OpenCL accelerated rigid body and collision detection

Erwin Coumans

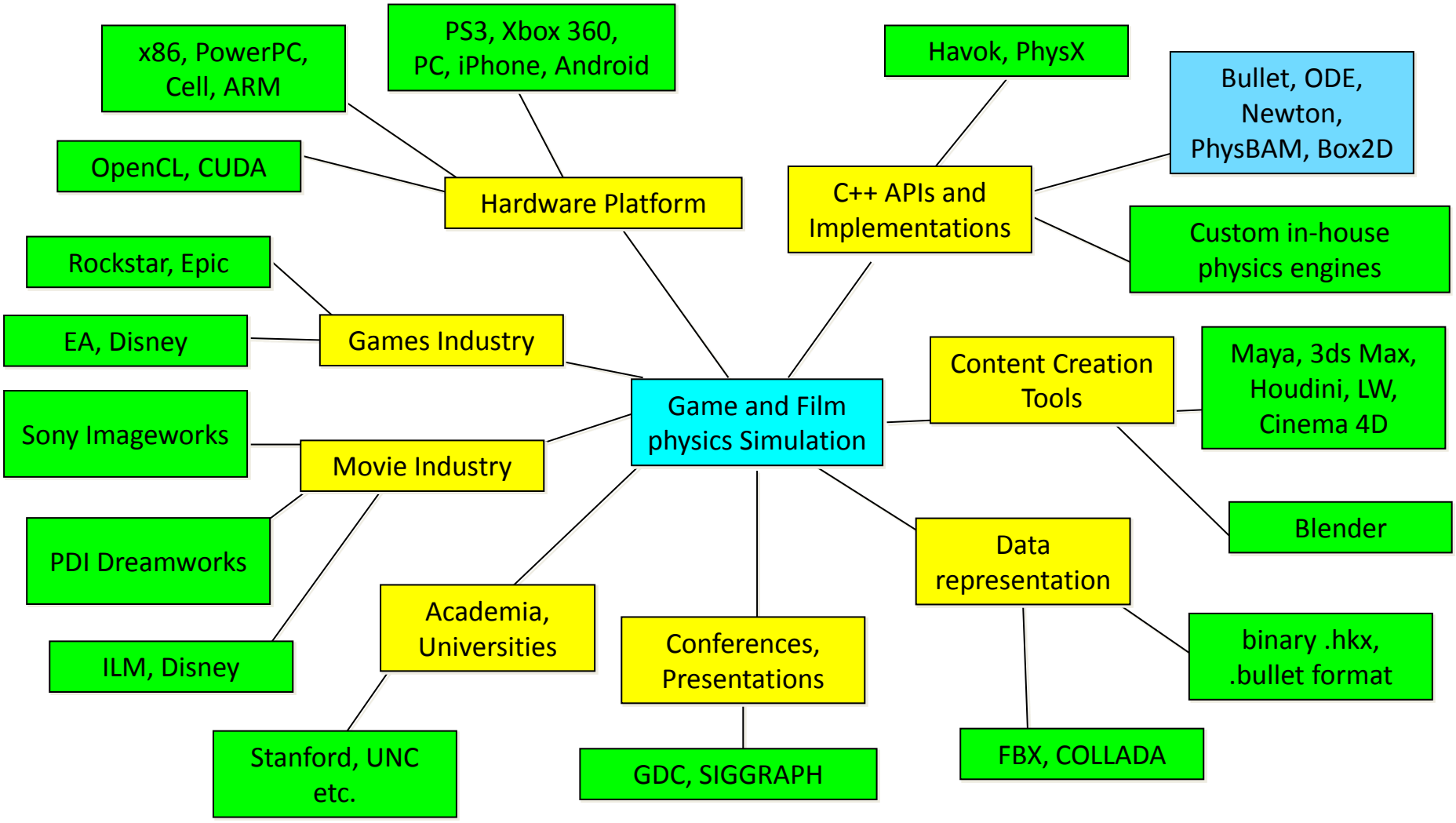
Advanced Micro Devices

Robotics: Science and Systems Conference 2011

Overview

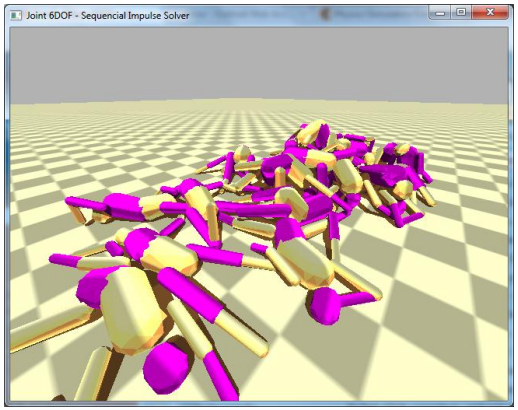
- Intro
- GPU broadphase acceleration structures
- GPU convex contact generation and reduction
- GPU BVH acceleration for concave shapes
- GPU constraint solver

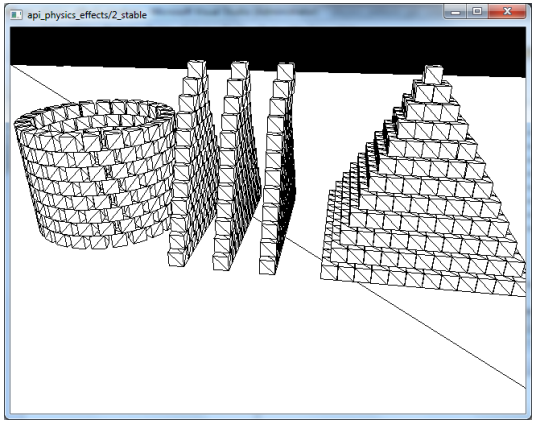
Industry view

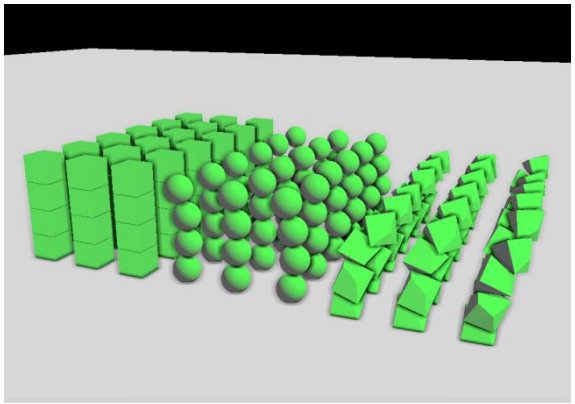


Our open source work

- 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

Rigid body

State vector

$$X = \begin{pmatrix} x \\ v \end{pmatrix}$$

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

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

$$\dot{X} = \begin{pmatrix} v \\ \frac{1}{2}\omega q \\ F / m \\ (\tau - \omega^{\times} I \omega) / I \end{pmatrix}$$

