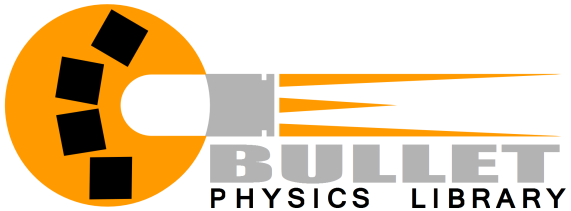

SIGGRAPH2015
Xroads of Discovery

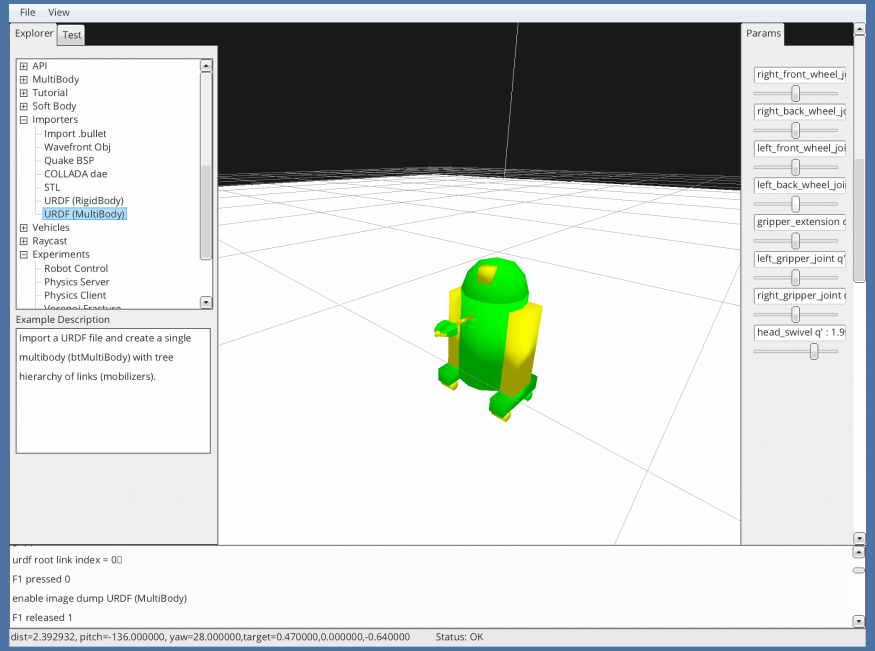


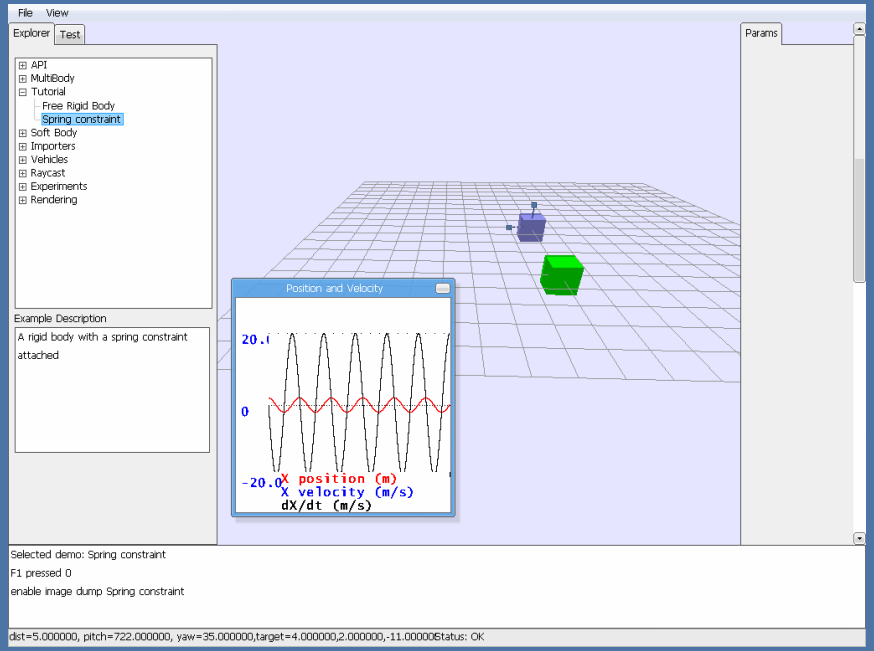
SIGGRAPH2015
Xroads of Discovery
The 42nd International Conference and Exhibition on Computer Graphics and Interactive Techniques

Bullet Physics Simulation
OpenCL GPU Acceleration of rigid body dynamics and collision detection

Erwin Coumans
Google Inc.

Bullet Physics Example Browser



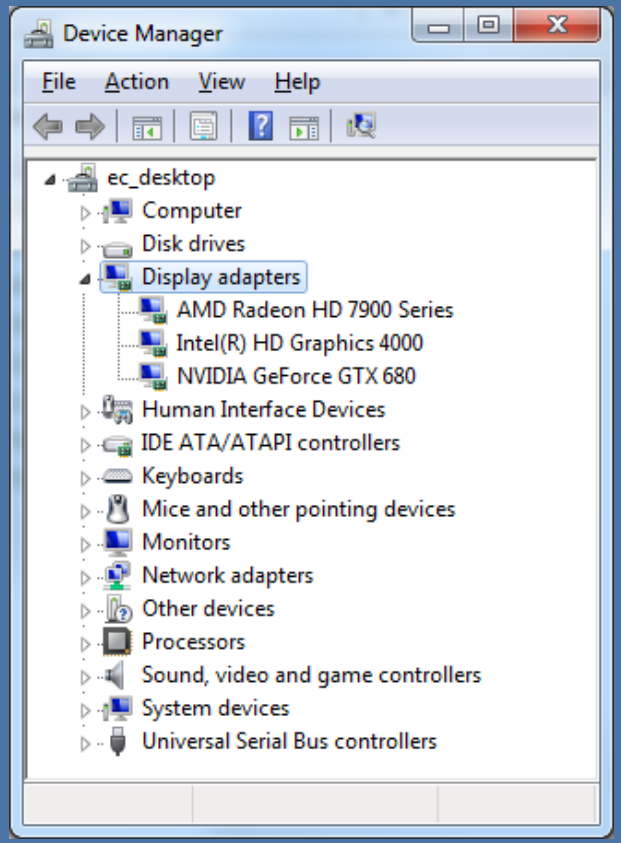


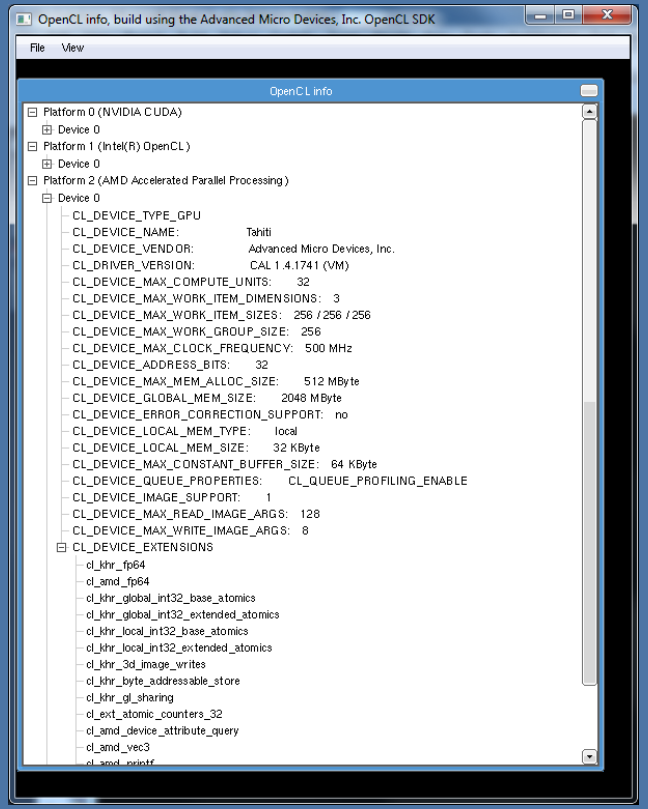
ExampleBrowser --enable_experimental_opencl

