad phase</th>
<th>Narrow phase</th>
<th>Solve</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPH</td>
<td>Cube map</td>
<td>Penalty</td>
</tr>
<tr>
<td>Uniform Grid (Atoms)</td>
<td>Cube map</td>
<td></td>
</tr>
<tr>
<td>Sort based grid</td>
<td>Particles</td>
<td></td>
</tr>
<tr>
<td>Le Grand(2007)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uniform Grid (Sort)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Particle based rigid body</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harada(2007)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

# GPU Rigid Body Techniques

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Uniform Grid (Atoms)</td>
<td>Uniform Grid (Sort)</td>
</tr>
<tr>
<td>Particle based rigid body</td>
<td>Harada(2007)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cube map</td>
<td>Particles</td>
</tr>
<tr>
<td>Solve</td>
<td>Penalty</td>
<td></td>
</tr>
</tbody>
</table>

**GPU Graphics**

**GPU Compute**

### GPU RIGID BODY TECHNIQUES

#### Broad phase
- **SPH**
  - Harada (2007)
  - Uniform Grid (Atomics)
- **Sort based grid**
  - Le Grand (2007)
  - Uniform Grid (Sort)

#### Narrow phase
- **Cube map**
  - Harada (2007)
  - Sathe (2006)
  - Cube map
  - Particles
  - Geometry

#### Solve
- **SPH**
  - Harada (2007)
  - Partially Serial Batching
- **Penalty**
- **Constraint**

---

- **Takahiro Harada**, Parallelizing the Physics Pipeline, GDC (2009)

---

**Figures:**
- Demo in 2D
- SIGGRAPH 2011
Richard Tonge, PhysX GPU Rigid Bodies in Batman, Game Programming Gems 8(2010)
**GPU RIGID BODY TECHNIQUES**

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

---

**Broad phase**

- **SPH**
  - Harada(2007)
  - Uniform Grid (Atomsics)

- **Sort based grid**
  - Le Grand(2007)
  - Uniform Grid (Sort)

- **GPU Sweep & Prune**
  - Liu(2010)
  - Sweep & Prune

**Narrow phase**

- **Cube map**
  - Harada(2007)
  - Sathe(2006)

- **Particles**
  - Geometry

- **LCP**
  - Kipfer(2007)

**Solve**

- **Batching**
  - Harada(2009)
  - Tonge(2010)

- **Partially Serial Batching**

- **Parallel Batching**

- **Penalty**
  - Constraint

- **Constraint**
Architecture & Algorithm
GPU RIGID BODY TECHNIQUES

- Used GPU as a processor with many ALUs
GPU ARCHITECTURE

Compute Unit (CU)

Global Memory
**GPU Rigid Body Techniques**

<table>
<thead>
<tr>
<th>Broad phase</th>
<th>Narrow phase</th>
<th>Solve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle based rigid body</td>
<td>LCP Kipfer (2007)</td>
<td>Constraint</td>
</tr>
<tr>
<td>Uniform Grid (Atomics)</td>
<td>Geometry</td>
<td>Constraint</td>
</tr>
</tbody>
</table>

- **LCP**
  - A CU processes a pair
  - Synchronization
  - LDS
- **Broad phase collision**
  - Radix sort

**Notes:**
- **SPH**
  - Particle-based rigid body simulation
- **LCP**
  - Linear Complementarity Problem
  - Used for collision detection

**GPU Architecture**

- **GPU Graphics**
- **GPU Compute**
GPU RIGID BODY TECHNIQUES

- Solver using CU
- Approximated narrowphase using CU
  - Performance

- Broad phase
    - Uniform Grid (Atomics)
  - Particle based rigid body: Harada(2007)

- Narrow phase
  - Cube map: Sathe(2006)
  - Geometry: LCP Kipfer(2007)
  - Geometry representation for GPU

- Solve
  - Penalty
  - Partially Serial Batching: Harada(2009)
  - Parallel Batching: Tonge(2010)
  - Batching

- GPU Graphics
- GPU Compute
Compute Unit Aware
Rigid Body Simulation
- Copy body and pair buffer
- GPU allocates big buffers
  - Contact
  - Constraints
- Narrow phase and solve is done on the GPU
- Don’t have to read back big buffers
NARROWPHASE – CU LEVEL PARALLELIZATION

- Collision of a pair is independent

- Use a SIMD lane for a pair
  - Not enough resource for a SIMD lane

- Use a CU for a pair

- A CU processes and “append” colliding pairs
  - HW accelerated append operation on AMD

- Load balancing
  - CU can fetch a pair from queue
  - Explicit pair split

- CU level parallelization
- A collision has to be split into parallel works
- Parallel collision detection
- Support arbitrary convex shapes
- Too many contact points
  - Increase cost of solver
- Redundancy
  - Use sync and LDS for vector wide operation
  - Eliminate redundant contacts
SHAPE REPRESENTATION
SHAPE REPRESENTATION
**SOLVER**

- Constraint based solver uses a Gauss Seidel method
  - Velocity is input and output
  - <-> Penalty method
- Sequential solve
- Parallel solve is possible only if
  - Bodies are not shared among constraints
- When a body is colliding to several bodies, parallel solve cannot be used
- Solution
  - Split constraints into batches
  - Constraints in a batch doesn’t share bodies
BATCHING

- 2 level batching
  - Global batching
    - Split into independent sections
  - Local batching
    - Batch in each section
ITERATIVE LOCAL BATCHING