Rigid body simulation loop

First update the linear and angular velocity

$$v_{t+h} = v_t + hFm^{-1}$$

$$\omega_{t+1} = \omega_t + h\tau I^{-1}$$

Then the position

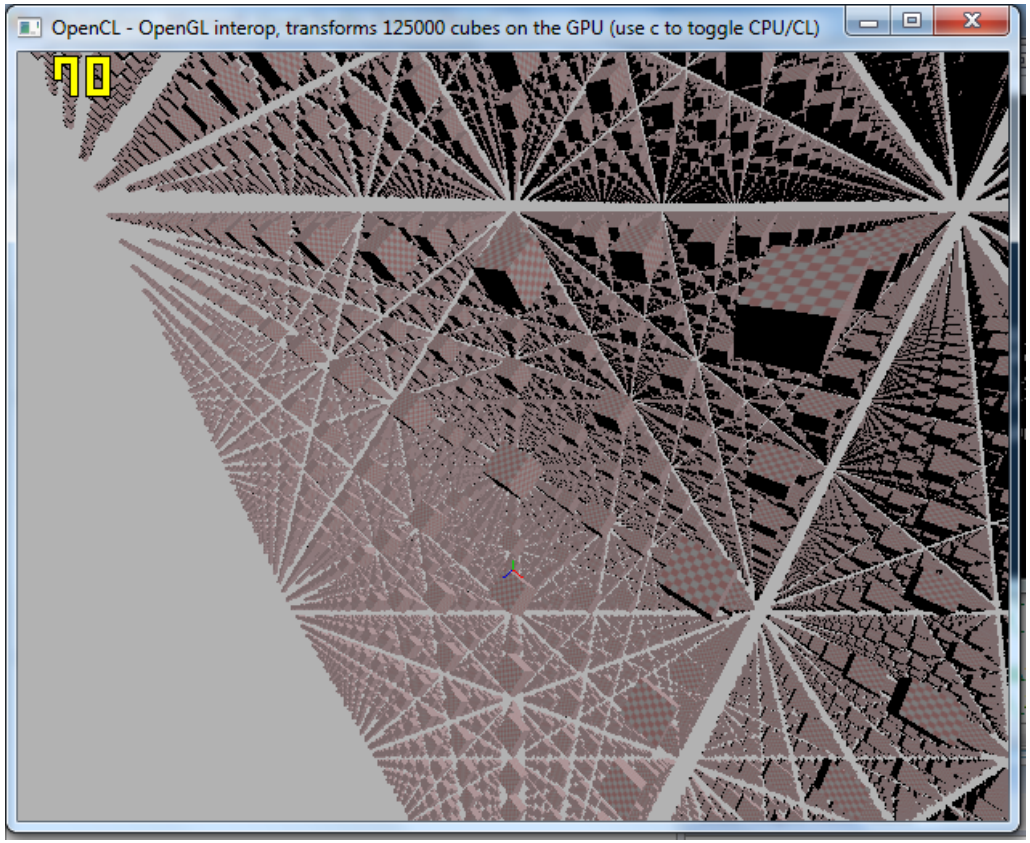
$$x_{t+h} = x_t + h\,v_{t+h}$$

$$q_{t+h} = q_t + h\frac{1}{2}\omega_{t+h}q_t$$

OpenCL kernel: Integration

```
