OpenCL Game Physics

Bullet: A Case Study in Optimizing Physics Middleware for the GPU

Erwin Coumans

Overview

- Introduction
- Particle Physics Pipeline from the NVIDIA SDK
  - Uniform grid, radix or bitonic sort, prefix scan, Jacobi
- Rigid Body Physics Pipeline
  - Parallel Neighbor search using dynamic BVH trees
  - Neighboring Pair Management
  - Convex Collision Detection: GJK in OpenCL on GPU
  - Concave Collision Detection using BVHs
  - Parallel Constraint Solving using PGS
- OpenCL cross-platform and debugging

Introduction

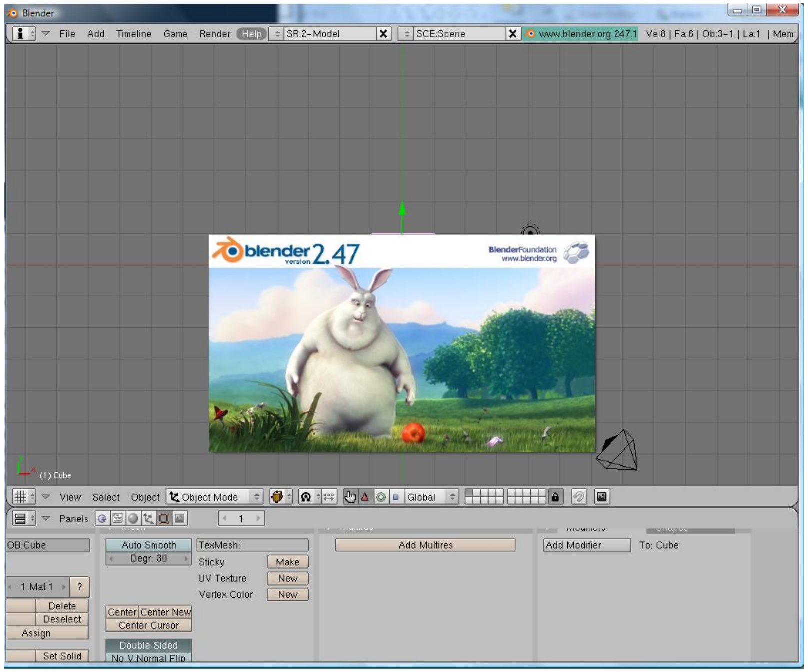
- Bullet is an open source Physics SDK used by game developers and movie studios
- PC, Mac, iPhone, Wii, Xbox 360, PlayStation 3
- Bullet 3.x will support OpenCL acceleration
  - Simplified rigid body pipeline fully running on GPU
  - Developer can mix stages between CPU and GPU
- Implementation is available, links at the end

Some games using Bullet Physics

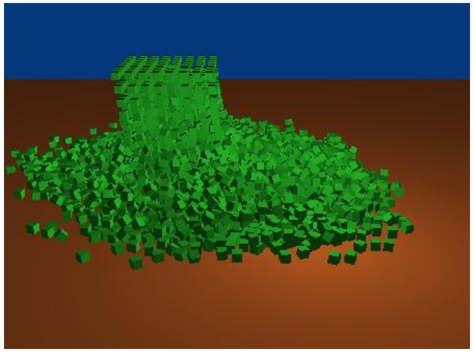
Some movies using Bullet Physics

Authoring tools

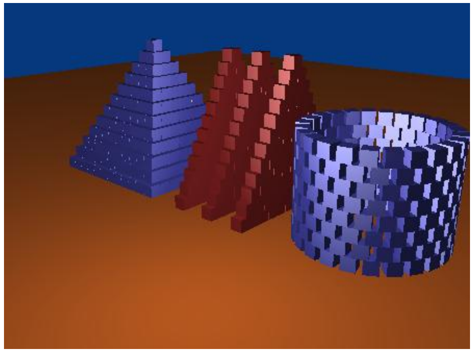
- Maya Dynamica Plugin
- Cinema 4D 11.5
- Blender