github.com/bulletphysics/bullet3

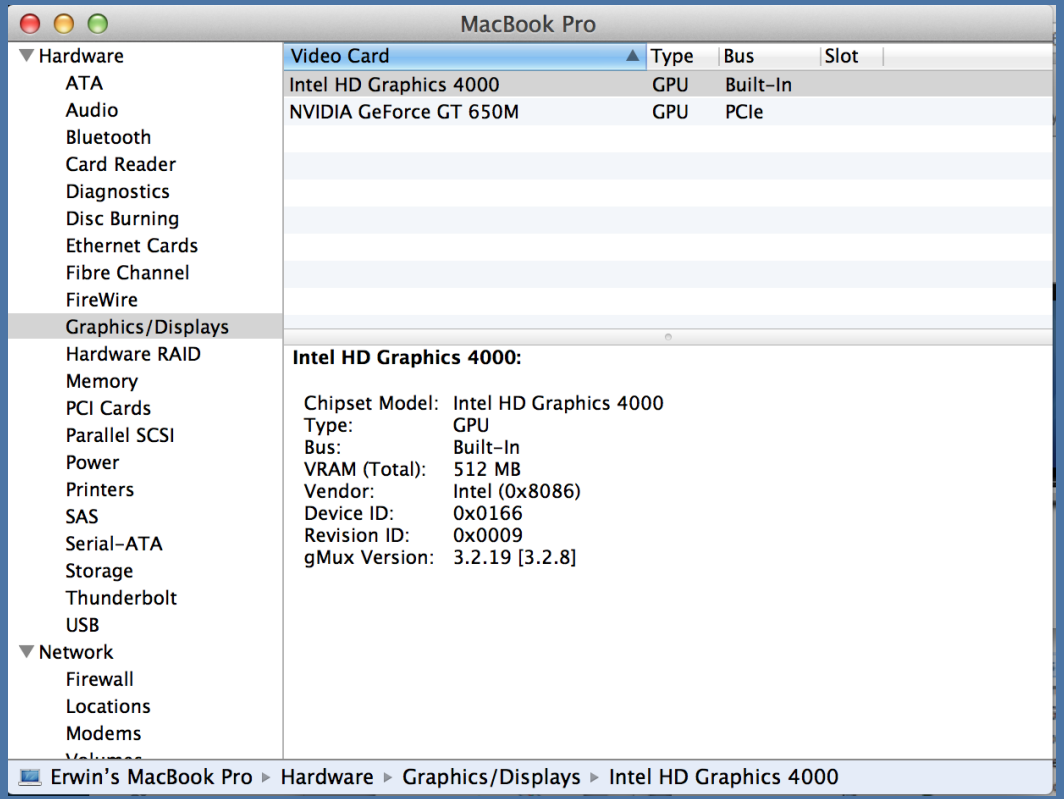
GNU Make

Windows OpenCL Devices

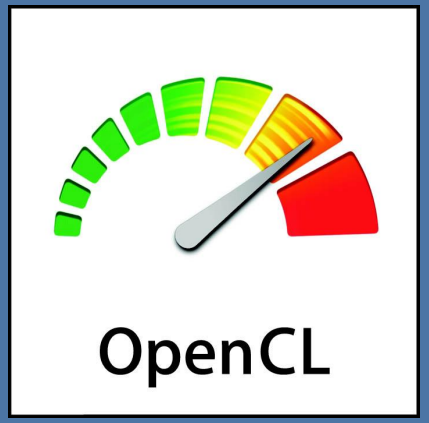




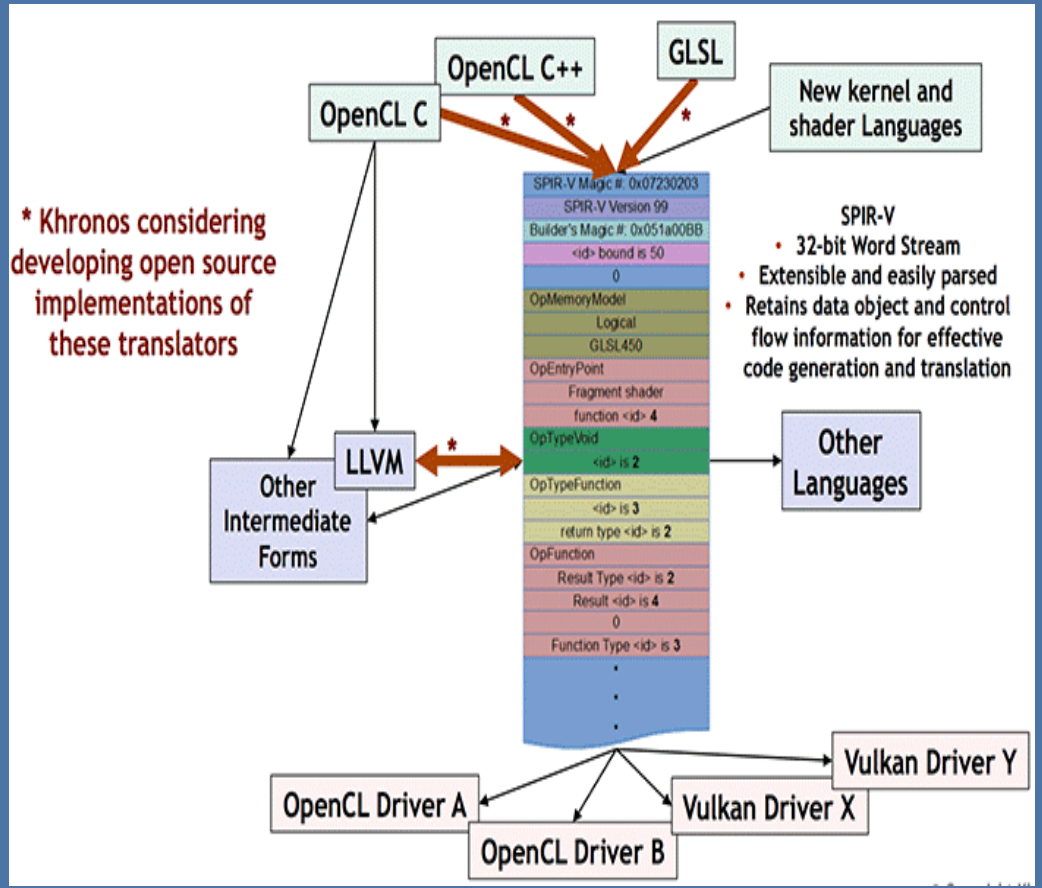
Mac OSX OpenCL Devices



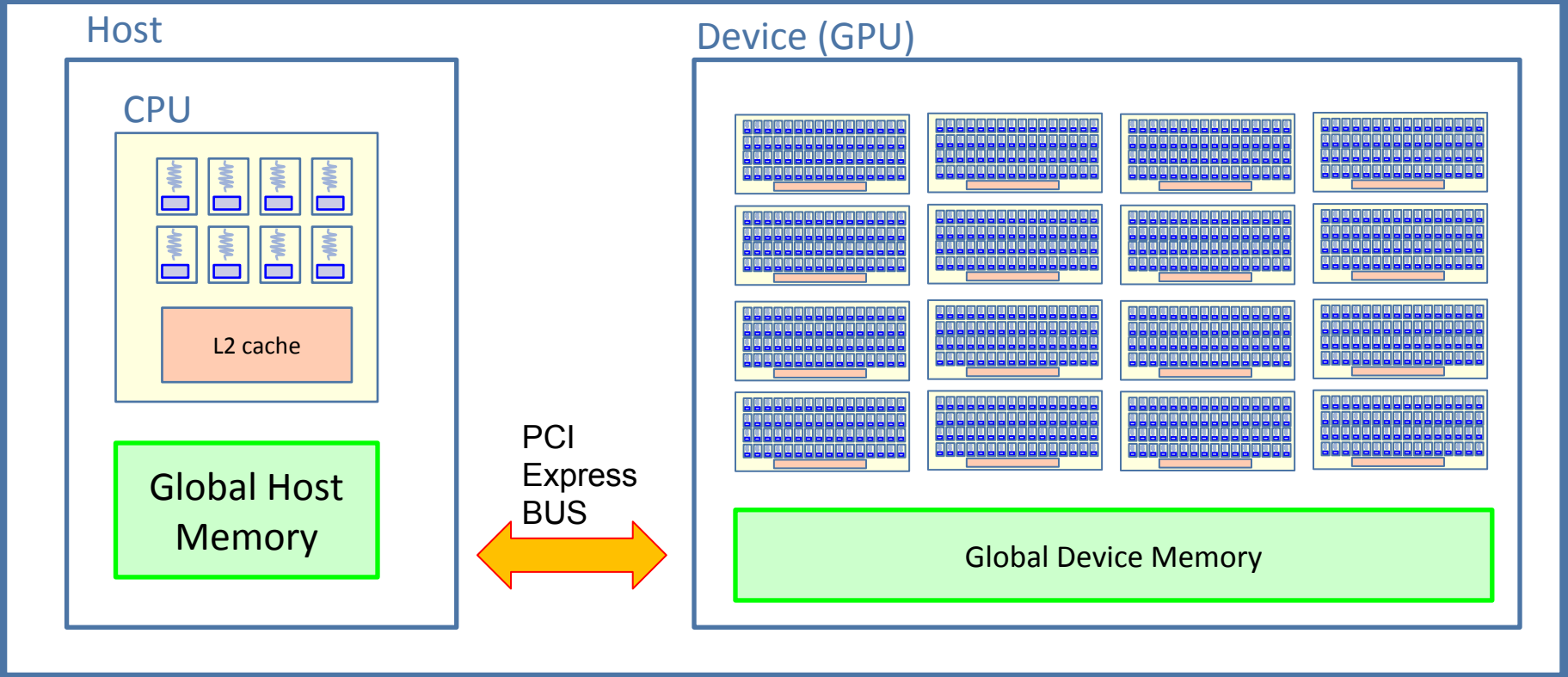
OpenCL, SPIR-V, Vulkan



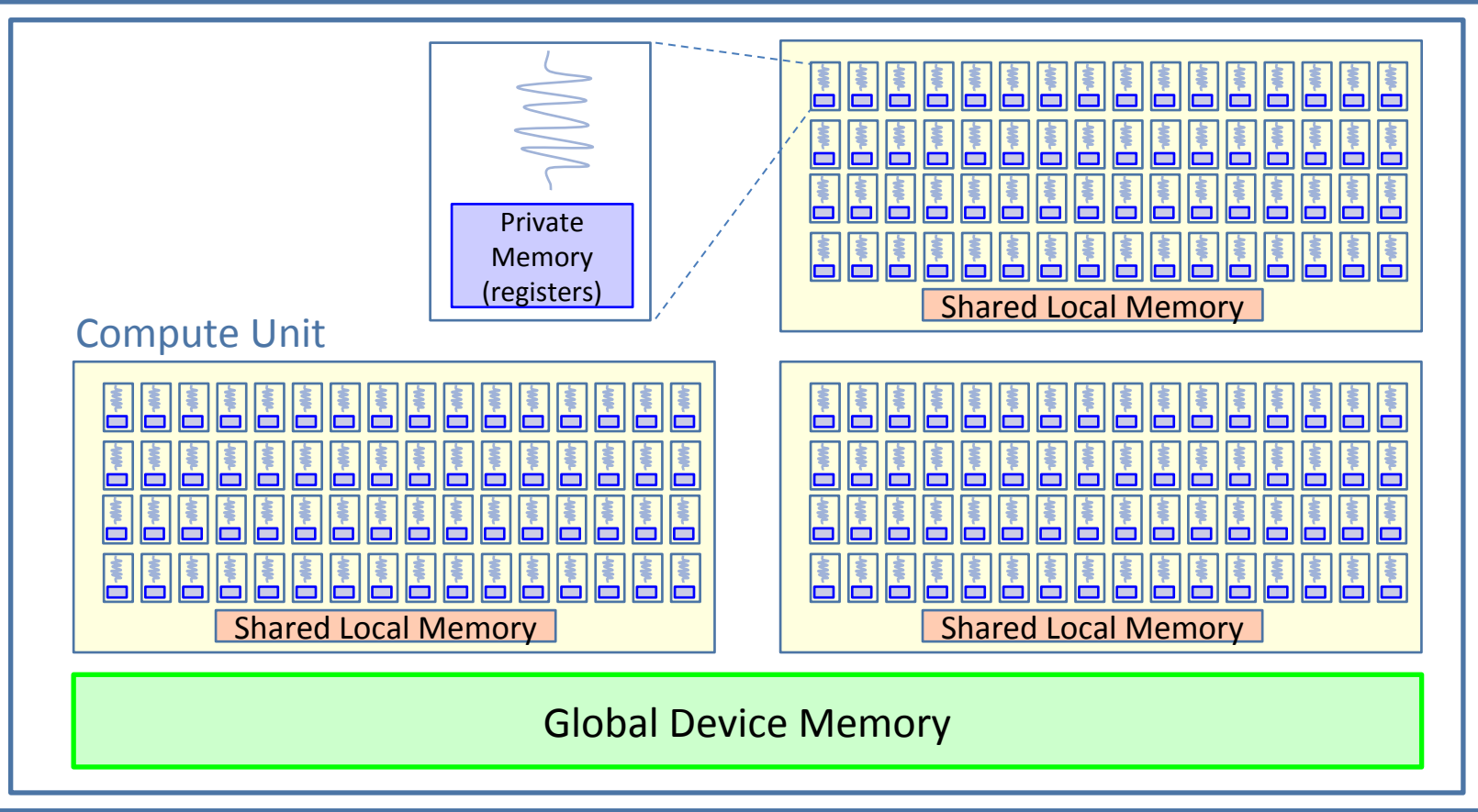
SPIR	SPIR 1.2	SPIR 2.0	SPIR-V
LLVM Interaction	Uses LLVM 3.2	Uses LLVM 3.4	100% Khronos defined Round-trip lossless conversion
Compute Constructs	Metadata/Intrinsics	Metadata/Intrinsics	Native
Graphics Constructs	No	No	Native
Supported Language Feature Set	OpenCL C 1.2	OpenCL C 1.2 OpenCL C 2.0	OpenCL C 1.2 / 2.0 OpenCL C++ GLSL
OpenCL Ingestion	OpenCL 1.2 Extension	OpenCL 2.0 Extension	OpenCL 2.1 Core
Vulkan Ingestion	-	-	Vulkan Core