__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



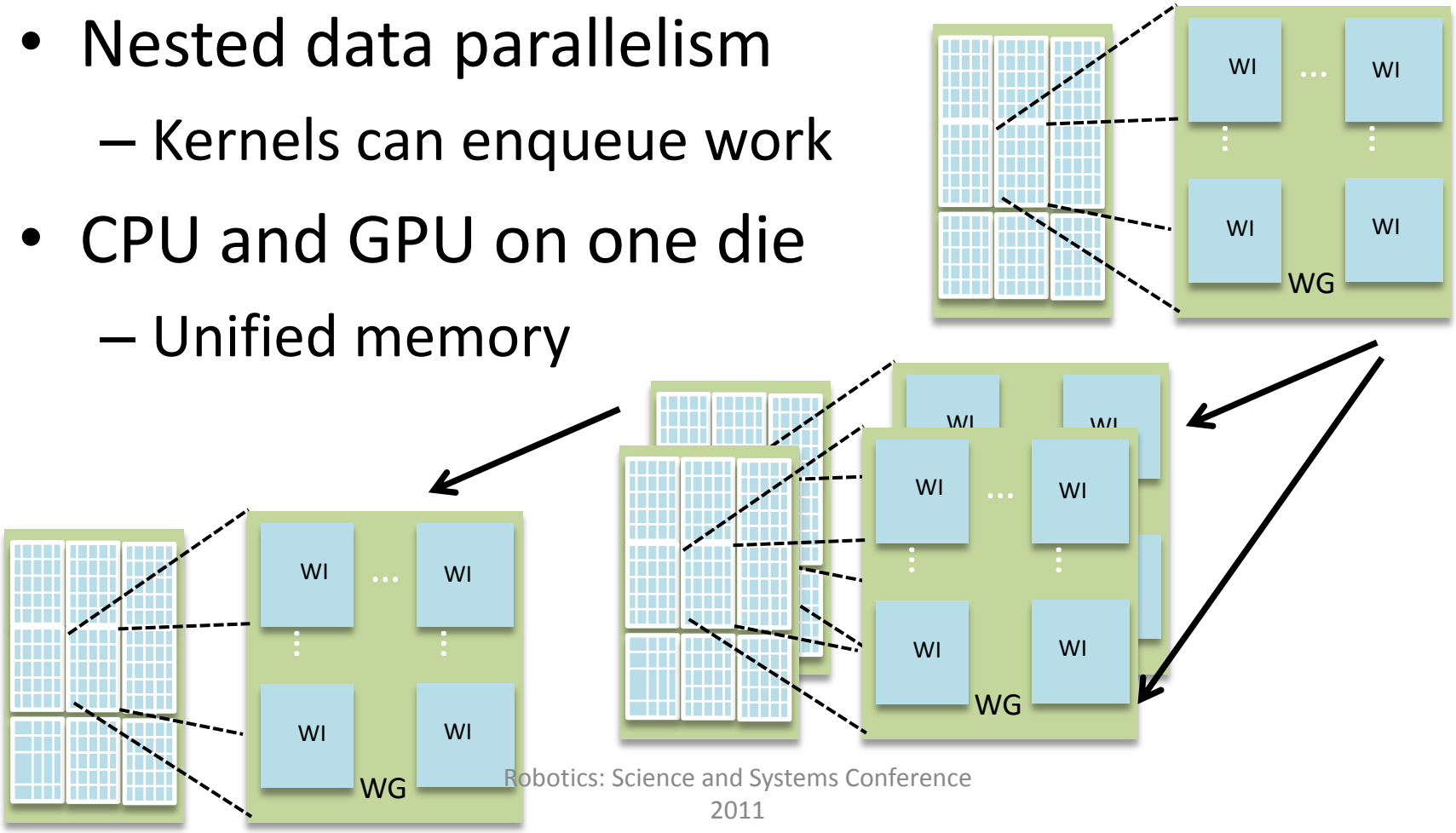
- See http://github.com/erwincoumans/experiments

Today's execution model

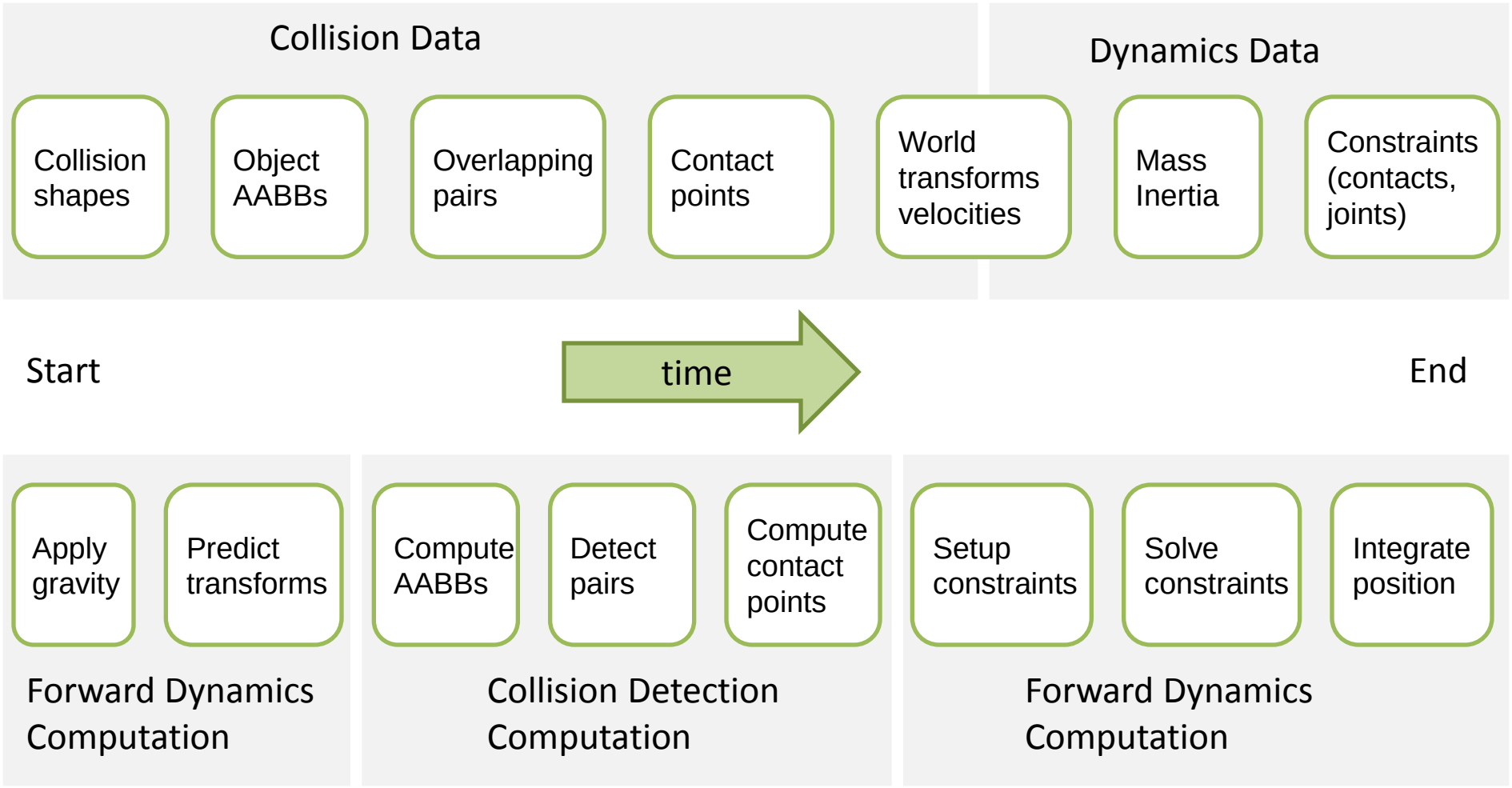
- Single Program Multiple Data (SPMD)
- Same kernel runs on:
  - All compute units
  - All processing elements