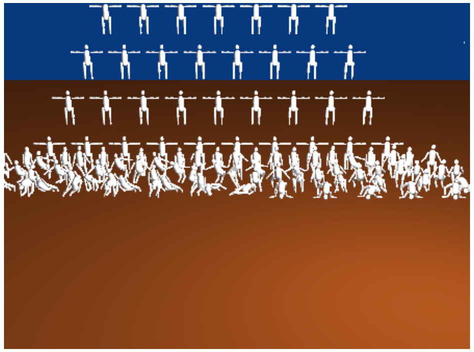
Rigid Body Scenarios



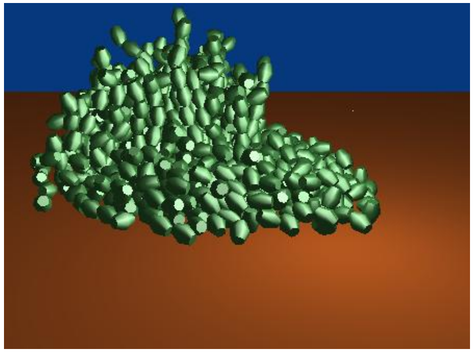
3000 falling boxes



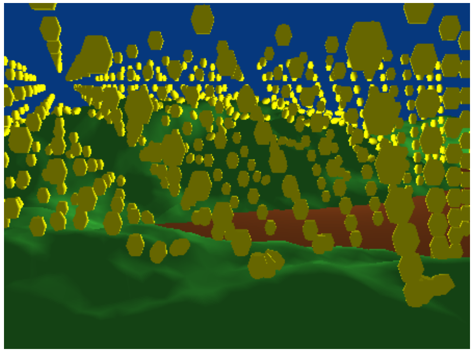
1000 stacked boxes



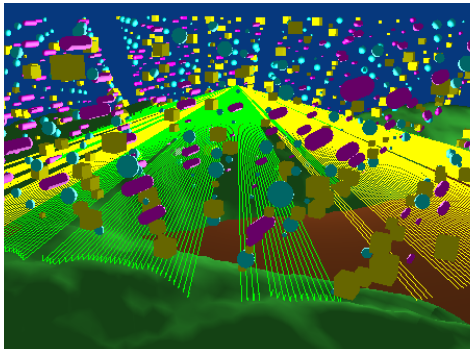
136 ragdolls



1000 convex hulls

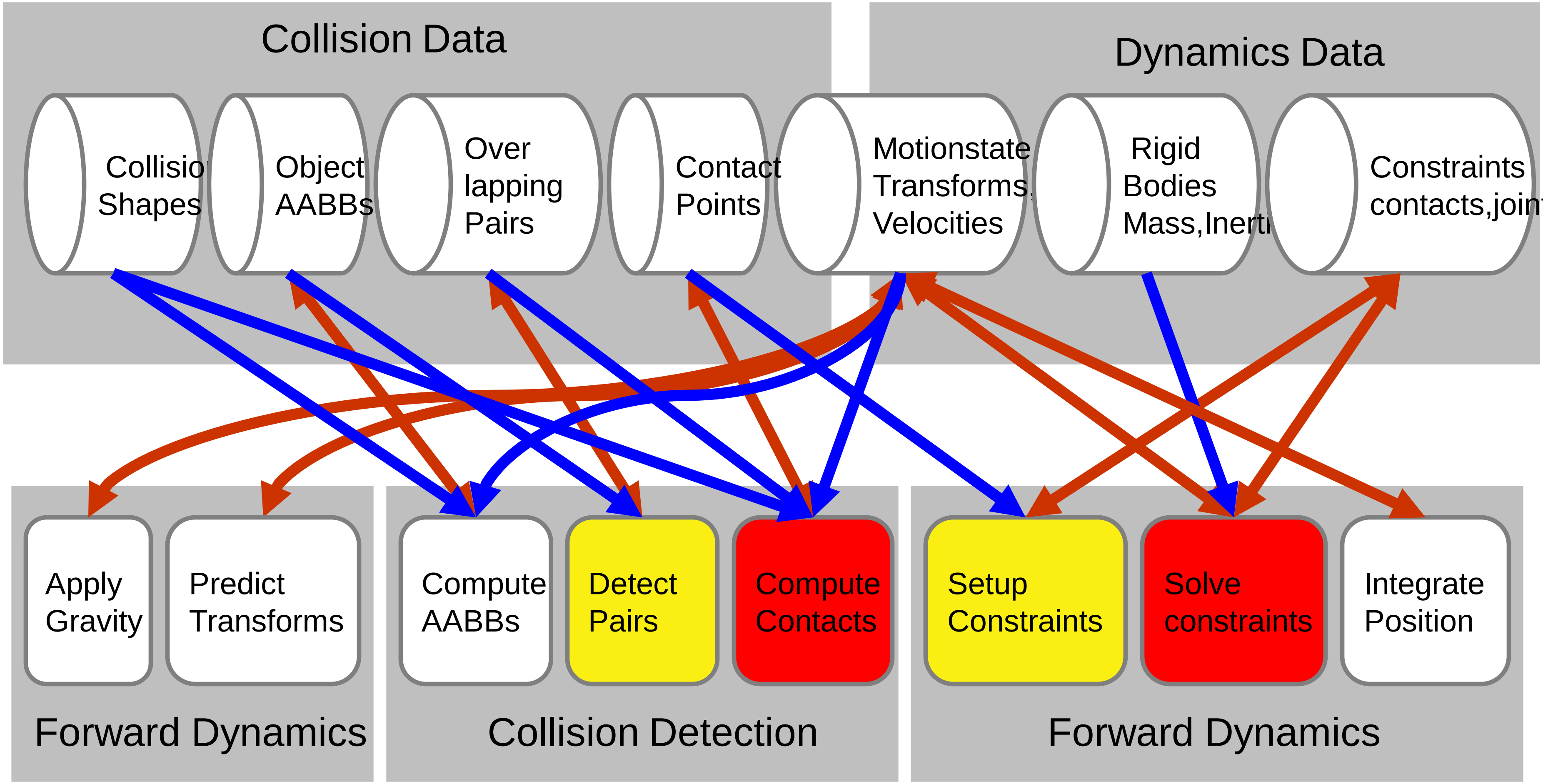


1000 convex against trimesh



ray casts against 1000 primitives and trimesh

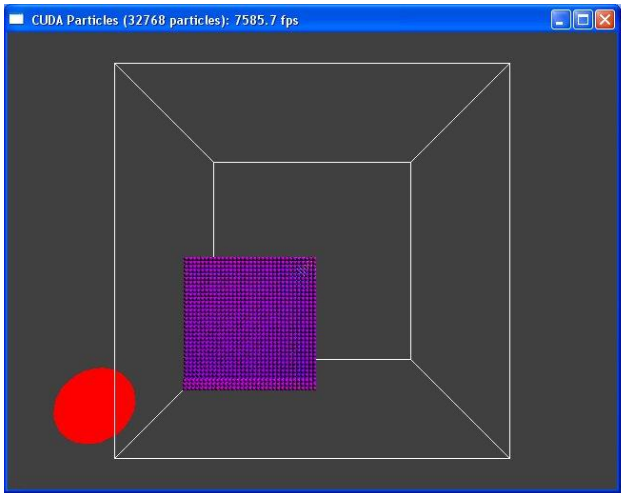
Performance bottlenecks

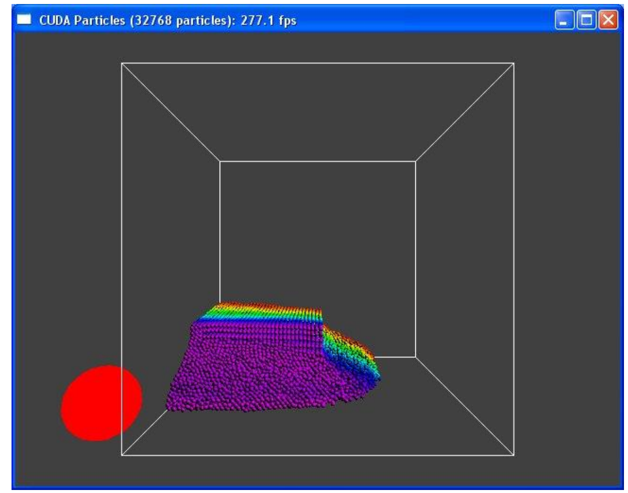


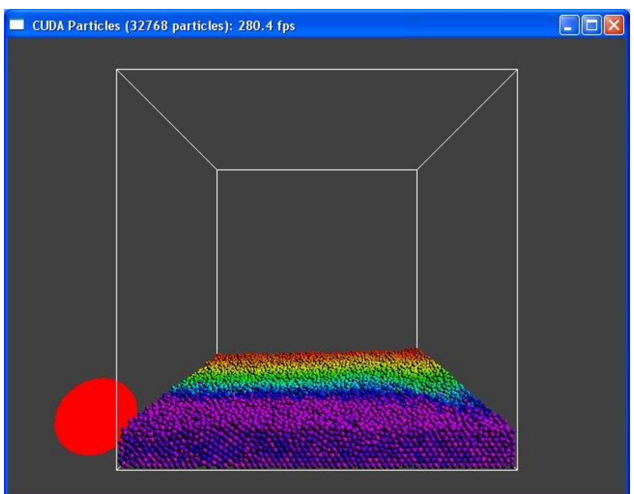
Leveraging the NVidia SDK

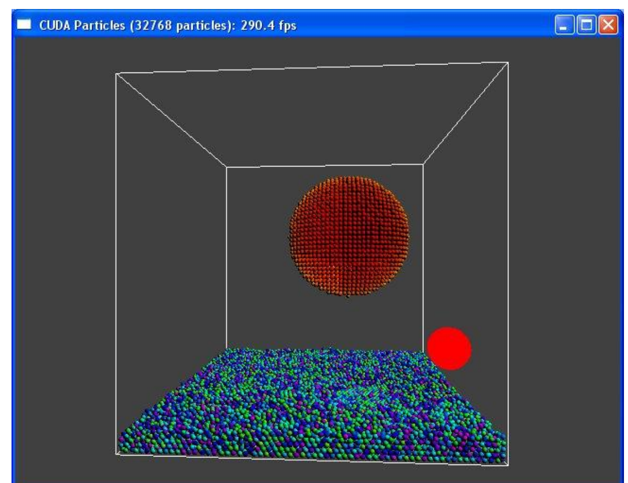
- Radix sort, bitonic sort
- Prefix scan, compaction
- Examples how to use fast shared memory
- Uniform Grid example in Particle Demo

Particle Physics
CUDA and OpenCL Demo

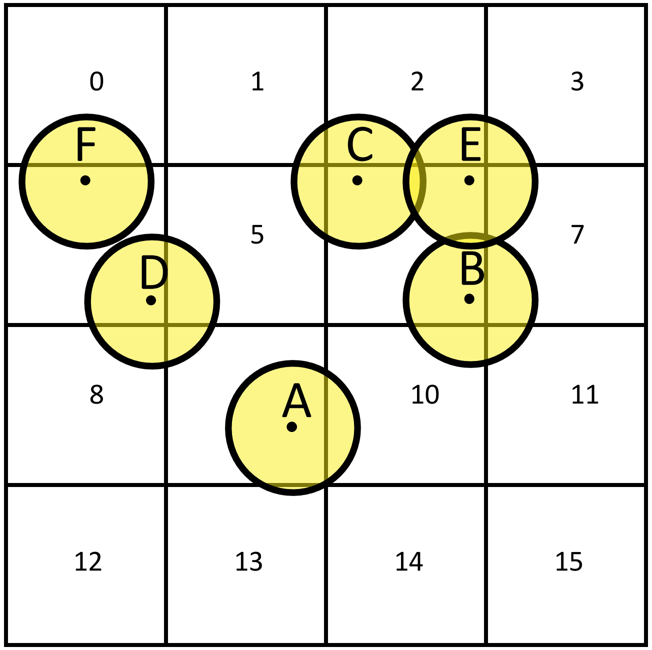






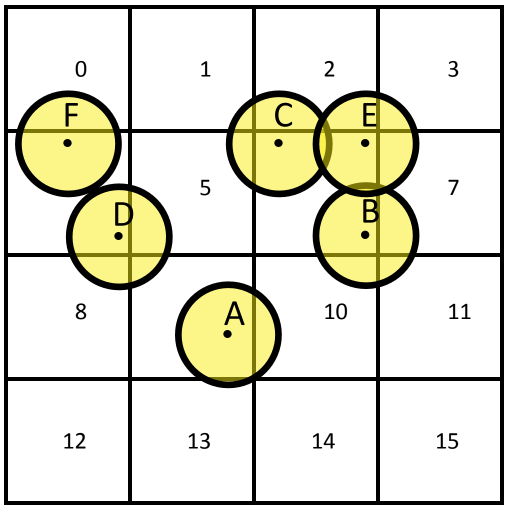


Uniform Grid



Cell ID	Count	Particle ID
0	0	
1	0	
2	0	
3	0	
4	2	D,F
5	0	
6	3	B,C,E
7	0	
8	0	
9	1	A
10	0	
11	0	
12	0	
13	0	
14	0	
15	0	

Sorting Particles per Cell



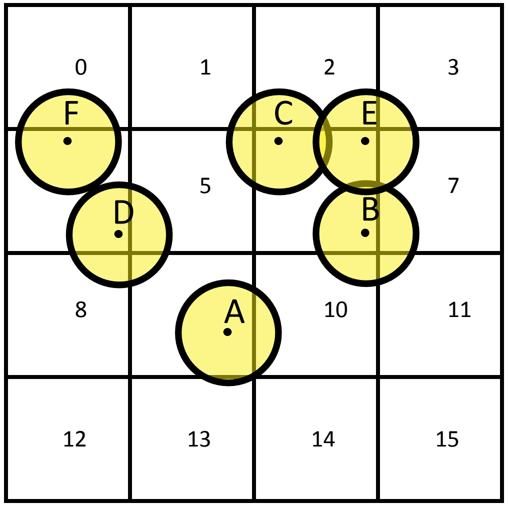
Cell Index	Cell Start
0	
1	
2	
3	
4	0
5	
6	2
7	
8	
9	5
10	
11	
12	
13	
14	
15	

Array Index	Unsorted Cell ID, Particle ID	Sorted Cell ID Particle ID
0	9, A	4,D
1	6,B	4,F
2	6,C	6,B
3	4,D	6,C
4	6,E	6,E
5	4,F	9,A

Neighbor search

- Calculate grid index of particle center
- Parallel Radix or Bitonic Sorted Hash Array
- Search 27 neighboring cells
  - Can be reduced to 14 because of symmetry
- Interaction happens during search
  - No need to store neighbor information
- Jacobi iteration: independent interactions

Interacting Particle Pairs



Array Index	Sorted Cell ID Particle ID
0	4,D
1	4,F
2	6,B
3	6,C
4	6,E
5	9,A

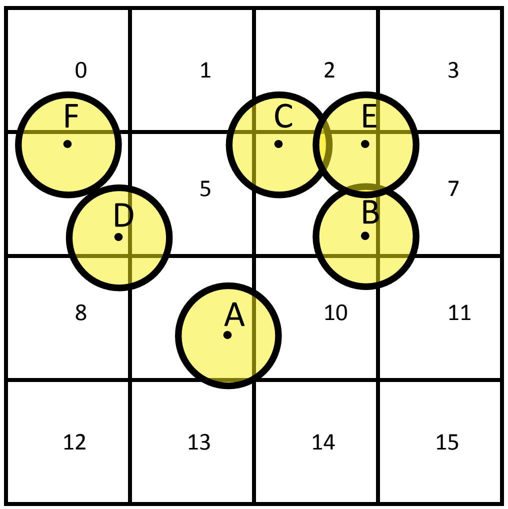
Interacting Particle Pairs
D,F
B,C
B,E
C,E
A,D
A,F
A,B
A,C
A,E

Using the GPU Uniform Grid as part of the Bullet CPU pipeline

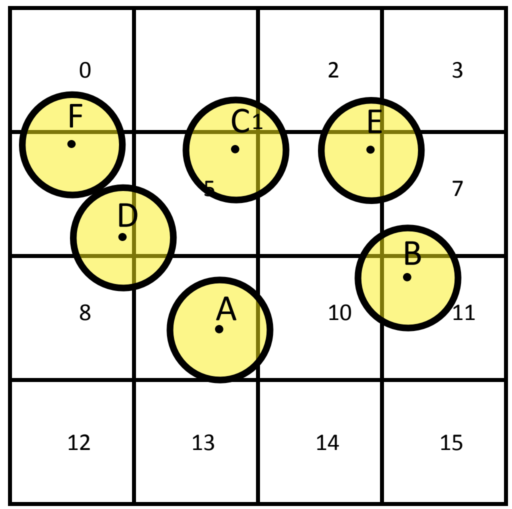
- Available through btCudaBroadphase
- Reduce bandwidth and avoid sending all pairs
- Bullet requires persistent contact pairs
  - to store cached solver information (warm-starting)
- Pre-allocate pairs for each object

Persistent Pairs

Before



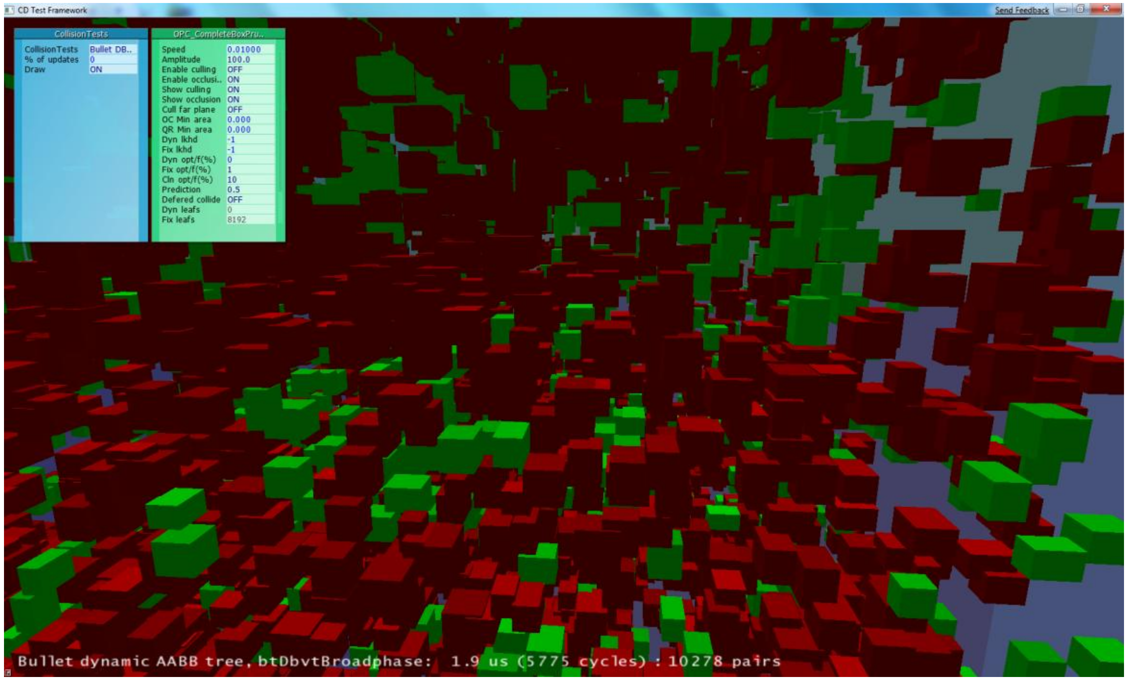
After



Particle Pairs Before	After	Differences
D,F	D,F	A,B removed
B,C	B,C	B,C removed
B,E	B,E	C,F added
C,E	C,E	C,D added
A,D	A,D	
A,F	A,F	
A,B	A,C	
A,C	A,E	
A,E	C,F	
	C,D	