Host and Device



GPU in a Nutshell



Refactor or Rewrite from Scratch

Sequential to Parallel Algorithms

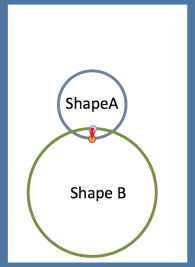
Physics API suitable for GPU? Callbacks, Extensibility etc.

C++ virtuals, templates to C99

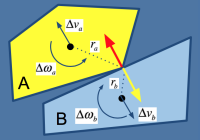
Array of Structures to Structure of Arrays

Rigid Body Simulation Loop

point on A
point on B
normal B->A
distance



$$p_{correction} = M_{effective} \Delta v_{desired} \cdot n$$



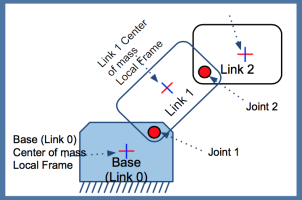
$$p_{correction} = \frac{-\Delta v_{ab} \cdot n}{n \cdot n(\frac{1}{M_a} + \frac{1}{M_a}) + (\frac{r_a \times n}{I_a}) \times r_a + (\frac{r_b \times n}{I_b}) \times r_b}$$

Collision Detection
Contact,
Time of Impact

Forward Dynamics
Compute Inertia,Forces and Accelerations

Numerical Time Integration
Update Velocity, Position

$$A\lambda + b \geq 0$$
$$\lambda \geq 0$$
$$\lambda(A\lambda + b) = 0$$



$$v_{t+\Delta t} = v_t + a\Delta t = v_t + \frac{F_{ext} + F_c}{m}\Delta t = v_t + \frac{F_{ext}}{m}\Delta t + \frac{Impulse_c}{m}$$
$$x_{t+\Delta t} = x_t + v_{t+\Delta t}\Delta t$$

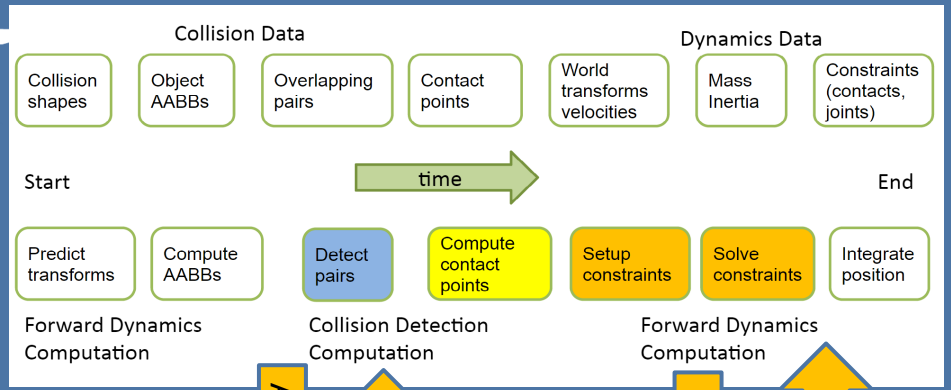
F_c could be constraint forces such as contact, friction, joints

Refactor: GPU as Co-Processor

Host Memory Gather and Scatter is SLOW

pointers to indices

PCI BUS latency & work item dispatch is SLOW



AABBs & Transforms

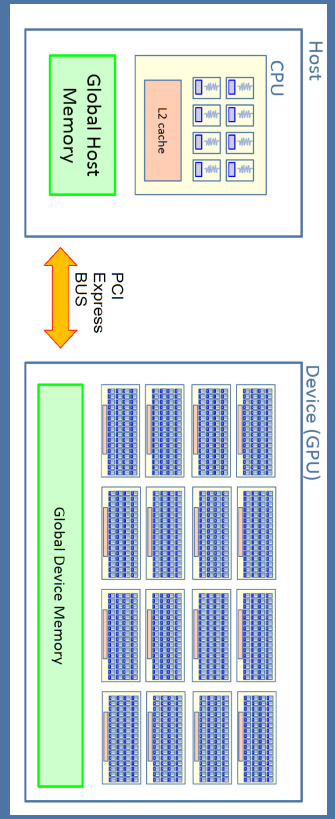
Overlapping Pairs

Constraint Rows

Forces, Accelerations

GPU Pair Detection

CUDA Constraint Solver



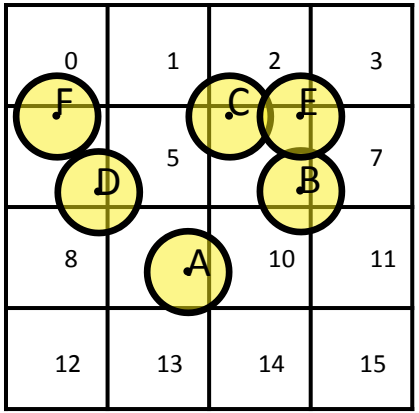
1st Rewrite: GPU Rigid Body Pipeline (~2008)

Detect Contact pairs

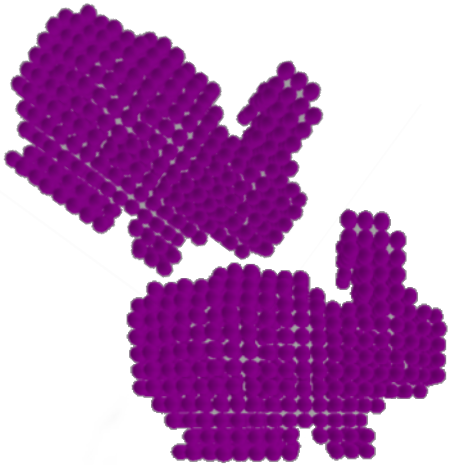
Compute contact points

Setup Contact constraints

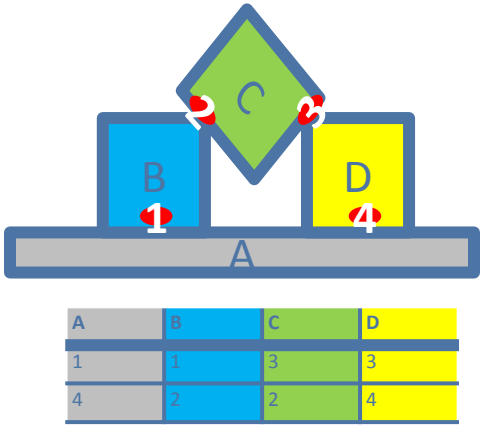
Solve constraints



Uniform grid

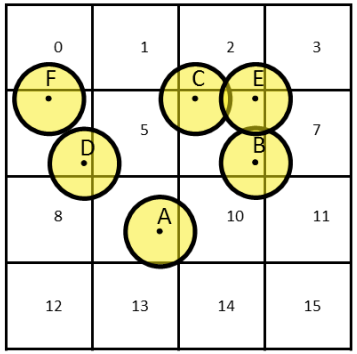


Spherical Voxelization



CPU batch and GPU solve
(Slow PCI BUS dispatch)

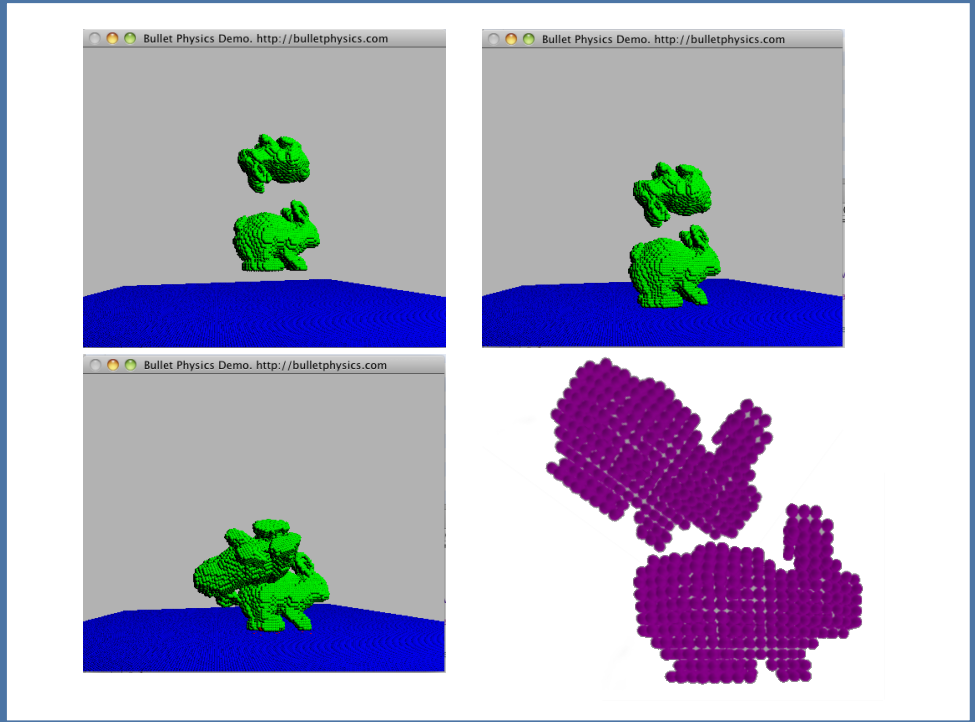
Parallel Primitives: Radix Sort, Prefix Scan