- Purely "data parallel" mode
- PCIe bus between CPU and GPU is a bottleneck

Tomorrow's execution model

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



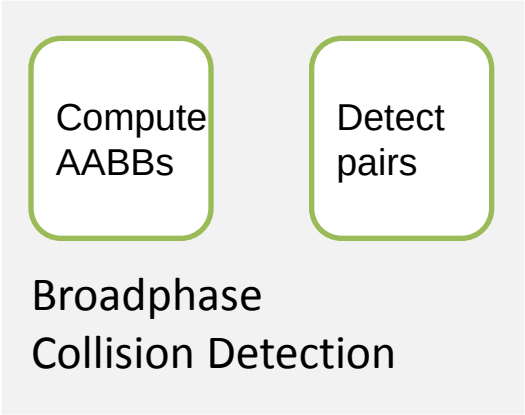
Physics Pipeline



AABB = axis aligned bounding box

Broadphase N-body problem

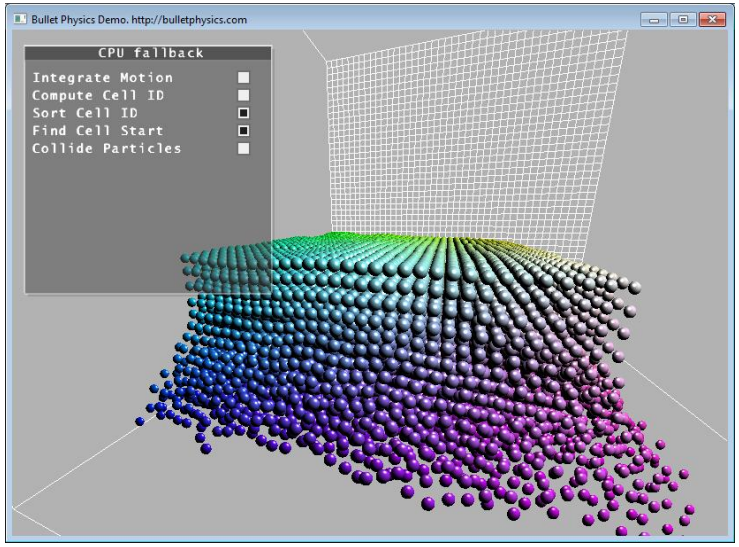