Broadphase benchmark

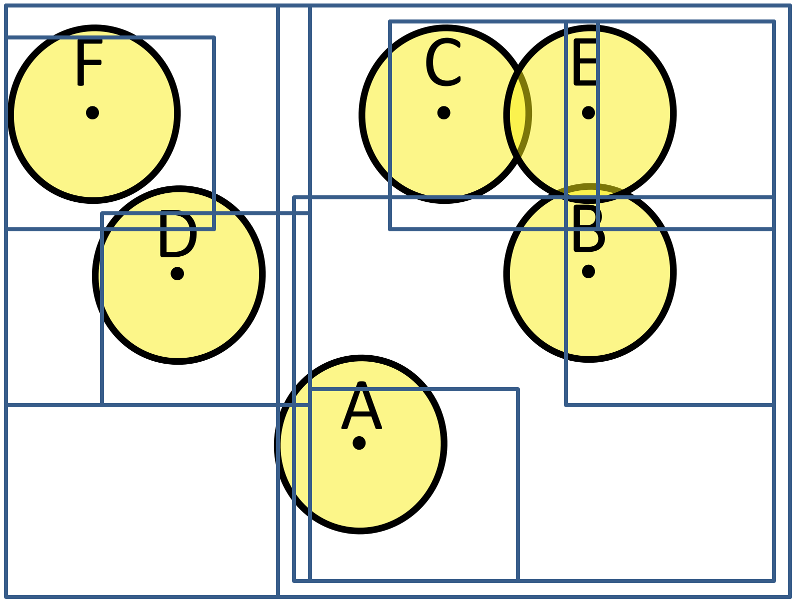
- Includes btCudaBroadphase
- Bullet SDK: Bullet/Extras/CDTestFramework

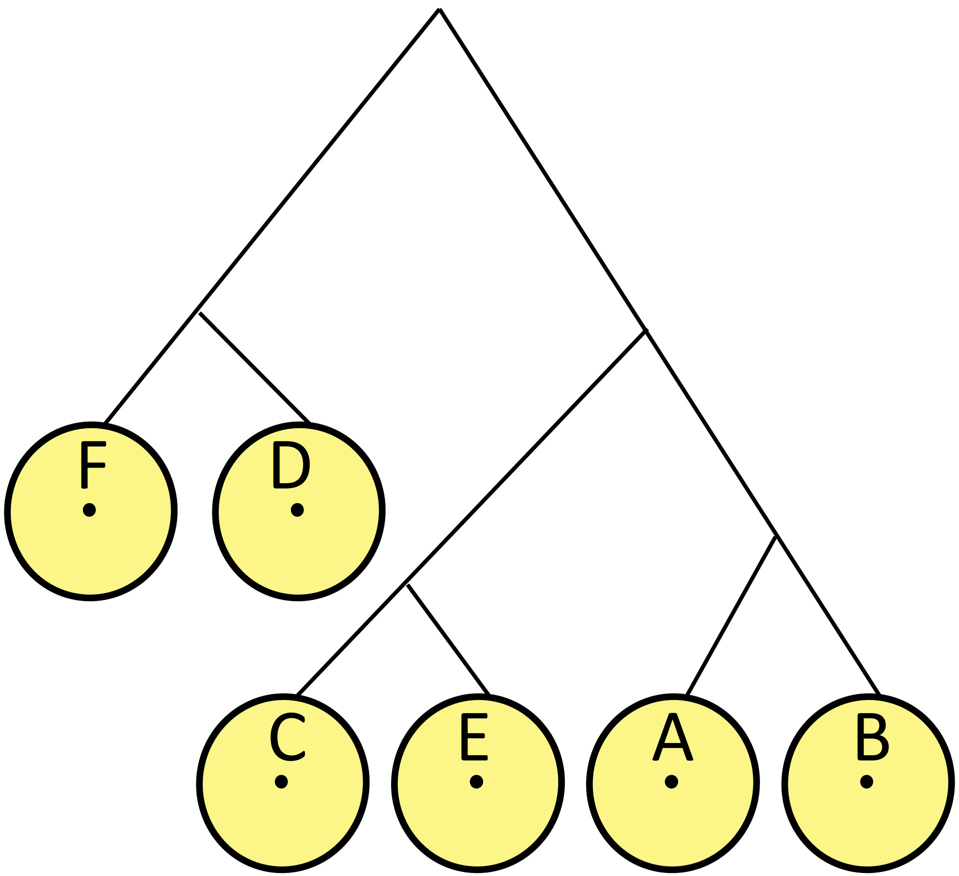


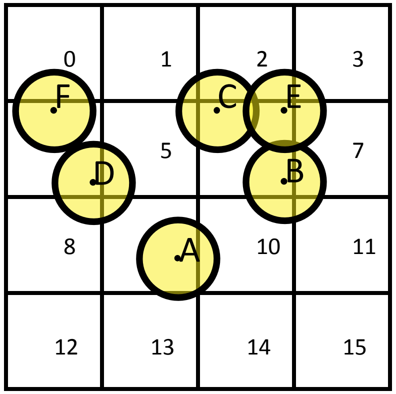
From Particles to Rigid Bodies

	Particles	Rigid Bodies
World Transform	Position	Position and Orientation
Neighbor Search	Uniform Grid	Dynamic BVH tree
Compute Contacts	Sphere-Sphere	Generic Convex Closest Points, GJK
Static Geometry	Planes	Concave Triangle Mesh
Solving method	Jacobi	Projected Gauss Seidel

Dynamic BVH Trees





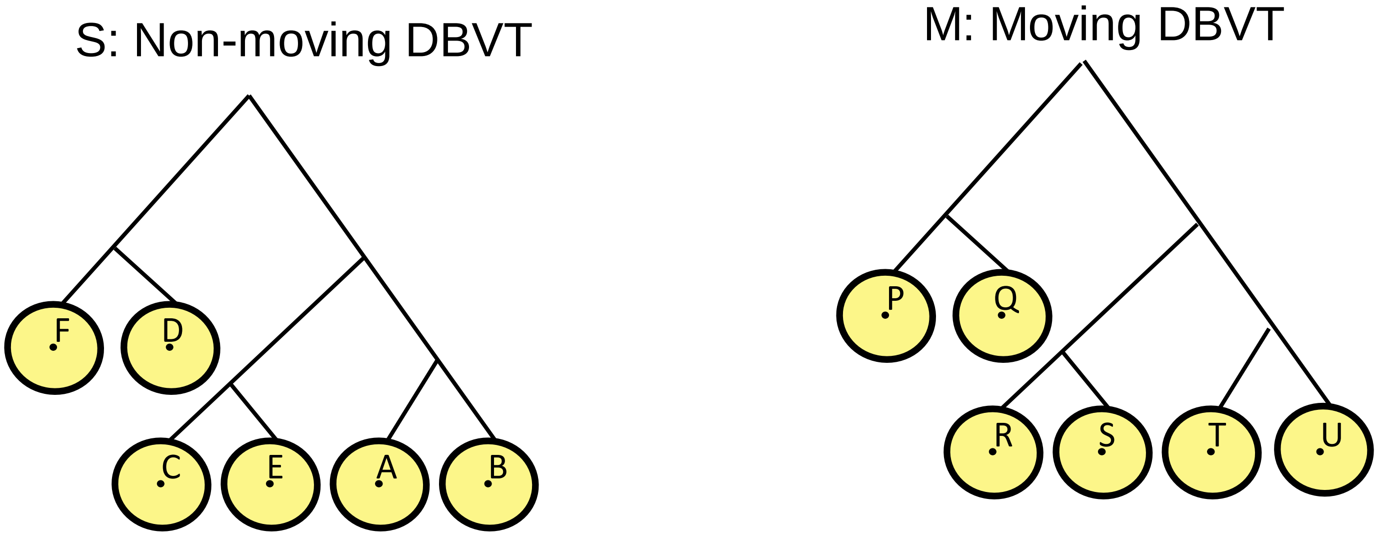


Dynamic BVH tree acceleration structure

- Broadphase n-body neighbor search
- Ray and convex sweep test
- Concave triangle meshes
- Compound collision shapes

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

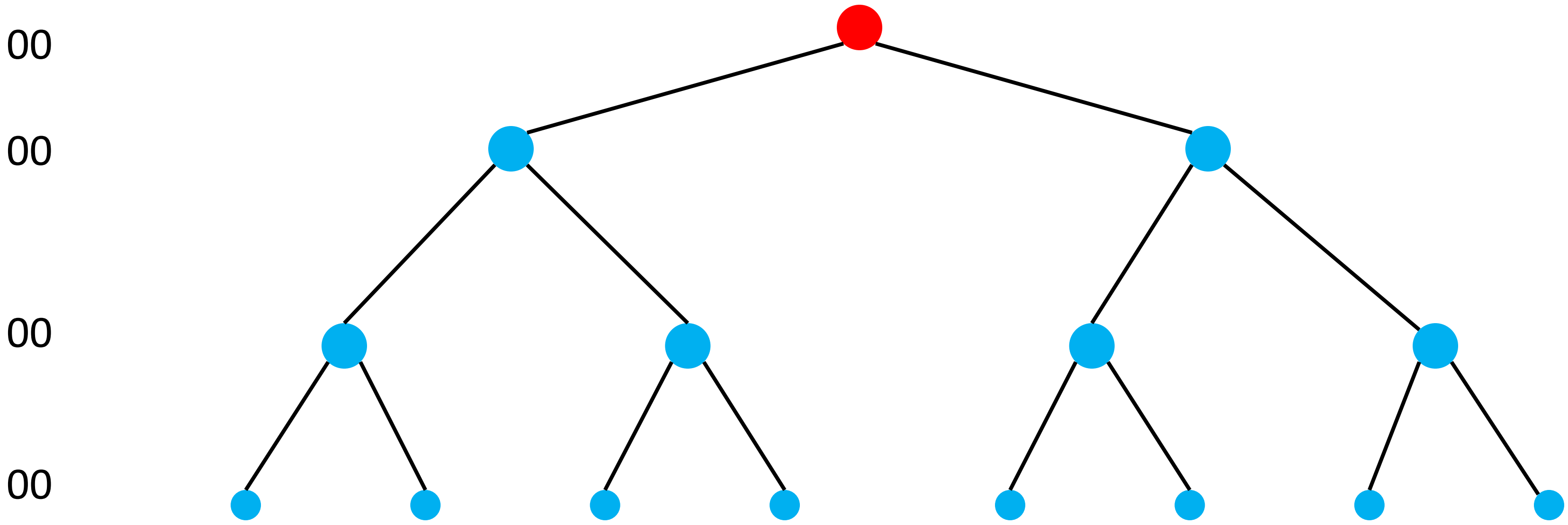


DBVT Broadphase Optimizations

- Objects can move from one tree to the other
- Incrementally update, re-balance tree
- Tree update hard to parallelize
- Tree traversal can be parallelized on GPU
  - Idea proposed by Takahiro Harada at GDC 2009