- A CU processes a section of contact buffer
  - Local batching <-> Global batching
  - Extract independent pairs in a set

- Stream processing pair buffer
  - Read to local, reorder

- Pairs are localized
  - X Parallel batching
  - X Serial batching
  - O Iterative batching
A CU processes a section of contact buffer

- GPU dispatches works by itself
  - Read constraints
  - Fill batch buffer
  - Solve
  - If the batch is done, go to the next batch

Previous work
- CPU had to dispatch a kernel per batch

Our method
- CPU dispatches some
- GPU dispatches works by itself
DEMO

- OpenCL
- Radeon HD5870
- 20K objects
- About 30fps
- GPU collide and solve ≈ 30ms (Inc. data transfer)
DEMO

- OpenCL
- Radeon HD5870
- 12K objects
- About 30fps
- GPU collide and solve ≈ 30ms (Inc. data transfer)
OFFLINE DEMO

- 400K objects
- GPU collide and solve < 1s
LIMITATIONS

- GPU prefers uniform work granularity
- Some works are better to use CPUs
### HETEROGENEOUS ERA

<table>
<thead>
<tr>
<th>Broad phase</th>
<th>Narrow phase</th>
<th>Solve</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPH</strong> Harada (2007)</td>
<td><strong>Cube map</strong> Sathe (2006)</td>
<td><strong>Penalty</strong></td>
</tr>
<tr>
<td><strong>Uniform Grid (Atomics)</strong></td>
<td><strong>Geometry</strong></td>
<td><strong>Constraint</strong></td>
</tr>
<tr>
<td><strong>Sort based grid</strong> Le Grand (2007)</td>
<td><strong>Particles</strong></td>
<td><strong>Parallel Batching</strong></td>
</tr>
<tr>
<td><strong>Uniform Grid (Sort)</strong></td>
<td></td>
<td><strong>Constraint?</strong></td>
</tr>
<tr>
<td><strong>GPU Sweep &amp; Prune</strong> Liu (2010)</td>
<td></td>
<td><strong>Batching?</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Particle based rigid body**

- **Harada (2007)**
- **LCP** Kipfer (2007)
- **Cube map** Sathe (2006)
- **Geometry**

**GPU Graphics**

- **SPH**
- **Sort based grid**
- **GPU Sweep & Prune**

**GPU Compute**

- **Penalty**
- **Constraint**
- **Constraint?**

**Heterogeneous**

- **Representati on for GPU**

**Accelerating Rigid Body Simulation on Today’s GPUs**
HETEROGENEOUS ERA

- AMD Fusion
  - Llano, Zacate
- Advantages
  - Shared memory
  - CPU & GPU can work together

---

**SPH**

Harada (2007)

**LCP**

Kipfer (2007)

**GPU Sweep & Prune**

Liu (2010)

---

**Geometry**

Le Grand (2007)

**Particles**

Sathe (2006)

---

**Parallel Batching**

Tonge (2010)

**Batching**

Batching Partially Serial Batch (2009)

**Constraint**

Batching Parallel Batching

**Representation for GPU**

---

**Uniform Grid (Sort)**

Liu (2010)

**Sort based grid**

Le Grand (2007)

---

**Constraint**

Batching Partially Serial Batch (2009)

**Constraint**

Batching Parallel Batching

**Constraint?**

---

**Sweep & Prune**

GPU Sweep & Prune

**Batching?**

---

**GPU Compute**

---

**Heterogeneous**

---

**Advantages**

- Shared memory
- CPU & GPU can work together

---

**CUDA**

- AMD Fusion
  - Llano, Zacate
  - Advantages
  - Shared memory
  - CPU & GPU can work together

---

**SIGGRAPH 2011**

**SPH**

Harada (2007)

**LCP**

Kipfer (2007)

**GPU Sweep & Prune**

Liu (2010)

---

**Geometry**

Le Grand (2007)

**Particles**

Sathe (2006)

---

**Parallel Batching**

Tonge (2010)

**Batching**

Batching Partially Serial Batch (2009)

**Constraint**

Batching Parallel Batching

**Constraint?**

---

**Sweep & Prune**

GPU Sweep & Prune

**Batching?**

---

**GPU Compute**

---

**Heterogeneous**

---

**Advantages**

- Shared memory
- CPU & GPU can work together

---

**CUDA**

- AMD Fusion
  - Llano, Zacate
  - Advantages
  - Shared memory
  - CPU & GPU can work together

---

**SIGGRAPH 2011**
HETEROGENEOUS PARTICLE BASED SIMULATION

Takahiro Harada, Physics Simulation on Fusion Architecture, AMD Fusion Developer Summit(2011)
REFERENCES

- Takahiro Harada et al., Smoothed Particle Hydrodynamics on GPUs, Proc. of CGI(2007)
- Takahiro Harada, Real-time Rigid Body Simulation on GPUs, GPU Gems 3 (2007)
- Liu et al., Real-time Collision Culling of a Million Bodies on Graphics Processing Units, Siggraph Asia(2010)
- Takahiro Harada, Parallelizing the Physics Pipeline, GDC(2009)
- Richard Tonge, PhysX GPU Rigid Bodies in Batman, Game Programming Gems 8(2010)
- Rahul Sathe, Rigid Body Collision Detection on the GPU, Sig Poster(2006)
- Takahiro Harada, Physics Simulation on Fusion Architecture, AMD Fusion Developer Summint(2011)