- Avoid brute-force N*N tests



- Input: world space BVs and unique IDs
- Output: array of potential overlapping pairs

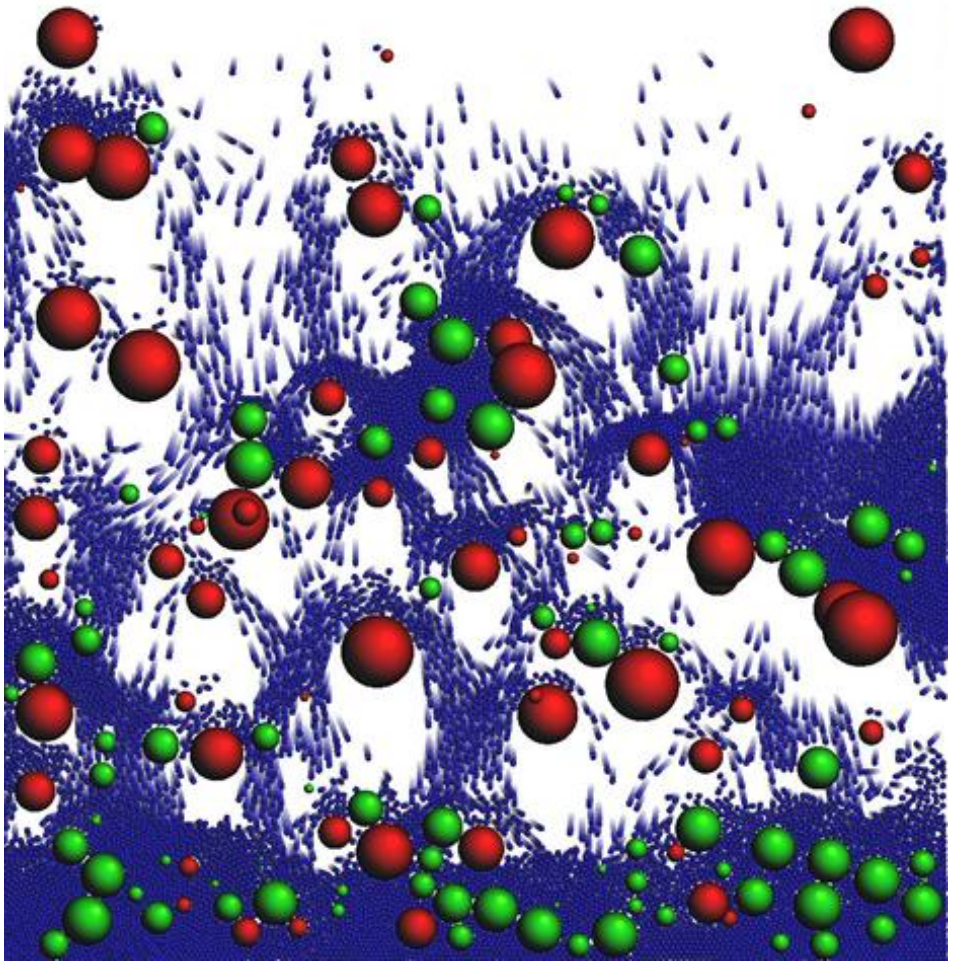
Uniform grid

- Very GPU friendly, parallel radix sort
- Use modulo to make grid unbounded



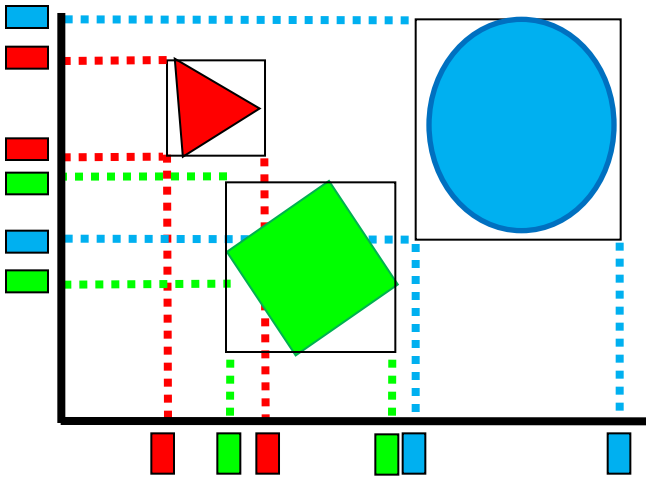
Non uniform work granularity