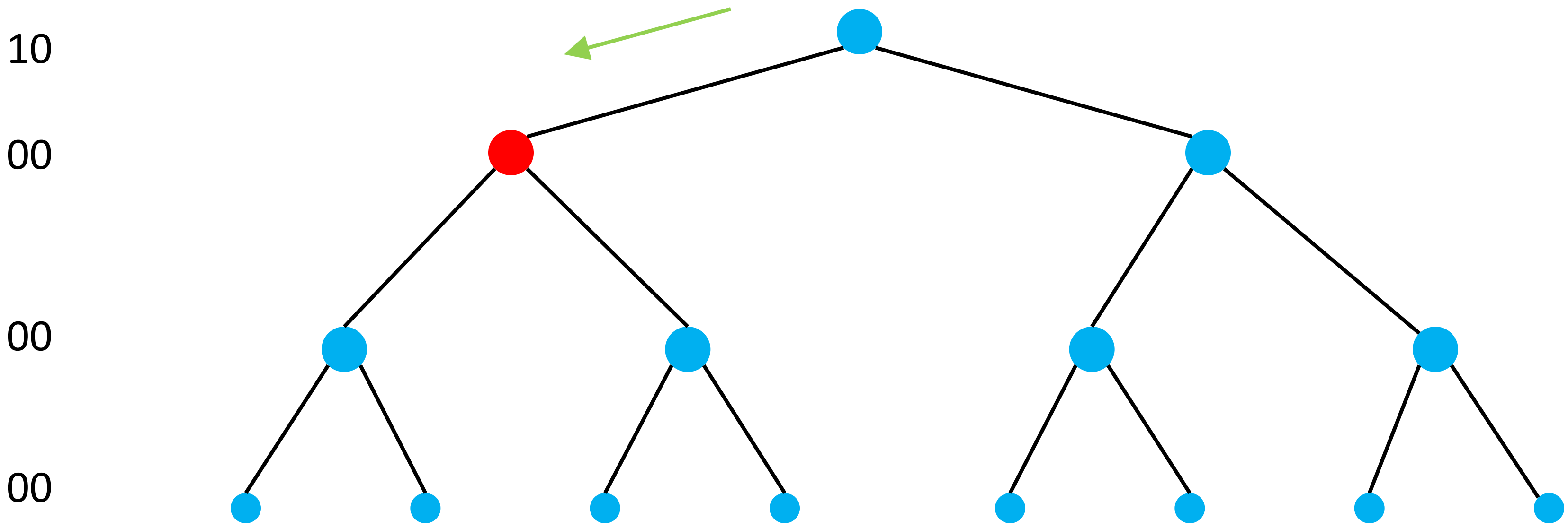
Parallel GPU Tree Traversal using History Flags

- Alternative to recursive or stackless traversal



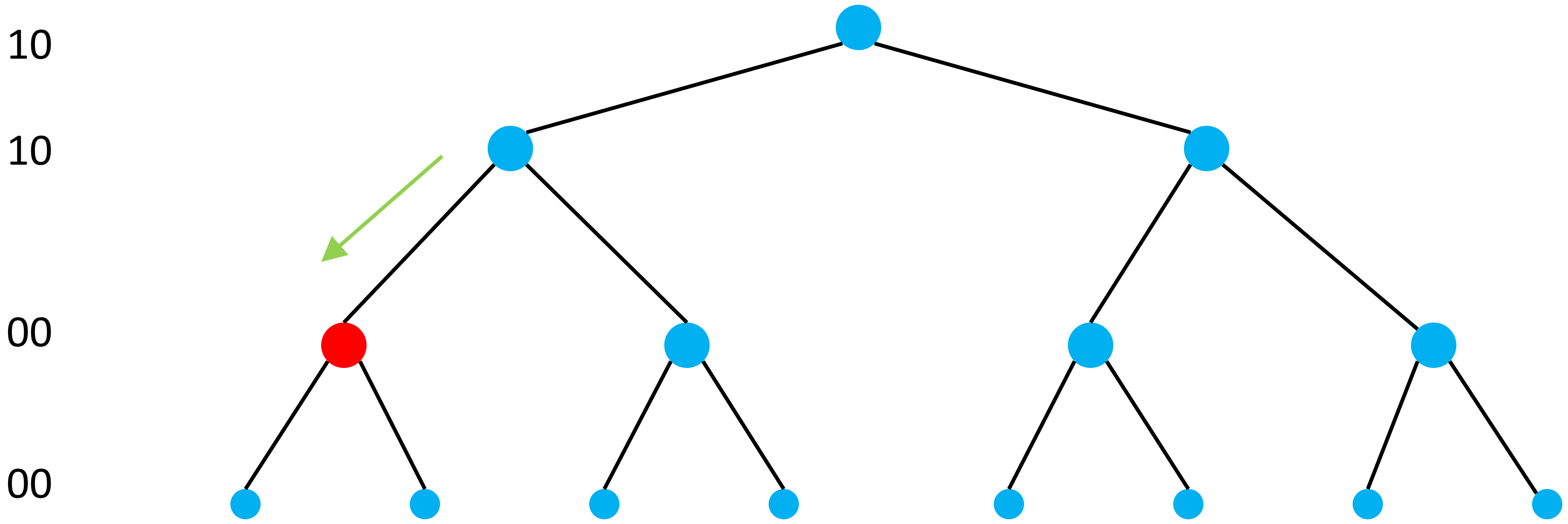
Parallel GPU Tree Traversal using History Flags

- 2 bits at each level indicating visited children



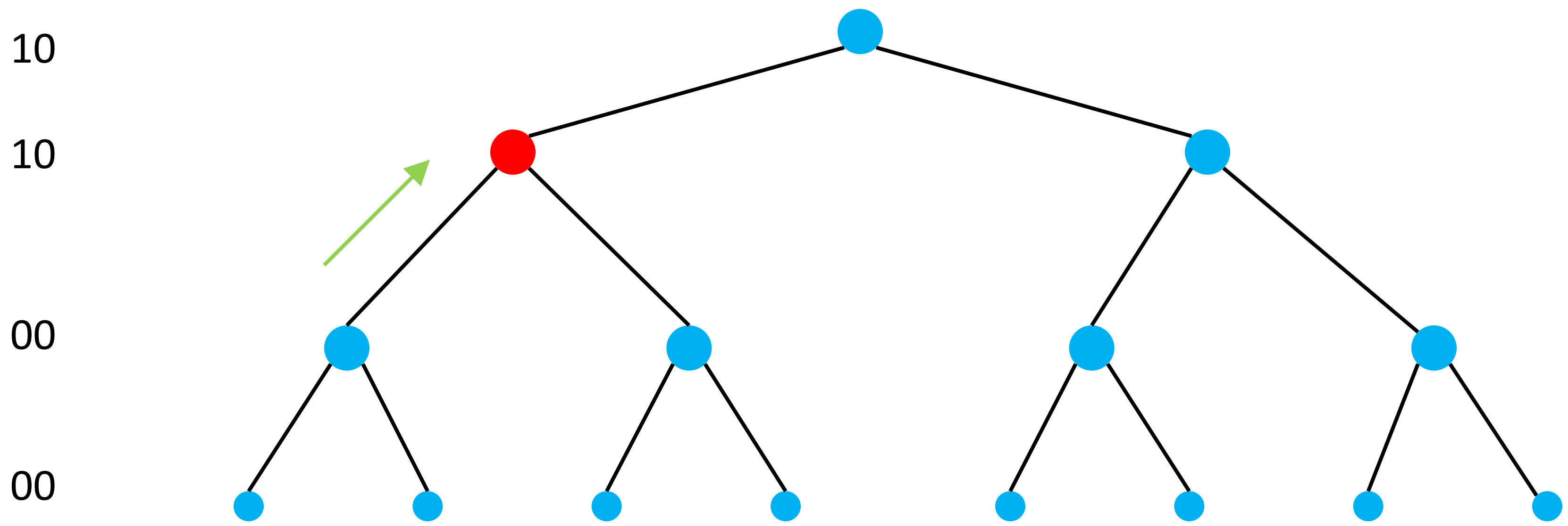
Parallel GPU Tree Traversal using History Flags

- Set bit when descending into a child branch



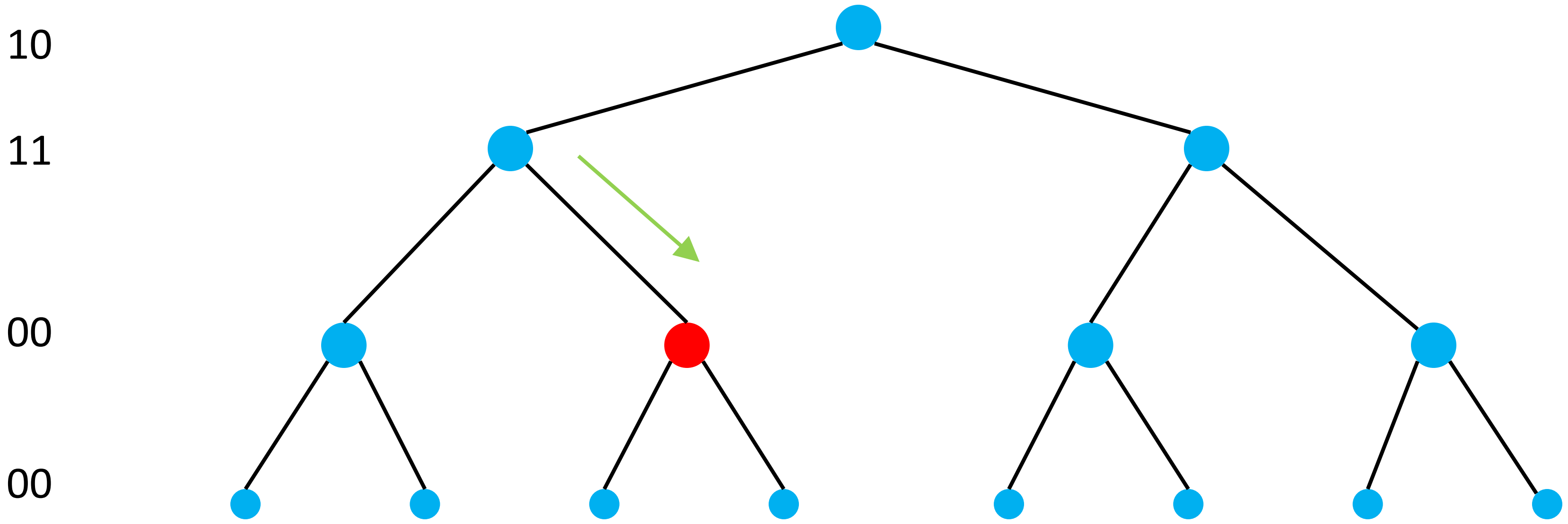
Parallel GPU Tree Traversal using History Flags