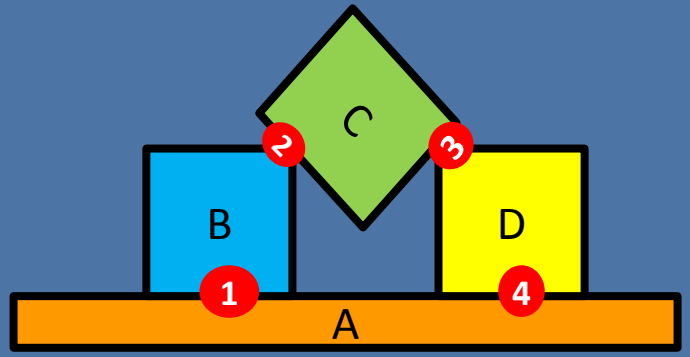
Cell Index	Cell Start
0	
1	
2	
3	
4	0
5	
6	2
7	
8	
9	5
10	
11	
12	
13	
14	
15	

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

GPUs love Fine Grain Parallelism



Reordering Constraints



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

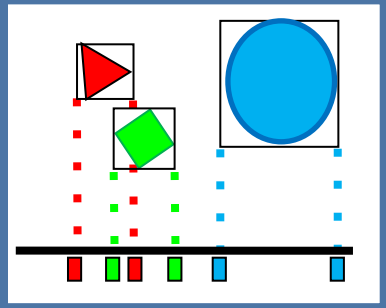
	A	B	C	D
Batch 0	1	1	3	3
Batch 1	4	2	2	4

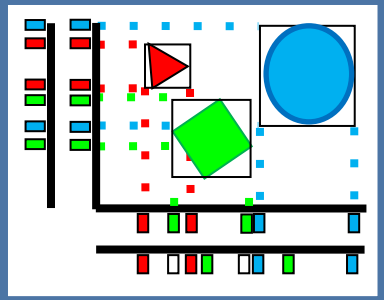
CPU sequential batch creation

```
while( nIdxSrc ) {
    nIdxDst = 0;    int nCurrentBatch = 0;
    for(int i=0; i<N_FLG/32; i++) flg[i] = 0; //clear flag
    for(int i=0; i<nIdxSrc; i++)    {
        int idx = idxSrc[i];  btAssert( idx < n );
        //check if it can go
        int aIdx = cs[idx].m_bodyAPtr & FLG_MASK;   int bIdx = cs[idx].m_bodyBPtr & FLG_MASK;
        u32 aUnavailable = flg[ aIdx/32 ] & (1<<(aIdx&31));u32 bUnavailable = flg[ bIdx/32 ] & (1<<(bIdx&31));
        if( aUnavailable==0 && bUnavailable==0 )   {
            flg[ aIdx/32 ] |= (1<<(aIdx&31));   flg[ bIdx/32 ] |= (1<<(bIdx&31));
            cs[idx].getBatchIdx() = batchIdx;
            sortData[idx].m_key = batchIdx; sortData[idx].m_value = idx;
            nCurrentBatch++;
            if( nCurrentBatch == simdWidth ) {
               nCurrentBatch = 0;
               for(int i=0; i<N_FLG/32; i++) flg[i] = 0;
            }
        }
        else  {
            idxDst[nIdxDst++] = idx;
        }
    }
    swap2( idxSrc, idxDst ); swap2( nIdxSrc, nIdxDst );
    batchIdx ++;
}
```

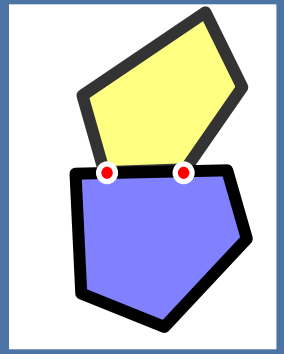
2st Rewrite: GPU Rigid Body Pipeline (~2012)

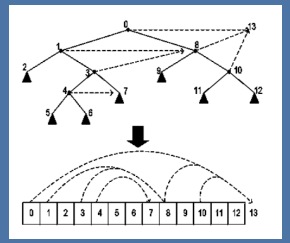
Detect Contact pairs



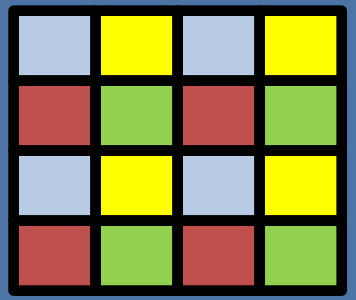


Compute contact points

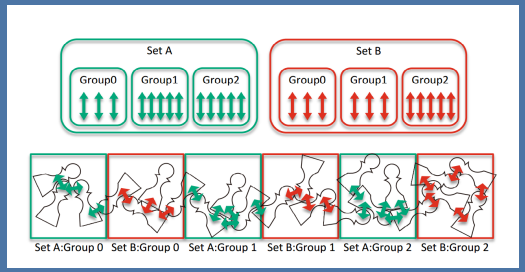




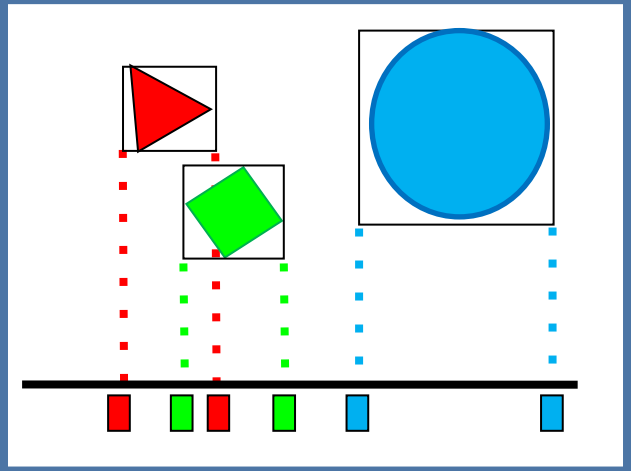
Setup Contact constraints

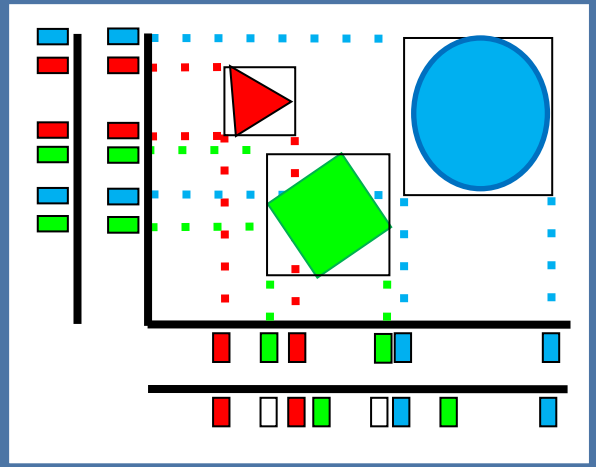


Solve constraint s

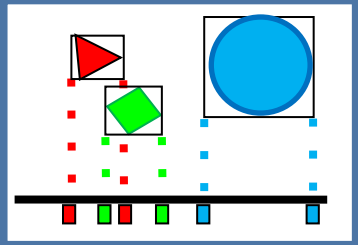


GPU Pair Search





Optimizations in GPU 1-axis SAP

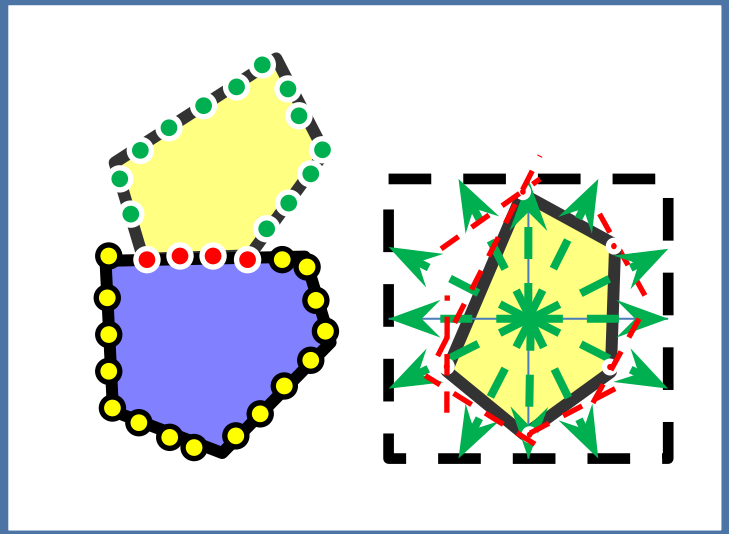


- Local memory
  - blocks to fetch AABBs and re-use them within a workgroup (requires a barrier)
- Reduce global atomic operations
  - Private memory to accumulate overlapping pairs (append buffer)
- Local atomics
  - Determine early exit condition for all work items within a workgroup
- Load balancing
  - One work item per object, multiple work items for large objects

- See opencl/gpu_broadphase/kernels/sapFast.cl and sap.cl

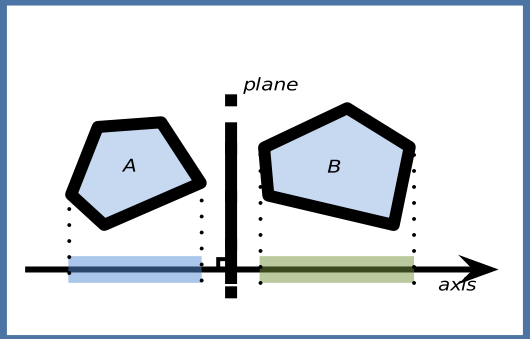
(contains un-optimized and optimized version of the kernel for comparison)