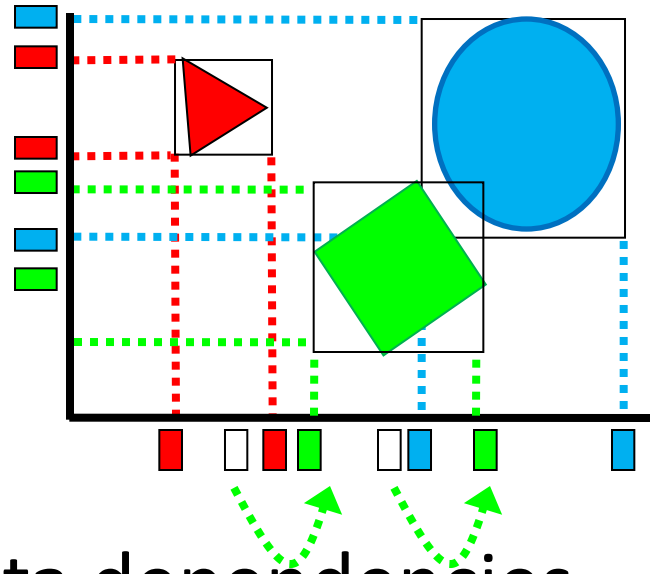
- Small versus small
  - GPU
- Large versus small
  - CPU
- Large versus large
  - CPU



Incremental sweep and prune

- Update 3 sorted axis and overlapping pairs

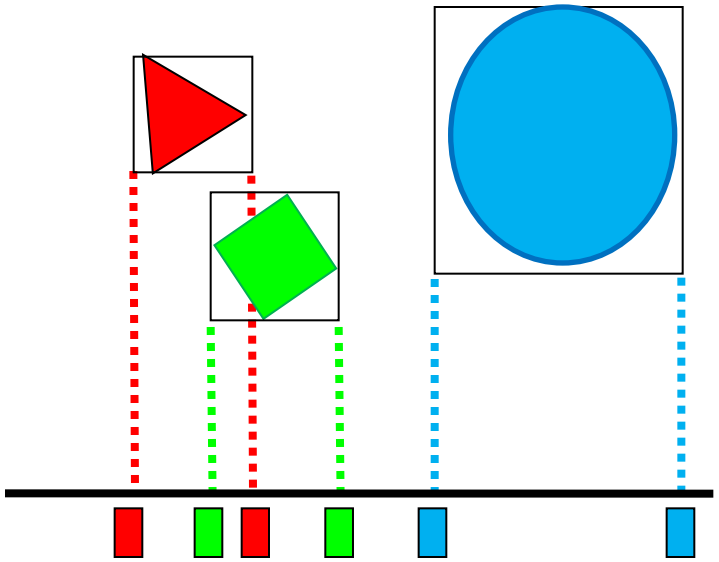




- Does’t parallelize easy: data dependencies

Parallel 1 axis sweep and prune