- Reset bits when ascending up the tree



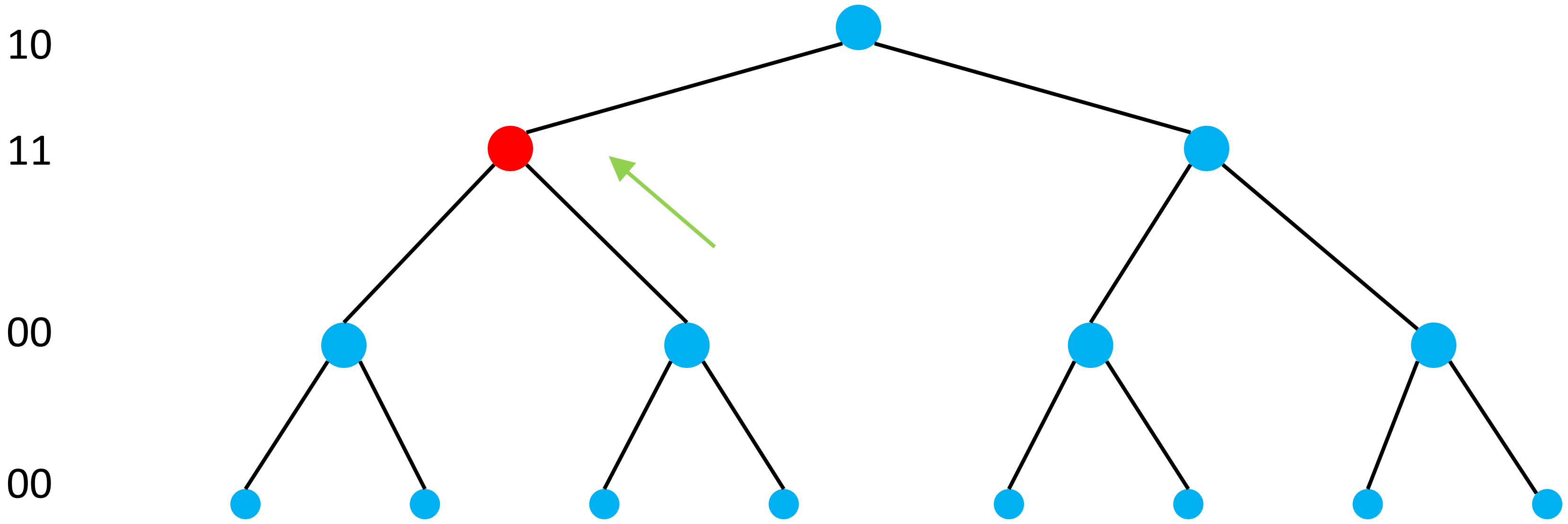
Parallel GPU Tree Traversal using History Flags

- Requires only twice the tree depth bits



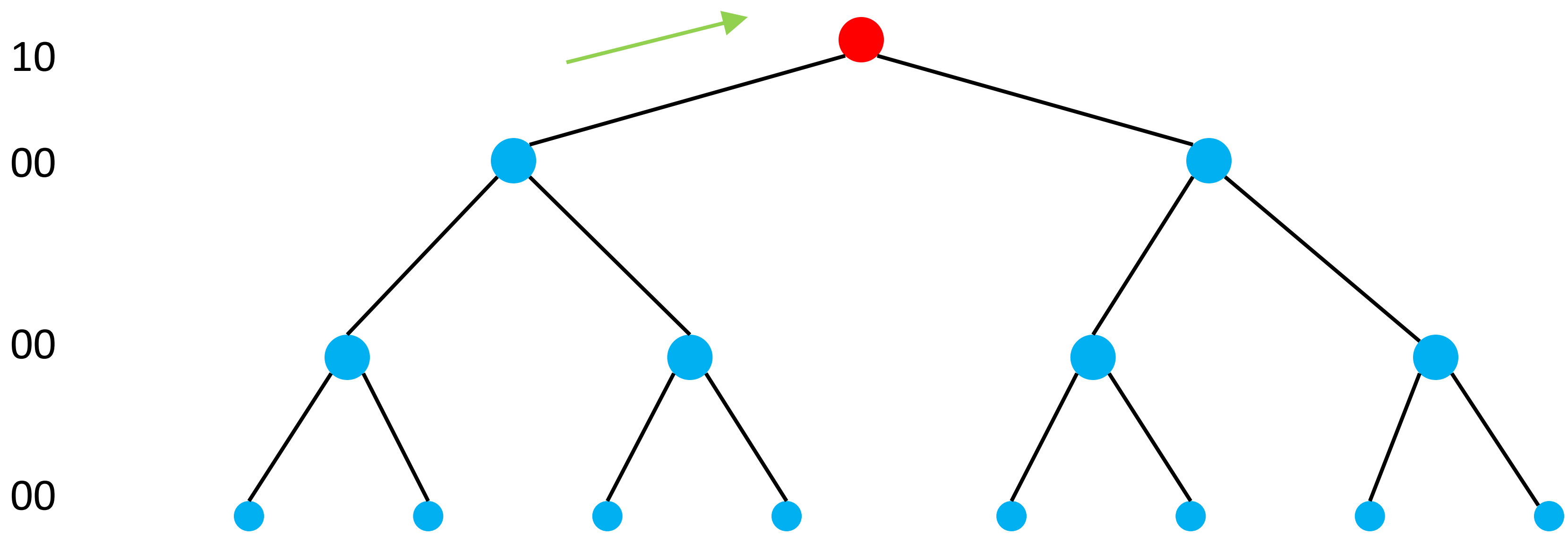
Parallel GPU Tree Traversal using History Flags

- When both bits are set, ascend to parent



Parallel GPU Tree Traversal using History Flags

- When both bits are set, ascend to parent

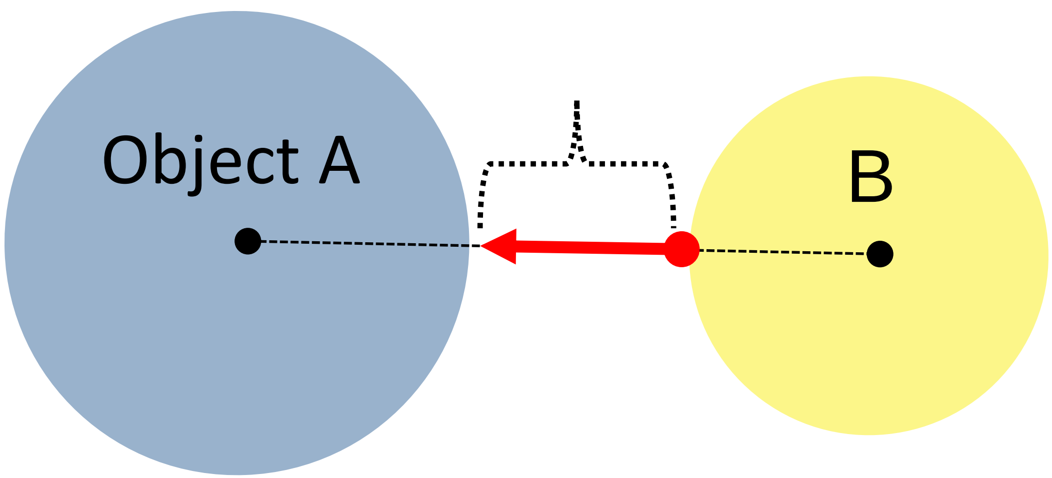


History tree traversal

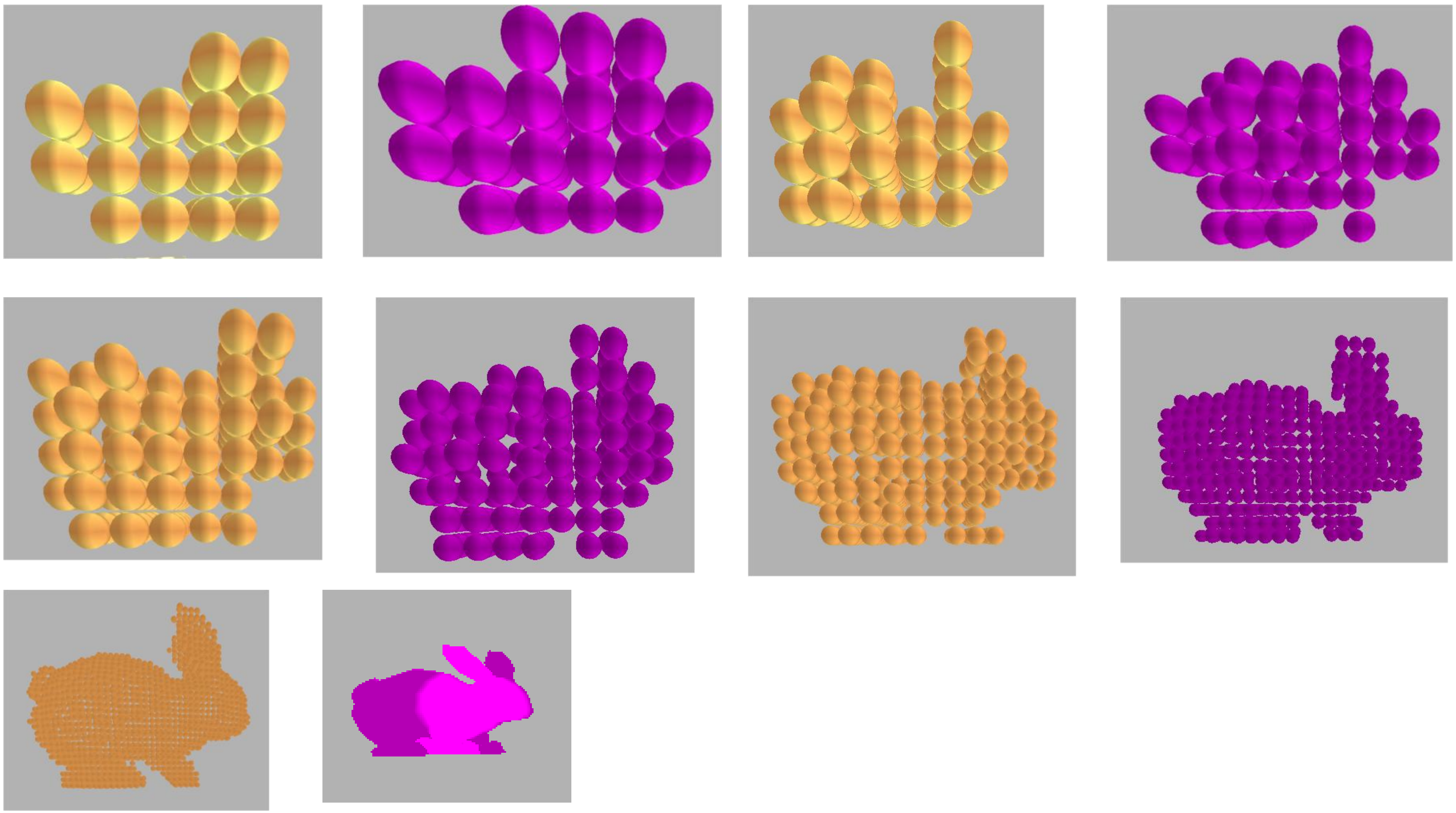
```
do{
        if(Intersect(n->volume,volume)){
            if(n->isinternal()) {
                if (!historyFlags[curDepth].m_visitedLeftChild){
                historyFlags[curDepth].m_visitedLeftChild = 1;
                n = n->childs[0];
                curDepth++;
                continue;}
            if (!historyFlags[curDepth].m_visitedRightChild){
                historyFlags[curDepth].m_visitedRightChild = 1;
                n = n->childs[1];
                curDepth++;
                continue;}
            }
            else
                policy.Process(n);
            }
        n = n->parent;
        historyFlags[curDepth].m_visitedLeftChild = 0;
        historyFlags[curDepth].m_visitedRightChild = 0;
        curDepth--;
} while (curDepth);
```