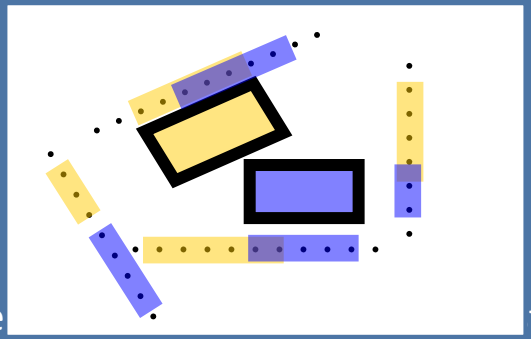
Cubemap: Convex Heightfield Contact

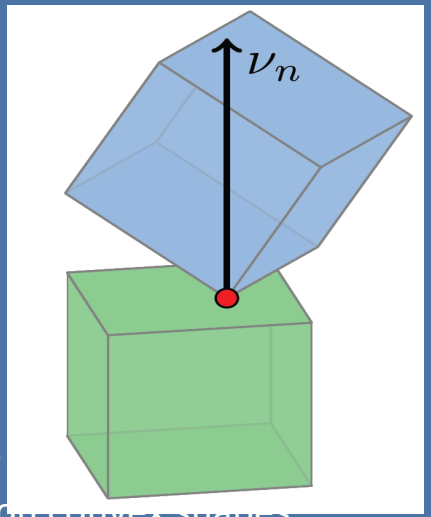


GPU Separating Axis Test

- Face normal A
- Face normal B
- Edge-edge normal

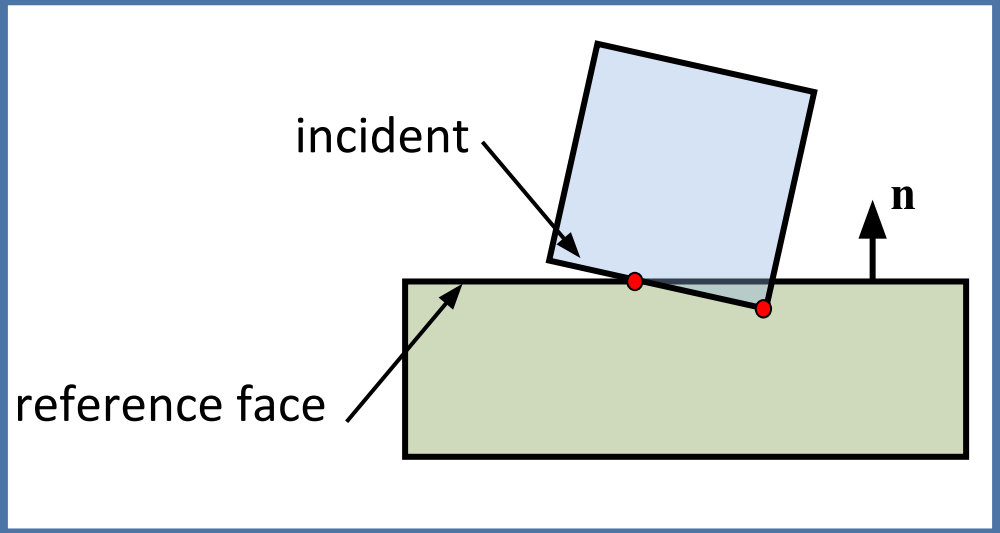


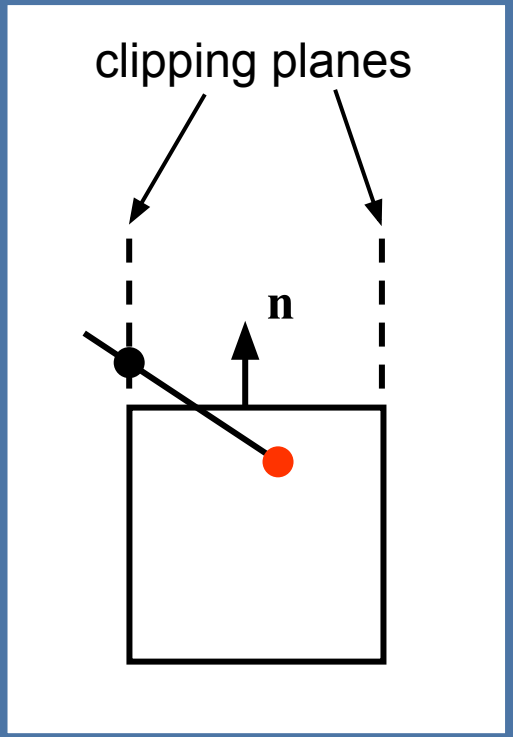




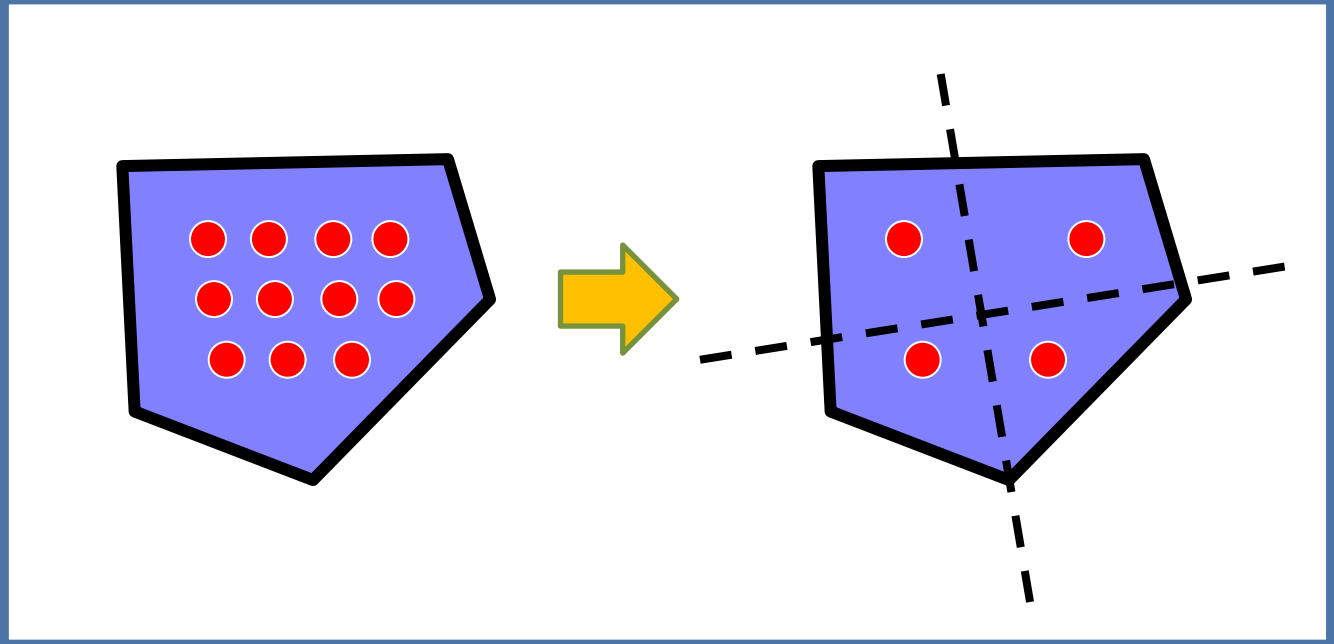
- U l: one ts for
- Precise solution and faster than height field approximation for low-resolution convex shapes
- See opencl/gpu_sat/kernels/sat.cl

GPU Sutherland Hodgeman Clipping





GPU Contact Reduction



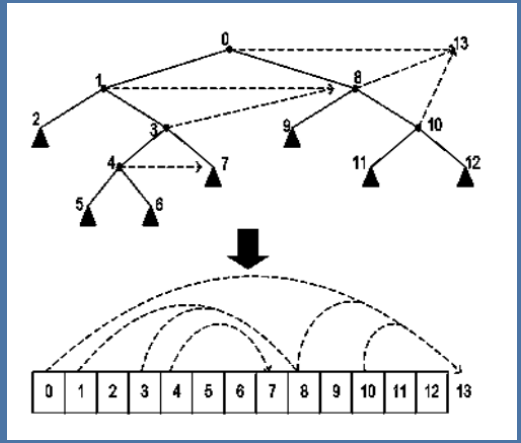
SAT Convex Collision Refactor

Find Potential Overlapping Pairs
For each convex-convex pair
findSeparatingAxis
For each SAT pass
findClippingFaces
clipHullHull
contactReduction
For each compound pair
Search potential overlapping children
For each pair of overlapping children
findSeparatingAxis
For each SAT pass
findClippingFaces
clipHullHull
contactReduction
For each concave pair
For each triangle-convex overlap
findSeparatingAxis
findClippingFaces
clipHullHull, contactReduction

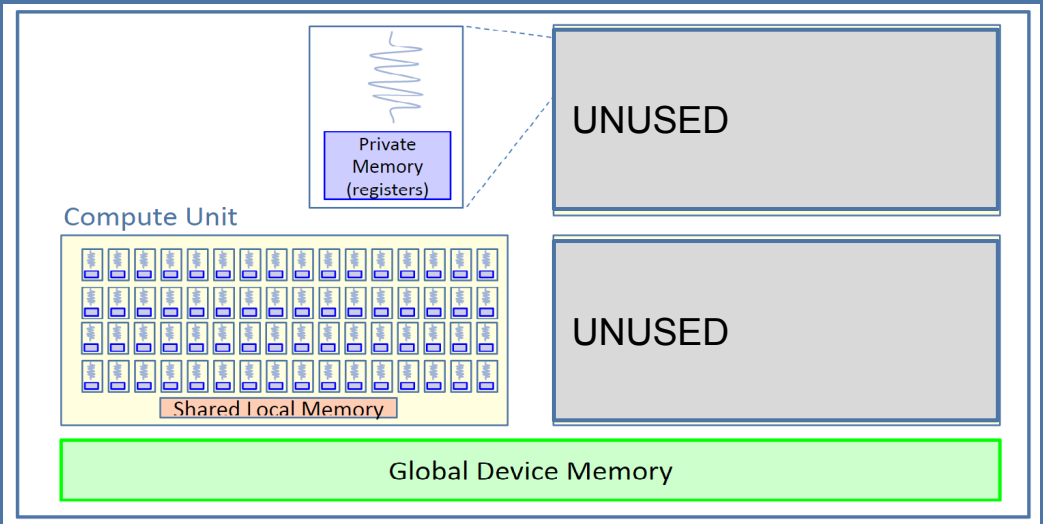
Find Potential Overlapping Pairs
For each compound pair
Find Overlapping Compound Pairs
For each concave pair
Find Overlapping tri-convex Pairs
For each Pair
findSeparatingAxisKernel
findConcaveSeparatingAxisKernel
For each SAT pass
clipHullHullKernel
clipCompoundsHullHullKernel
clipHullHullConcaveConvexKernel
findClippingFacesKernel
clipFacesAndContactReductionKernel
newContactReductionKernel

