OpenCL accelerated rigid body and collision detection

Erwin Coumans
Advanced Micro Devices
Overview

- Intro
- GPU broadphase acceleration structures
- GPU convex contact generation and reduction
- GPU BVH acceleration for concave shapes
- GPU constraint solver
Industry view

Hardware Platform

- PS3, Xbox 360, PC, iPhone, Android
- x86, PowerPC, Cell, ARM
- OpenCL, CUDA

Games Industry

- Sony Imageworks
- PDI Dreamworks
- Rockstar, Epic
- EA, Disney

Movie Industry

- ILM, Disney
- Stanford, UNC etc.

Game and Film physics Simulation

- Havok, PhysX
- Bullet, ODE, Newton, PhysBAM, Box2D
- Custom in-house physics engines

C++ APIs and Implementations

- C++ APIs and Implementations

Content Creation Tools

- Maya, 3ds Max, Houdini, LW, Cinema 4D
- Blender

Data representation

- FBX, COLLADA
- binary .hkx, .bullet format

GDC, SIGGRAPH

Conferences, Presentations

- Conferences, Presentations

Academia, Universities

- Academia, Universities

Robotics: Science and Systems Conference
2011
Our open source work

• Bullet Physics SDK, http://bulletphysics.org
• Sony Computer Entertainment Physics Effects
• OpenCL/DirectX11 GPU physics research
OpenCL™

• Open development platform for multi-vendor heterogeneous architectures

• The power of AMD Fusion: Leverages CPUs and GPUs for balanced system approach

• Broad industry support: Created by architects from AMD, Apple, IBM, Intel, NVIDIA, Sony, etc. AMD is the first company to provide a complete OpenCL solution

• Kernels written in subset of C99
Particle

State vector

\[ X = \begin{pmatrix} x \\ v \end{pmatrix} \]

\[ \dot{X} = \begin{pmatrix} v \\ F / m \end{pmatrix} \]

Rigid body

\[ X = \begin{pmatrix} x \\ q \\ v \\ \omega \end{pmatrix} \]

\[ \dot{X} = \begin{pmatrix} \frac{1}{2} \omega q \\ F / m \\ (\tau - \omega^\times I \omega) / I \end{pmatrix} \]
Rigid body simulation loop

First update the linear and angular velocity

\[ \mathbf{v}_{t+h} = \mathbf{v}_t + h \mathbf{F} \mathbf{m}^{-1} \]

\[ \mathbf{\omega}_{t+1} = \mathbf{\omega}_t + h \mathbf{\tau} \mathbf{I}^{-1} \]

Then the position

\[ \mathbf{x}_{t+h} = \mathbf{x}_t + h \mathbf{v}_{t+h} \]

\[ \mathbf{q}_{t+h} = \mathbf{q}_t + h \frac{1}{2} \mathbf{\omega}_{t+h} \mathbf{q}_t \]
__kernel void
interopKernel( const int startOffset, const int numNodes, __global float4 *g_vertexBuffer, __global float4 *linVel, __global float4 *pAngVel)
{
    int nodeID = get_global_id(0);
    float timeStep = 0.0166666;
    if( nodeID < numNodes )
    {
        g_vertexBuffer[nodeID + startOffset/4] += linVel[nodeID]*timeStep;
        float4 axis;
        float4 angvel = pAngVel[nodeID];
        float fAngle = native_sqrt(dot(angvel, angvel));
        axis = angvel * (native_sin(0.5f * fAngle * timeStep) / fAngle);
        float4 dorn = axis;
        dorn.w = native_cos(fAngle * timeStep * 0.5f);
        float4 orn0 = g_vertexBuffer[nodeID + startOffset/4+numNodes];
        float4 predictedOrn = quatMult(dorn, orn0);
        predictedOrn = quatNorm(predictedOrn);
        g_vertexBuffer[nodeID + startOffset/4+numNodes]=predictedOrn;
    }
}
OpenCL – OpenGL interop

- See [http://github.com/erwincooumans/experiments](http://github.com/erwincooumans/experiments)
Today’s execution model

• Single Program Multiple Data (SPMD)
• Same kernel runs on:
  • All compute units
  • All processing elements
• Purely “data parallel” mode
• PCIe bus between CPU and GPU is a bottleneck
Tomorrow’s execution model

- Multiple Program Multiple Data (SPMD)
- Nested data parallelism
  - Kernels can enqueue work
- CPU and GPU on one die
  - Unified memory
Physics Pipeline

Collision Data
- Collision shapes
- Object AABBs
- Overlapping pairs
- Contact points

Dynamics Data
- World transforms velocities
- Mass Inertia
- Constraints (contacts, joints)

Forward Dynamics Computation
- Apply gravity
- Predict transforms
- Compute AABBs
- Detect pairs
- Compute contact points

Collision Detection Computation
- Setup constraints
- Solve constraints
- Integrate position

Forward Dynamics Computation
- Compute contact points
- World transforms velocities
- Mass Inertia
- Constraints (contacts, joints)

AABB = axis aligned bounding box
Broadphase N-body problem

- Avoid brute-force N*N tests
- Input: world space BVs and unique IDs
- Output: array of potential overlapping pairs
Uniform grid

• Very GPU friendly, parallel radix sort
• Use modulo to make grid unbounded
Non uniform work granularity

- Small versus small
  - GPU
- Large versus small
  - CPU
- Large versus large
  - CPU
Incremental sweep and prune

- Update 3 sorted axis and overlapping pairs

- Doesn’t parallelize easy: data dependencies
Parallel 1 axis sweep and prune

• From scratch sort 1 axis sweep to find all pairs

• Parallel radix sort and parallel sweep

[Game Physics Pearls, 2010, AK Peters]
Dynamic BVH tree broadphase

- Keep two dynamic trees, one for moving objects, other for objects (sleeping/static)
- Find neighbor pairs:
  - Overlap M versus M and Overlap M versus S
Update/move a leaf node

• If new AABB is contained by old do nothing
• Otherwise remove and re-insert leaf
  – Re-insert at closest ancestor that was not resized during remove
• Expand AABB with margin
  – Avoid updates due to jitter or small random motion
• Expand AABB with velocity
  – Handle the case of linear motion over n frames
Parallel BVH tree traversal

- Incremental update on CPU (shared memory)
- Use parallel history traversal on GPU
Contact generation

• Input: overlapping pairs, collision shapes
• Output: contact points
Closest point algorithms

• Algebraic closed forms, ie. sphere-sphere
• Separating axis theorem for convex polyhedra
  • Only computes overlap, no positive distances
• Gilbert Johnson Kheerthi (GJK) for general convex objects
  • Doesn’t compute overlap, needs a companion algorithm for penetrating case, for example the Expanding Polytope Algorithm
Separating axis test

- Test all axis in parallel, use cube map
GJK or EPA
Sutherland–Hodgman clipping

- Clip incident face against reference face side planes (but not the reference face).
- Consider clip points with positive penetration.
Cube map

- GPU friendly convex versus convex

- See “Rigid body collision detection on the GPU” by Rahul Sathe et al, SIGGRAPH 2006 poster
Contact reduction

• Keep only 4 points
Concave shapes

- Hierarchical approximate convex decomposition (ICIP 2009 proceedings. Khaled Mamou)

- http://sourceforge.net/projects/hacd
Parallelizing Constraint Solver

• Projected Gauss Seidel iterations are not embarrassingly parallel
Reordering constraint batches

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td></td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Robotics: Science and Systems Conference 2011
Thanks!

• For more information: [http://bulletphysics.org](http://bulletphysics.org)
• erwin.coumans@amd.com