Find contact points

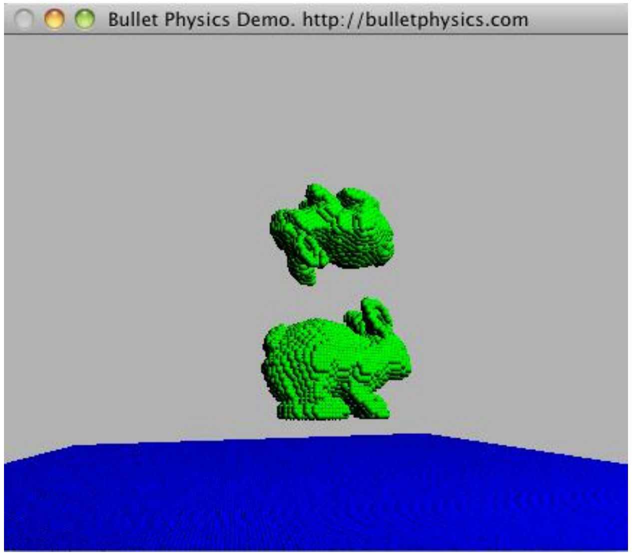
- Closest points, normal and distance
- Convention: positive distance -> separation
- Contact normal points from B to A

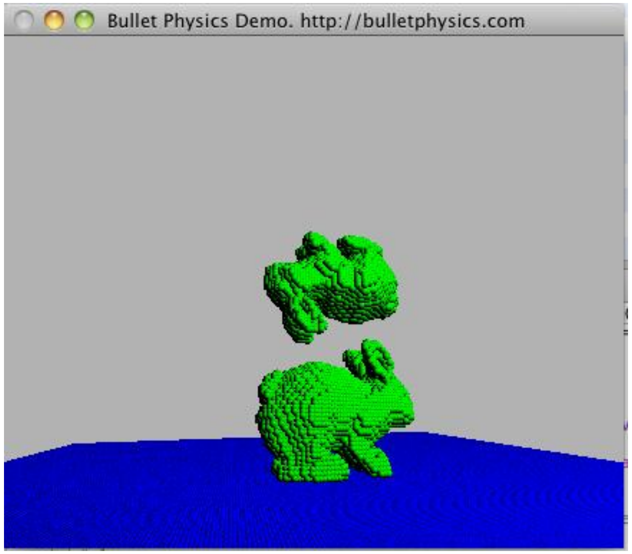


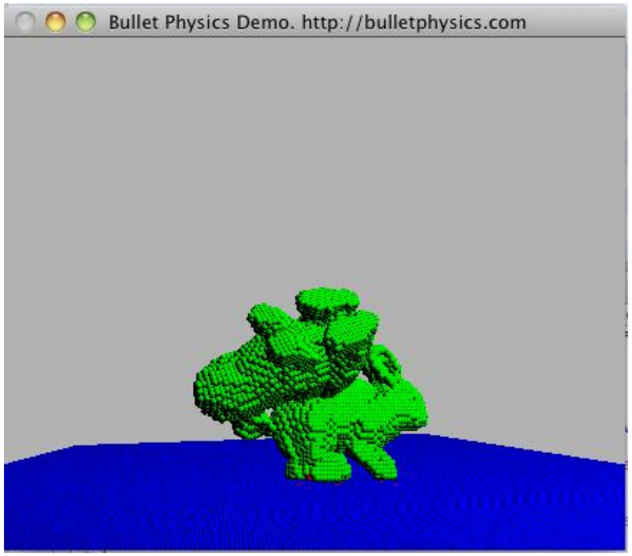
Voxelizing objects

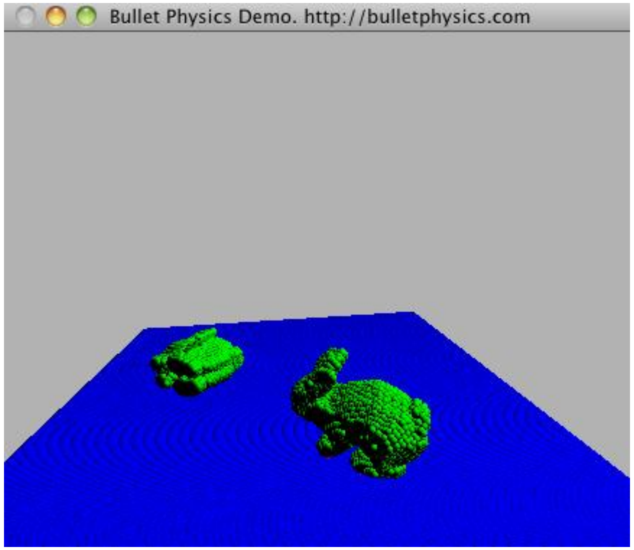


OpenCL Rigid Particle Bunnies



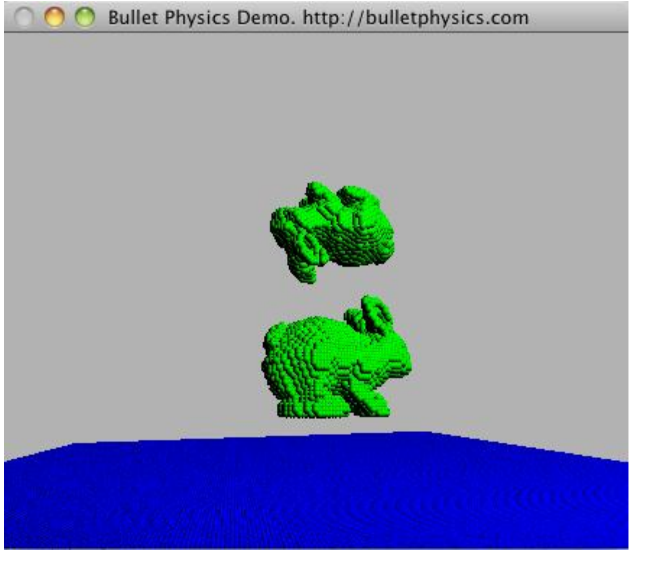




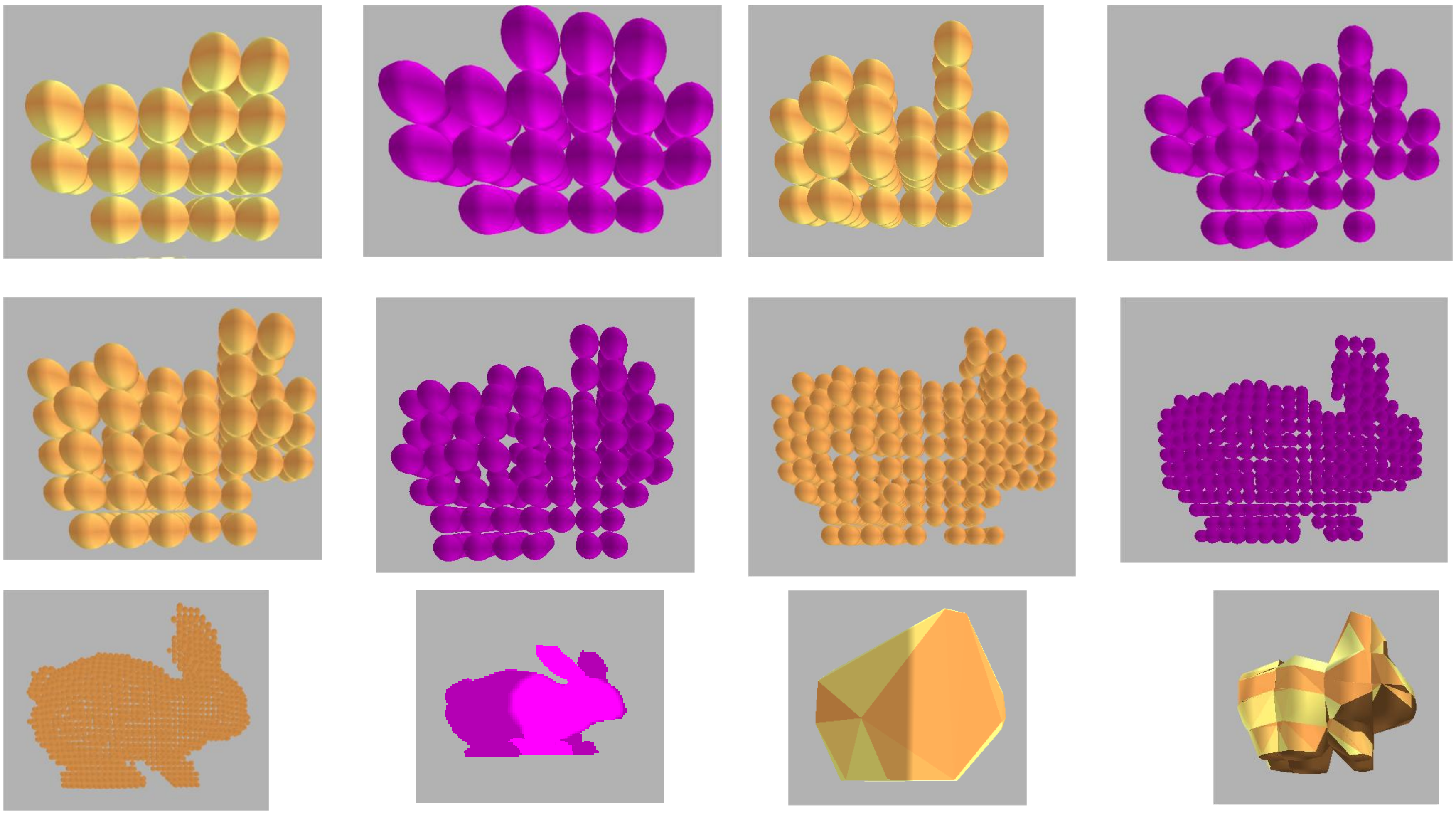


Broadphase

- The bunny demo broadphase has entries for each particle to avoid n^2 tests
- Many sphere-sphere contact pairs between two rigid bunnies
- Uniform Grid is not sufficient