GPU trimesh and compound shapes

- Create skip indices for faster traversal
- Create subtrees that fit in Local Memory
- Stream subtrees for entire wavefront/warp
- Quantize Nodes
  - 16 bytes/node

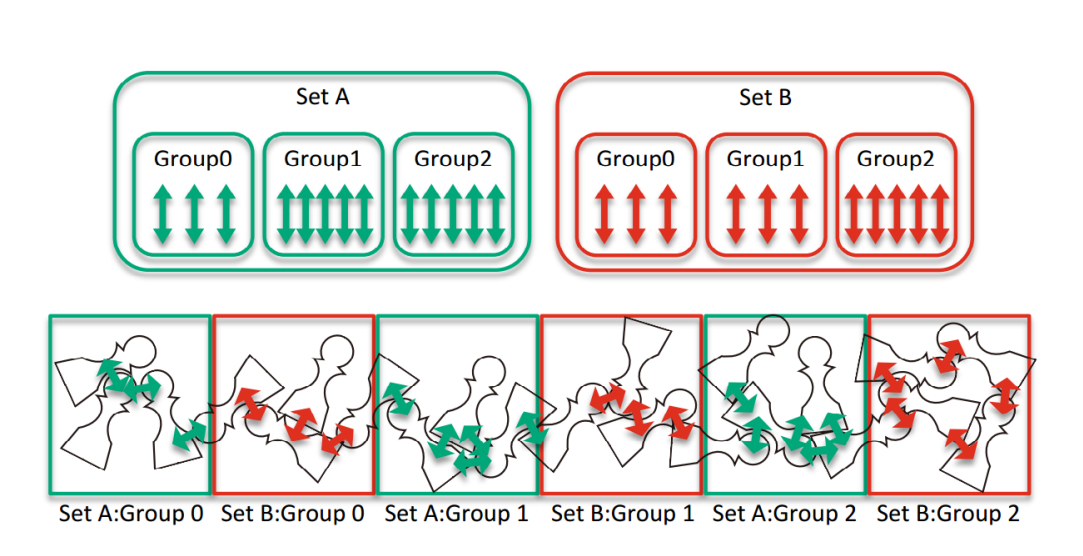


Naïve GPU batch creation



- Use a single Compute Unit
- All threads in the Compute Unit synchronize the locking of bodies using atomics and barriers
- Didn’t scale well for larger scale simulations (>~30k)

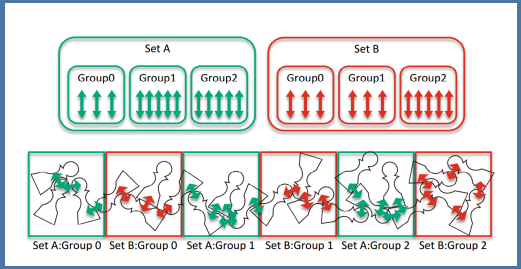
2-way Batch Generation



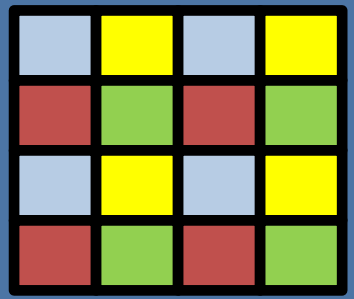
See “A parallel constraint solver for a rigid body simulation”, Takahiro Harada, http://dl.acm.org/citation.cfm?id=2077378.2077406
Source code at opencl\gpu_rigidbody\kernels\solveContact.cl and other solve*.cl

2-way Batch Generation

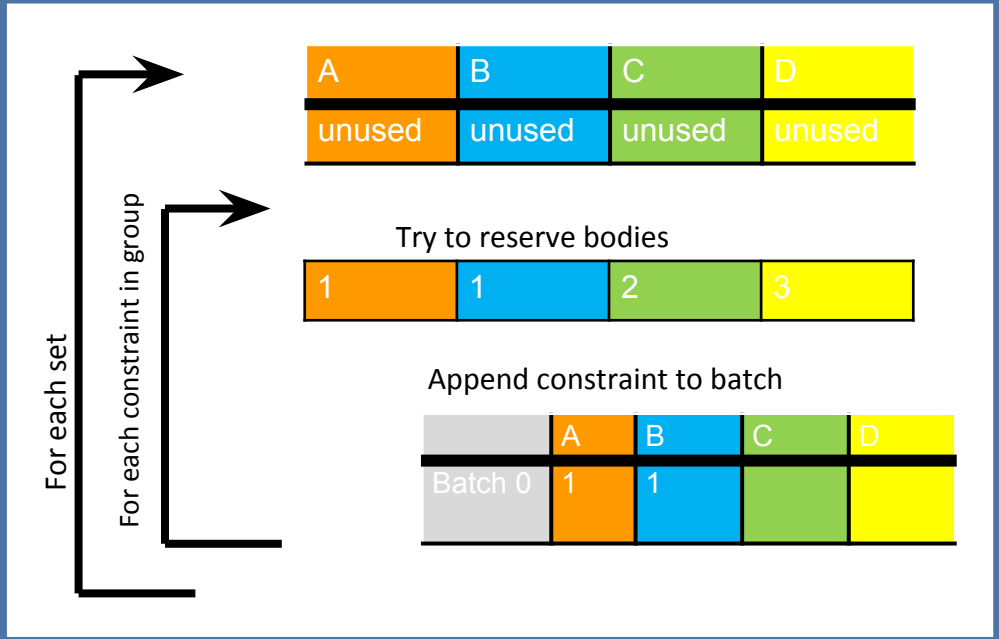
1: Create Disjoint Sets



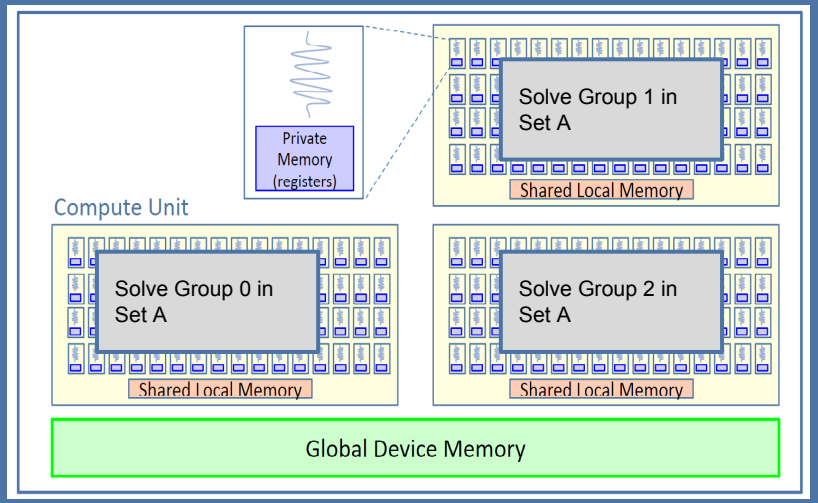
1: Spatial Sort to create Disjoint Sets, or alternative method



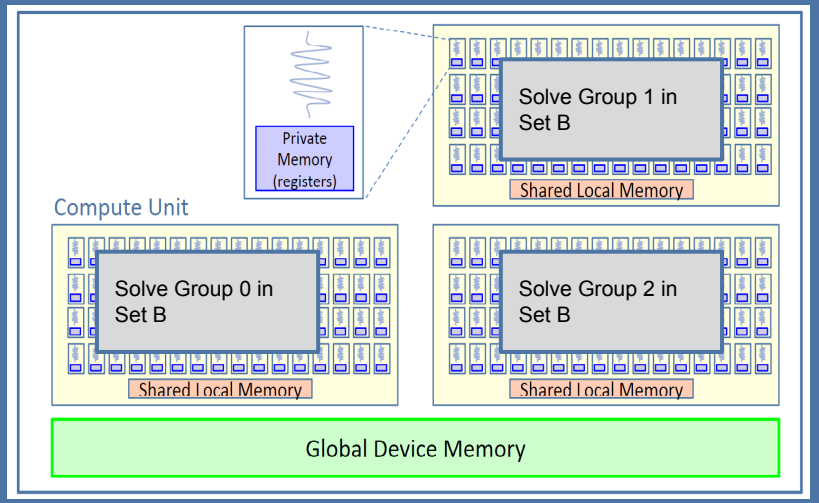
2: For each set, Compute Units solve groups in parallel



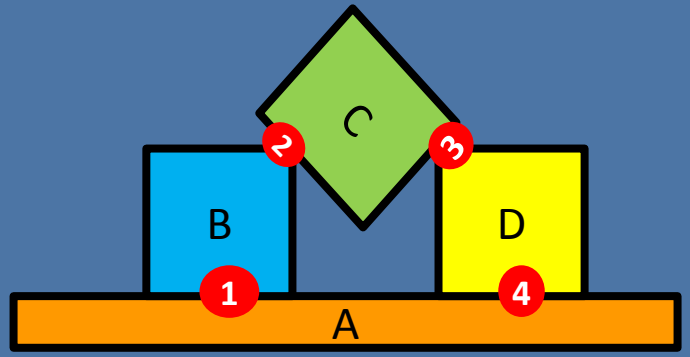
2-way Batch Generation



Host SYNC (enqueue work)