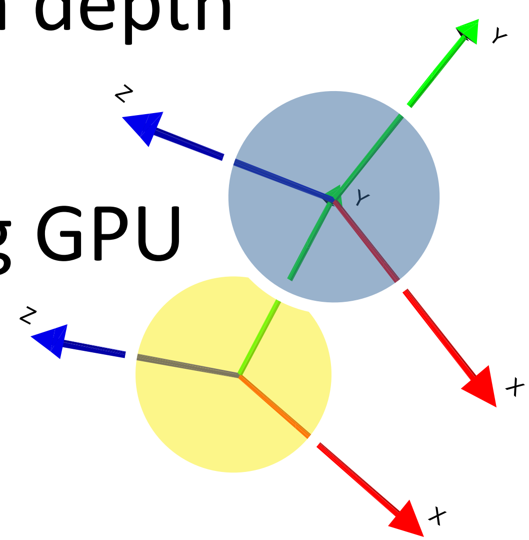


Voxelizing objects



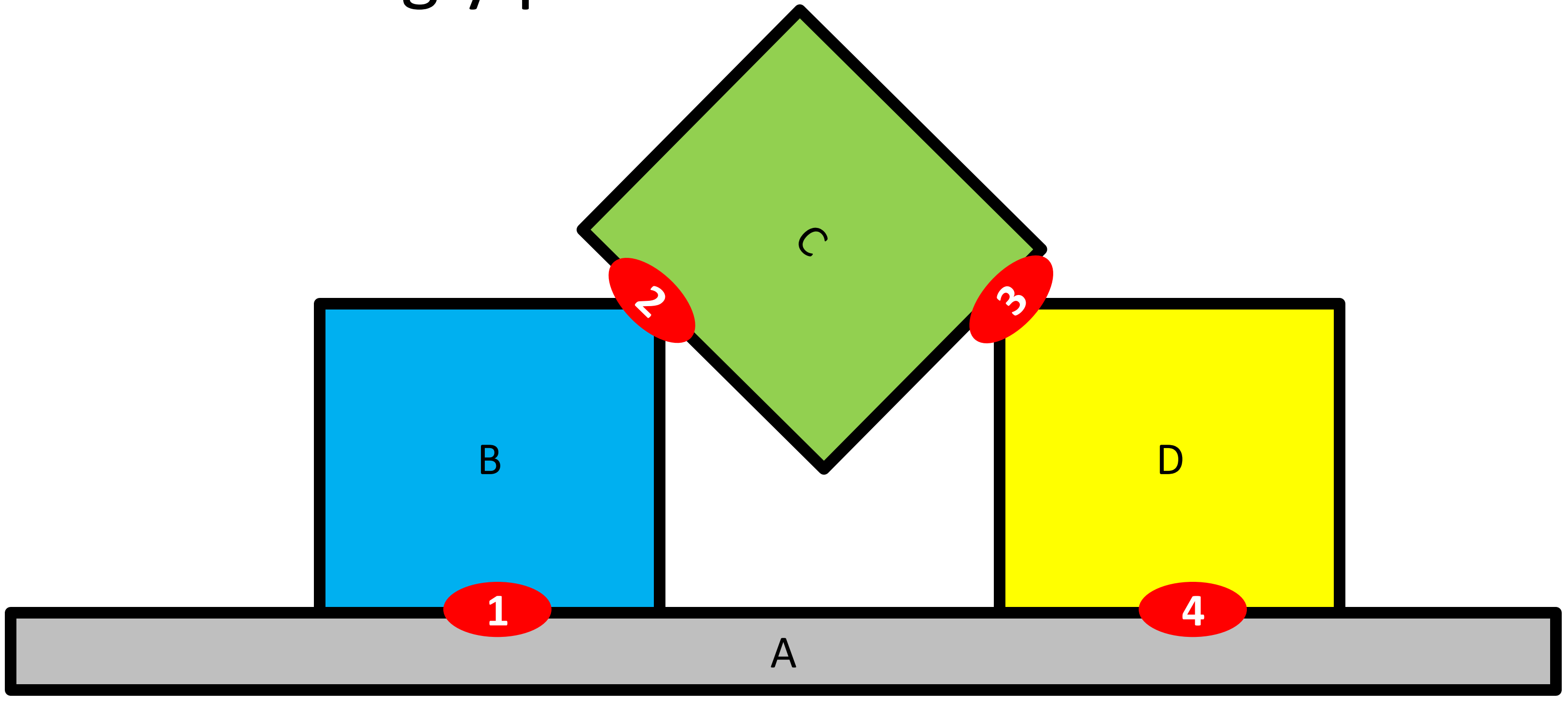
General convex collision detection on GPU

- Bullet uses hybrid GJK algorithm with EPA
- GJK convex collision detection fits current GPU
- EPA penetration depth harder to port to GPU
  - Larger code size, dynamic data structures
- Instead of EPA, sample penetration depth
  - Using support mapping
- Support map can be sampled using GPU hardware



Parallelizing Constraint Solver

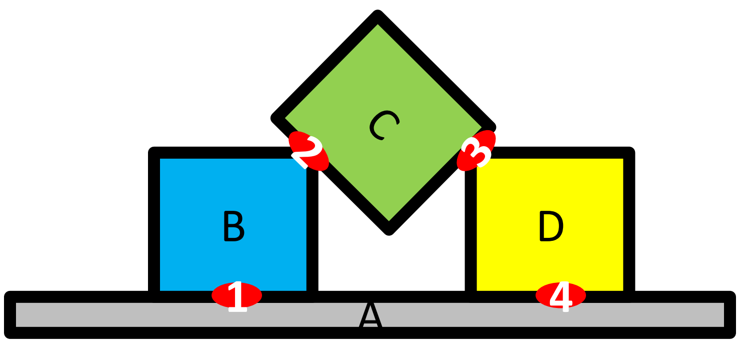
- Projected Gauss Seidel iterations are not embarrassingly parallel



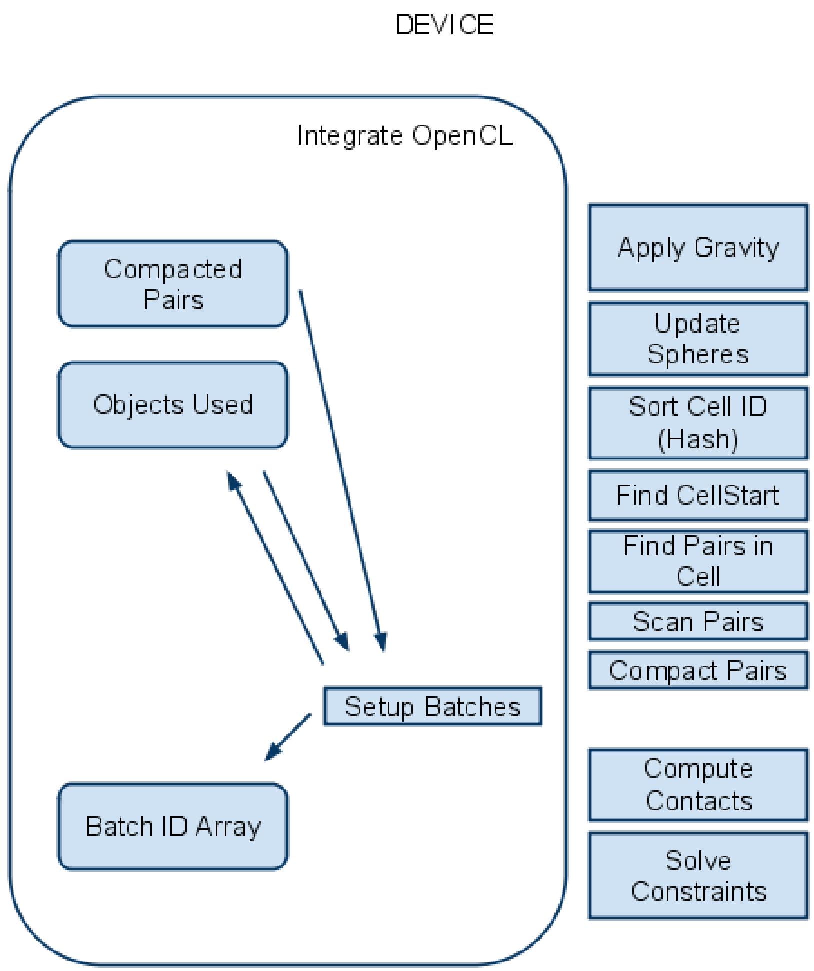
Reordering constraint batches

A	B	C	D
1	1		
	2	2	
		3	3
4			4

A	B	C	D
1	1	3	3
4	2	2	4



Creating Parallel Batches

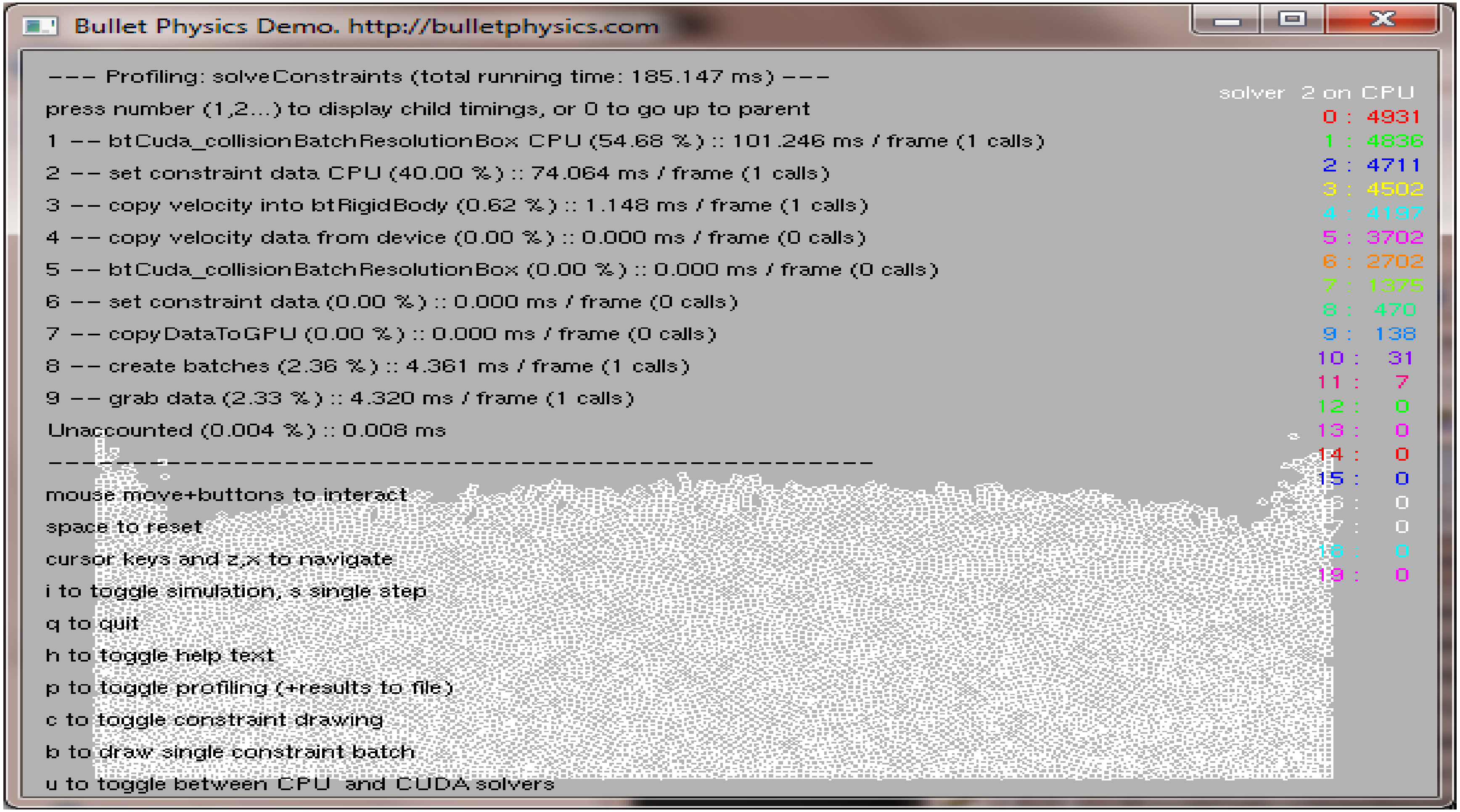


OpenCL kernel Setup Batches

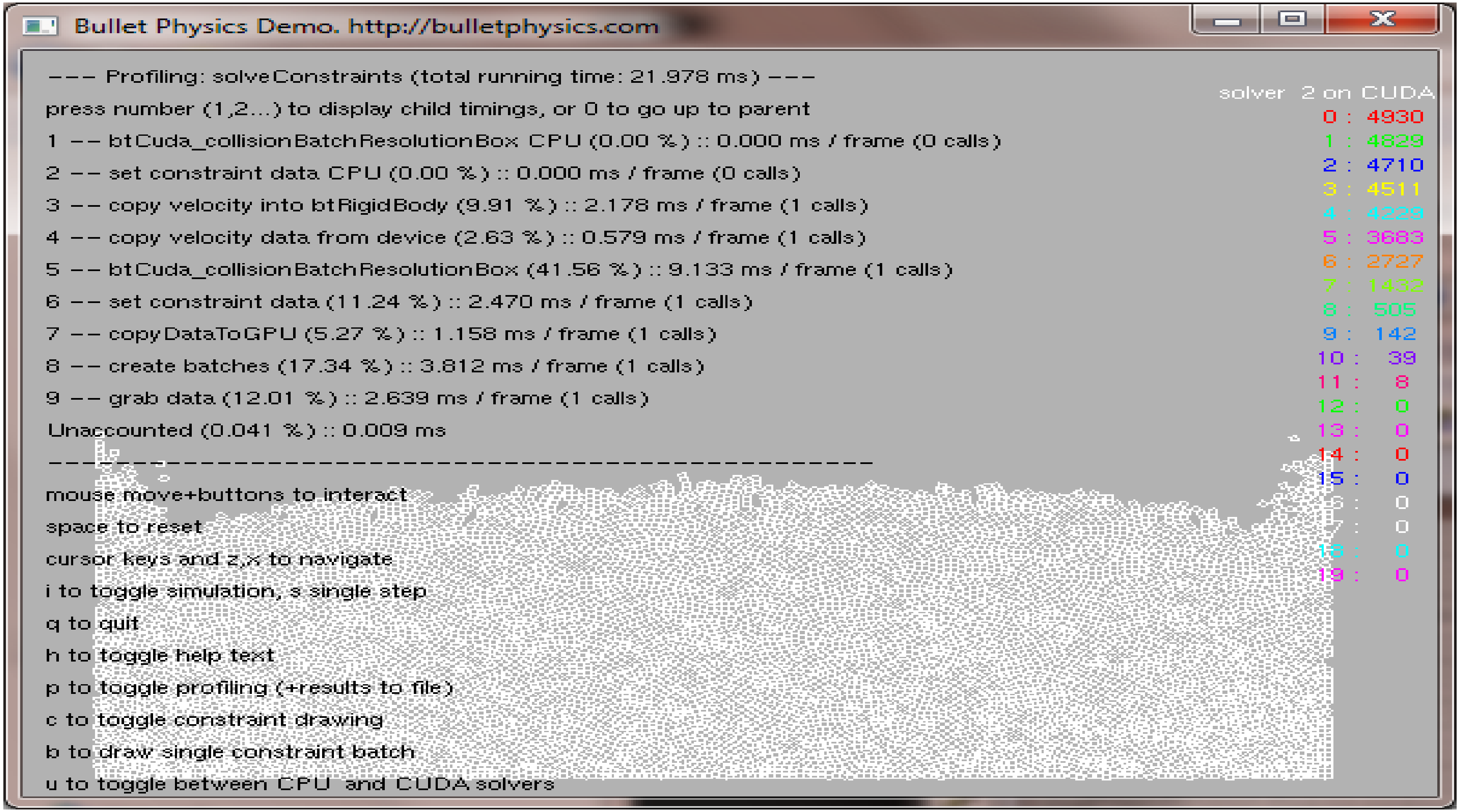
```
__kernel void kSetupBatches(...)
{
    int index = get_global_id(0);
    int currPair = index;
    int objIdA = pPairIds[currPair * 2].x;
    int objIdB = pPairIds[currPair * 2].y;
    int batchId = pPairIds[currPair * 2 + 1].x;
    int localWorkSz = get_local_size(0);
    int localIdx = get_local_id(0);
    for(int i = 0; i < localWorkSz; i++)
    {
        if((i==localIdx)&&(batchId < 0)&&(pObjUsed[objIdA]<0)&&(pObjUsed[objIdB]<0))
        {
                if(pObjUsed[objIdA] == -1)
                        pObjUsed[objIdA] = index;
                if(pObjUsed[objIdB] == -1)
                        pObjUsed[objIdB] = index;
        }

        barrier(CLK_GLOBAL_MEM_FENCE);
    }
}
```

Colored Batches

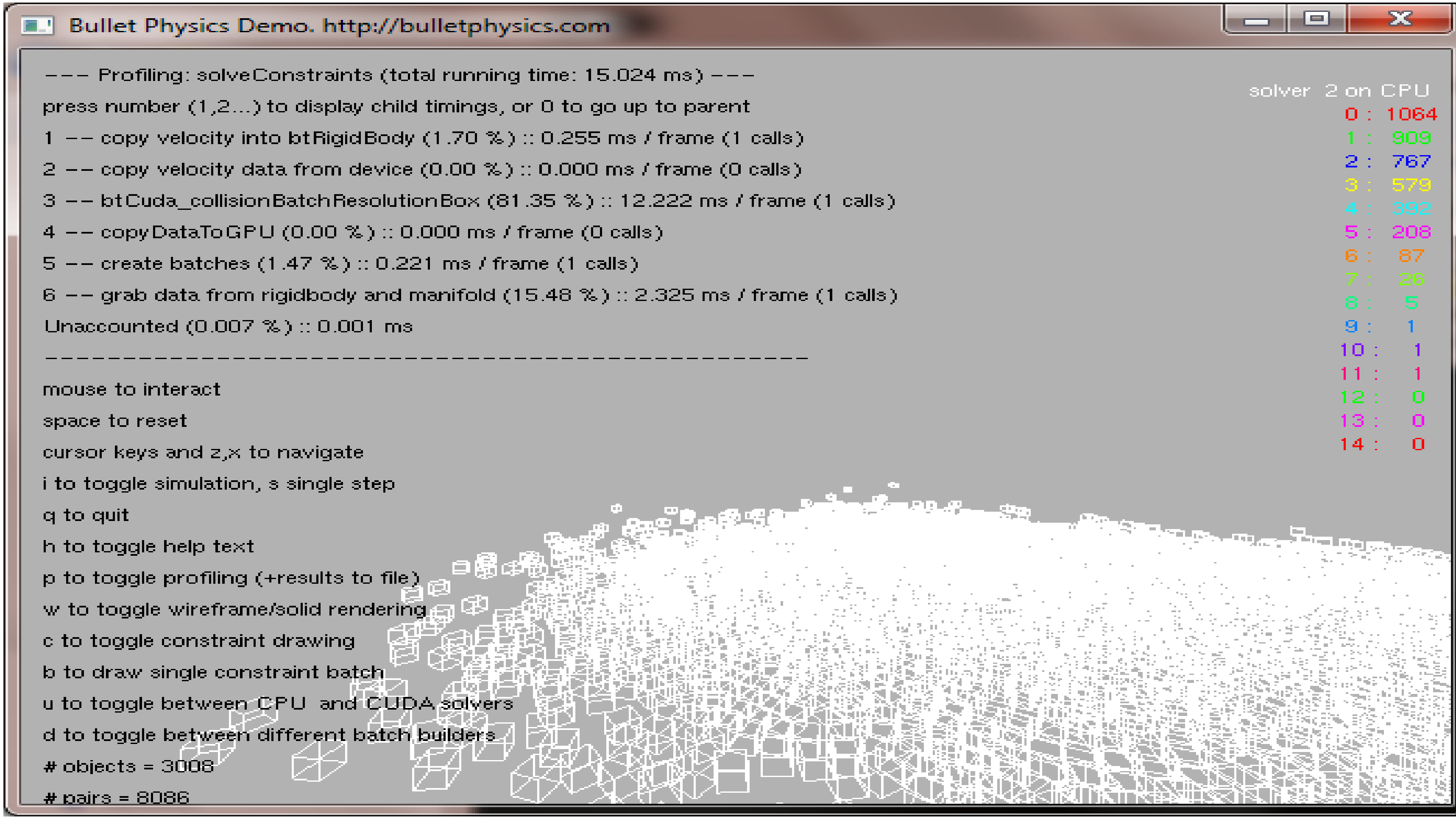
CPU 3Ghz single thread, 2D, 185ms



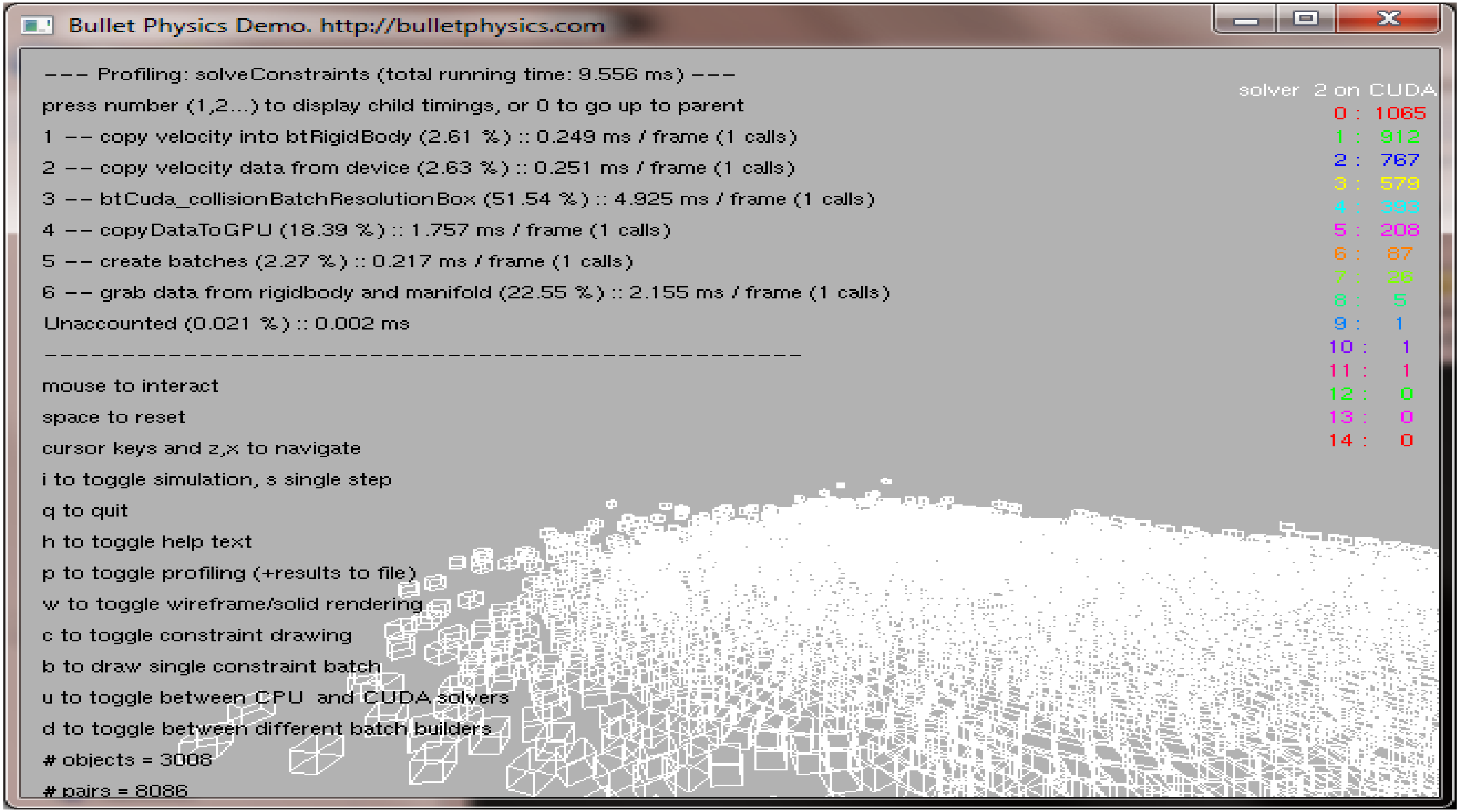
Geforce 260 CUDA, 2D, 21ms



CPU 3Ghz single thread, 3D, 12ms



Geforce 260 CUDA, 3D, 4.9ms



OpenCL Implementation

- Available in SVN branches/OpenCL
  - http://bullet.googlecode.com
- Tested various OpenCL implementations
  - NVidia GPU on Windows PC
  - Apple Snow Leopard on Geforce GPU and CPU
  - Intel, AMD CPU, ATI GPU (available soon)
  - Generic CPU through MiniCL
    - OpenCL kernels compiled and linked as regular C
    - Multi-threaded or sequential for easier debugging

Thanks!

- Questions?
- Visit the Physics Simulation Forum at
  - http://bulletphysics.com
- Email: erwin_coumans@playstation.sony.com