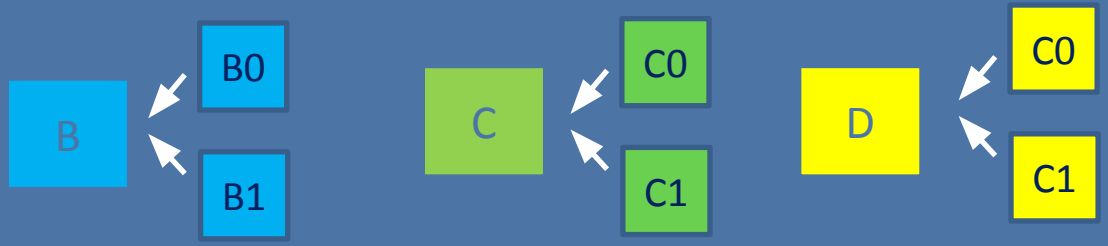


Mass Splitting+Jacobi ≅ PGS



A	B0	B1	C0	C1	D1	D1	A
1	1	2	2	3	3	4	4

Parallel Jacobi



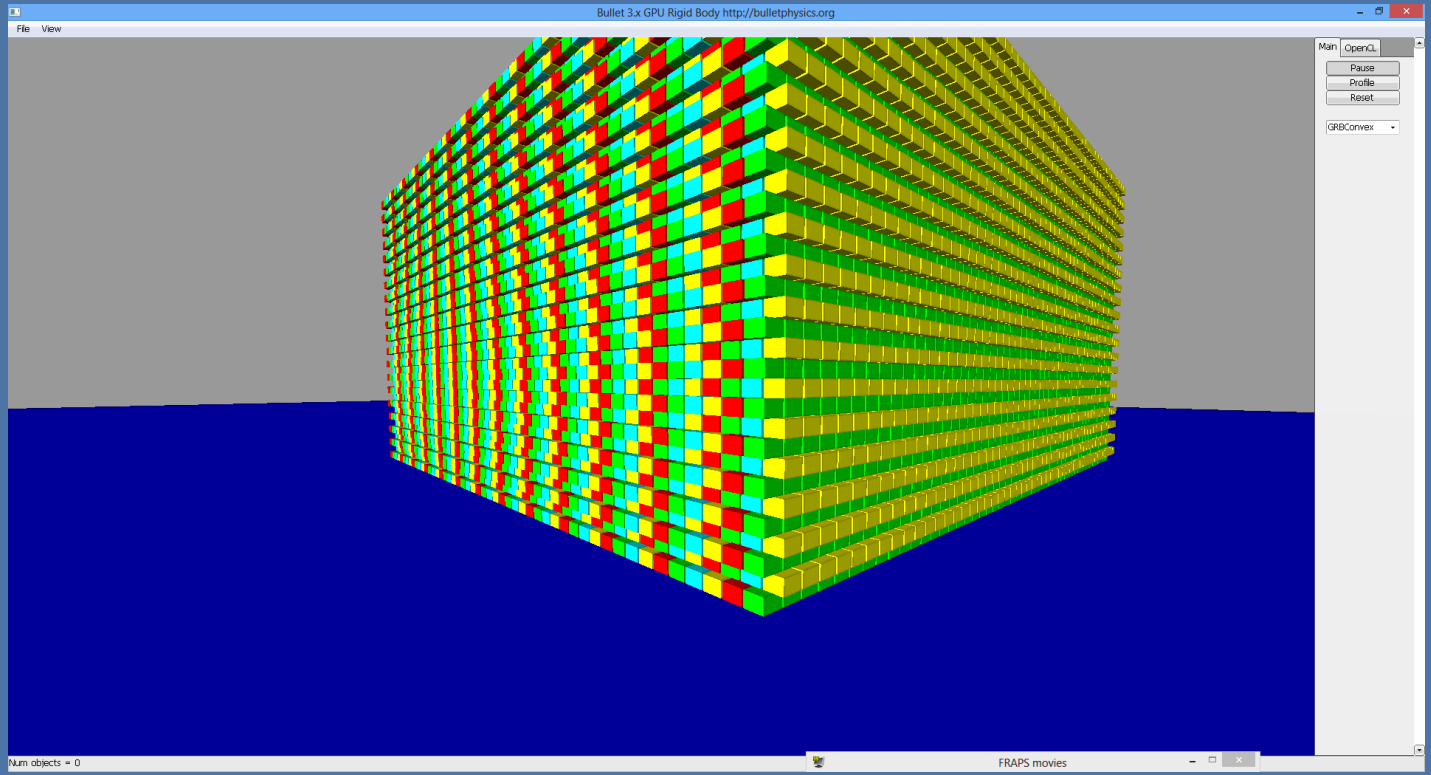
Averaging velocities

- See “Mass Splitting for Jitter-Free Parallel Rigid Body Simulation” by Tonge et. al.

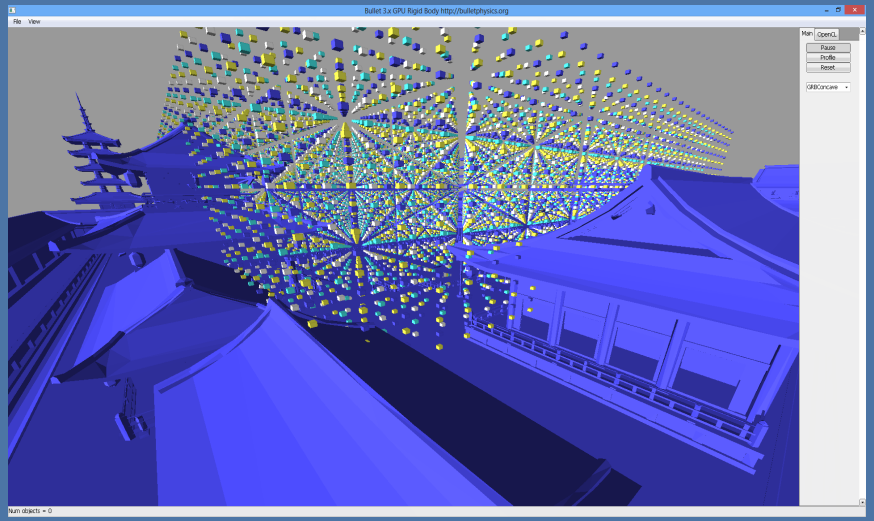
Over 50 OpenCL™ kernels

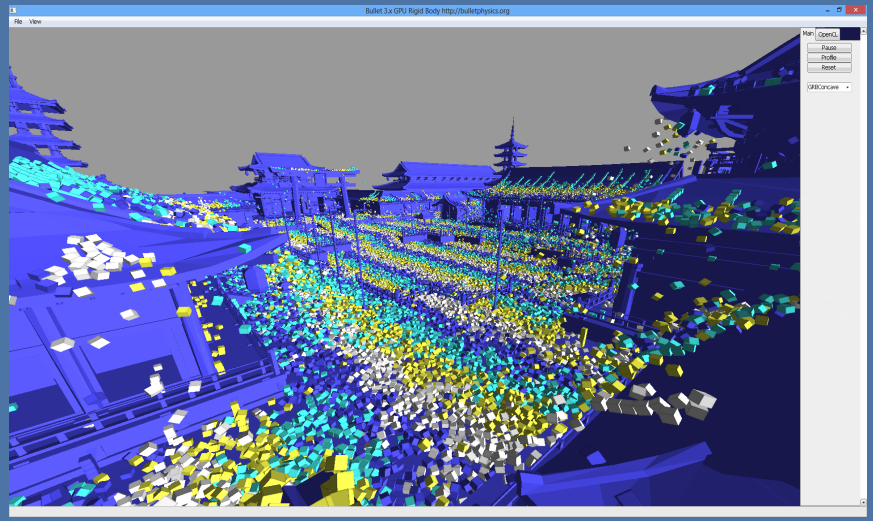
AddOffsetKernel	AverageVelocitiesKernel	BatchSolveKernelContact	BatchSolveKernelFriction	ClearVelocitiesKernel	ContactToConstraintKernel
ContactToConstraintSplitKernel	CopyConstraintKernel	CountBodiesKernel	CreateBatches	CreateBatchesNew	FillFloatKernel
FillInt2Kernel	FillIntKernel	FillUnsignedIntKernel	LocalScanKernel	PrefixScanKernel	ReorderContactKernel
SearchSortDataLowerKernel	SearchSortDataUpperKernel	SetSortDataKernel	SolveContactJacobiKernel	SolveFrictionJacobiKernel	SortAndScatterKernel
SortAndScatterSortDataKernel	StreamCountKernel	StreamCountSortDataKernel	SubtractKernel	TopLevelScanKernel	UpdateBodyVelocitiesKernel
bvhTraversalKernel	clipCompoundsHullHullKernel	clipFacesAndContactReductionKernel	clipHullHullConcaveConvexKernel	clipHullHullKernel	computePairsKernel
computePairsKernelTwoArrays	copyAabbsKernel	copyTransformsToVBOKernel	extractManifoldAndAddContactKernel	findClippingFacesKernel	findCompoundPairsKernel
findConcaveSeparatingAxisKernel	findSeparatingAxisKernel	flipFloatKernel	initializeGpuAabbsFull	integrateTransformsKernel	newContactReductionKernel
processCompoundPairsKernel	scatterKernel				

100k Box Stack at 60FPS

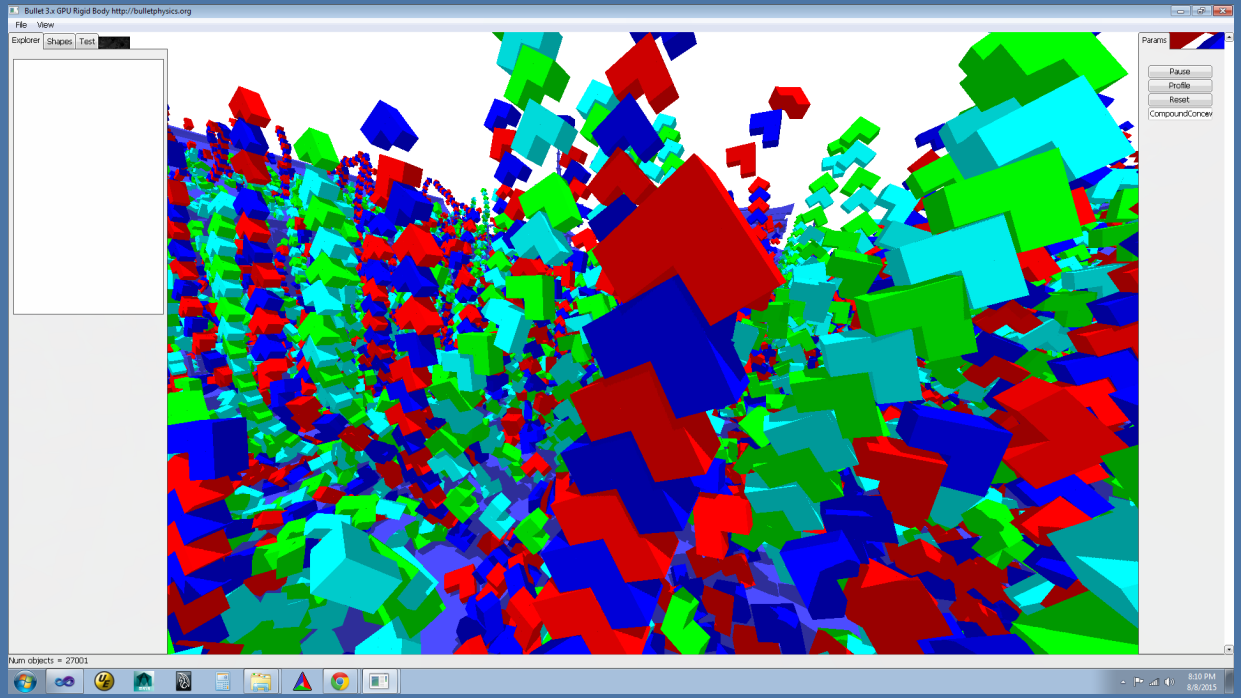


Concave Static Triangle Mesh

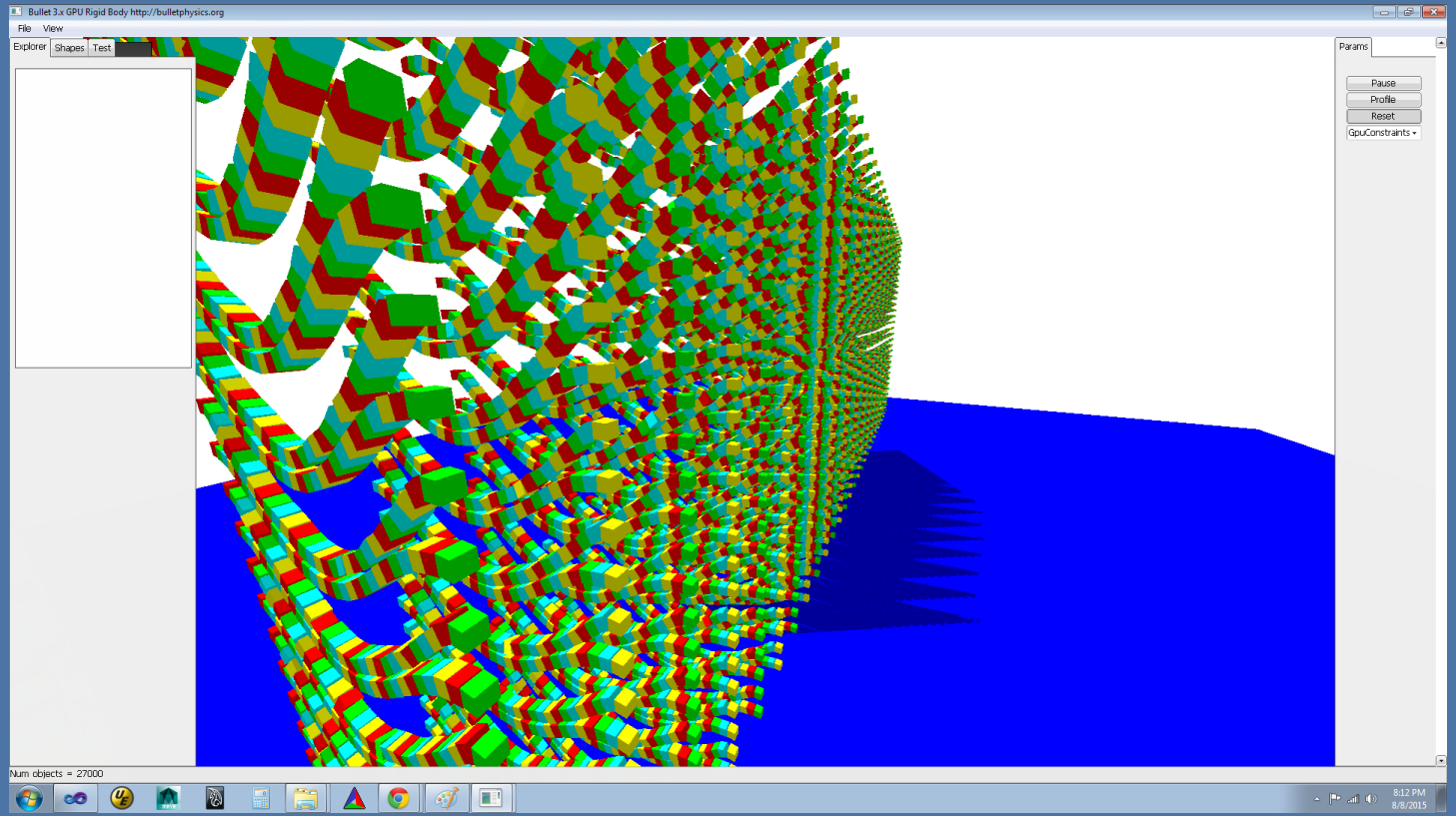




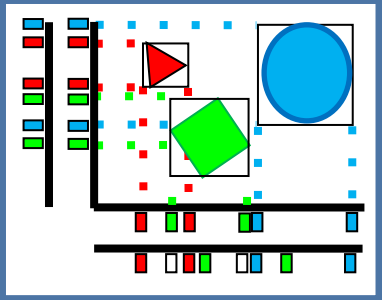
GPU Compound Collision

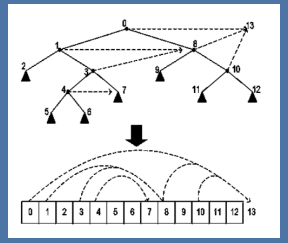


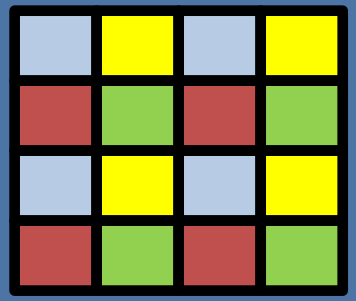
GPU Hinge Joints

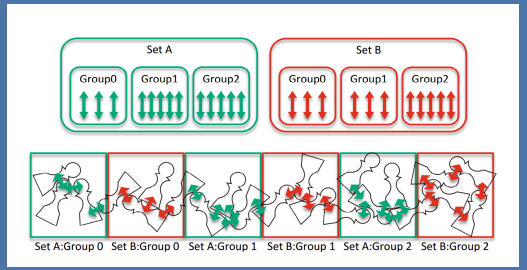


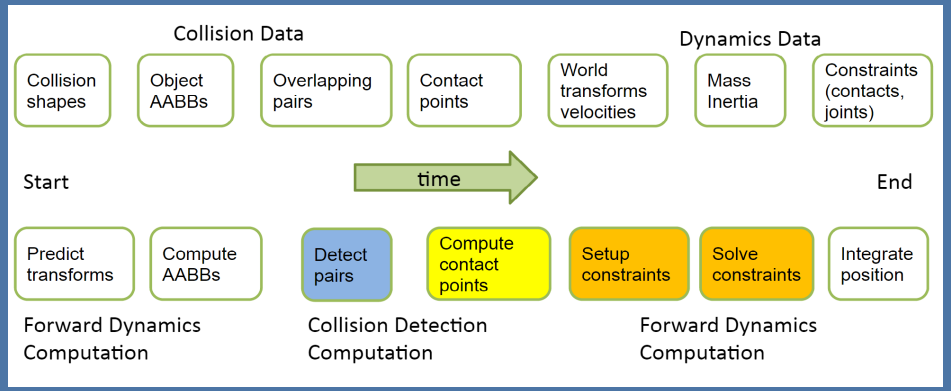
Ongoing: uniting API and code bases

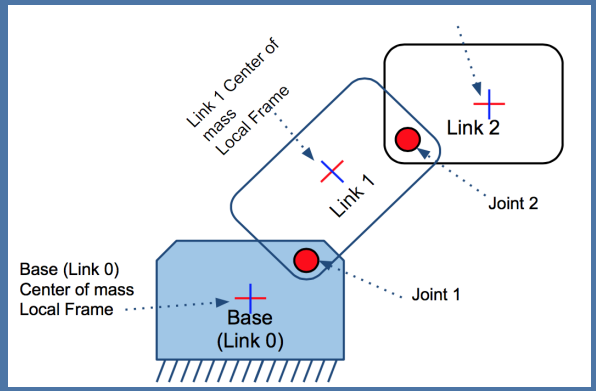








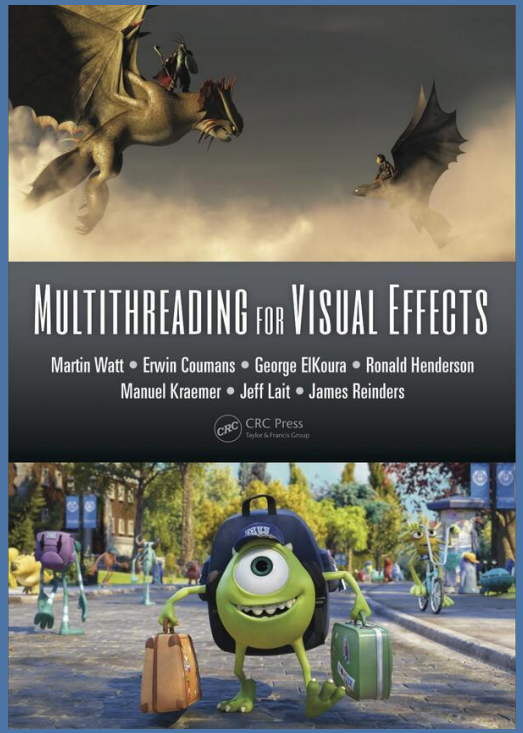




Thank you! Questions?

See https://github.com/bulletphysics

ExampleBrowser --enable_experimental_opencl



https://github.com/bulletphysics/bullet3/tree/old_demos

App_Bullet3_OpenCL_Demos_clew