- From scratch sort 1 axis sweep to find all pairs

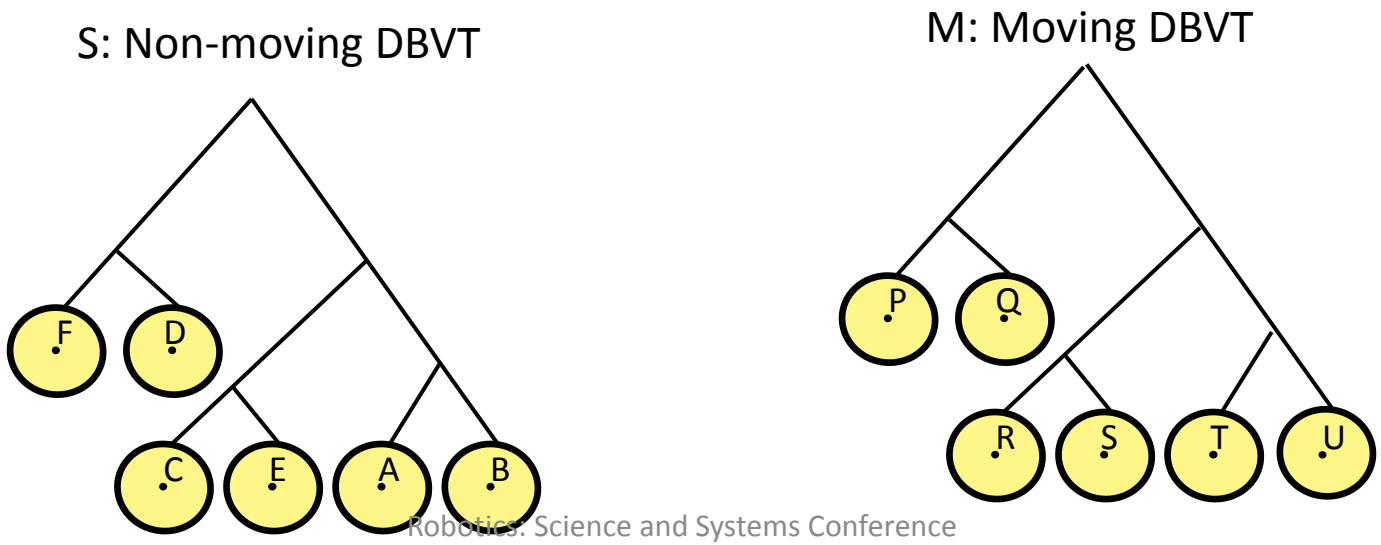


- Parallel radix sort and parallel sweep

[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

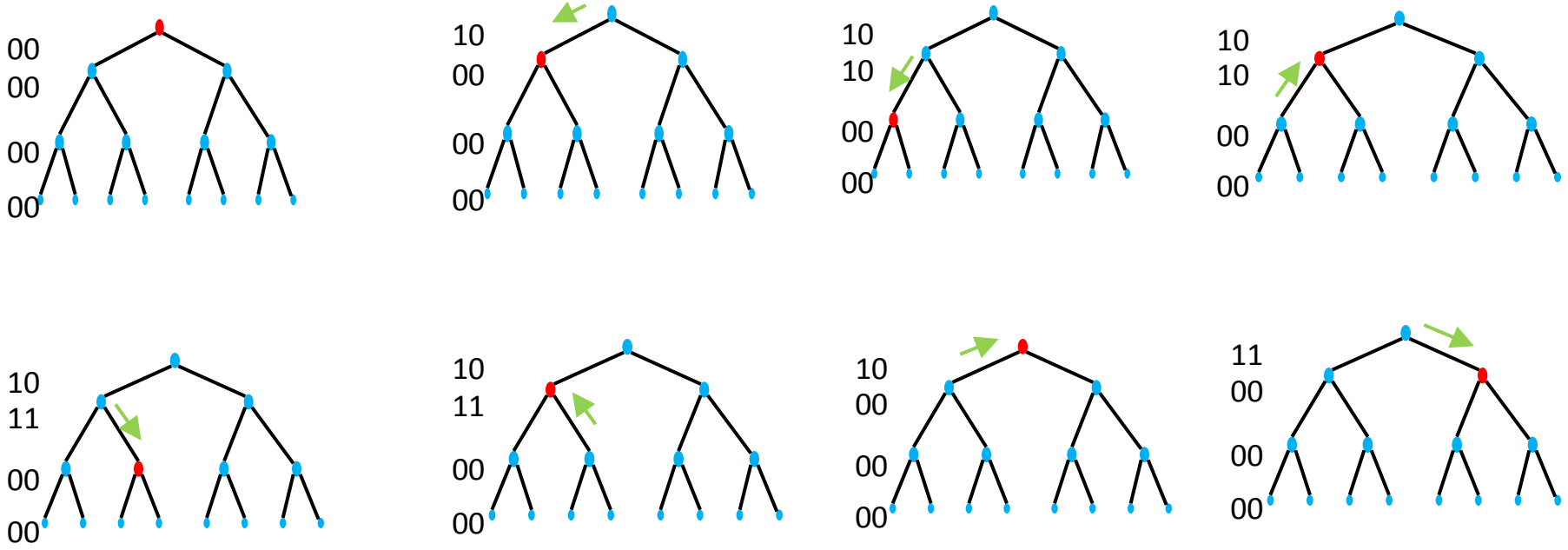


Update/move a leaf node

- 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

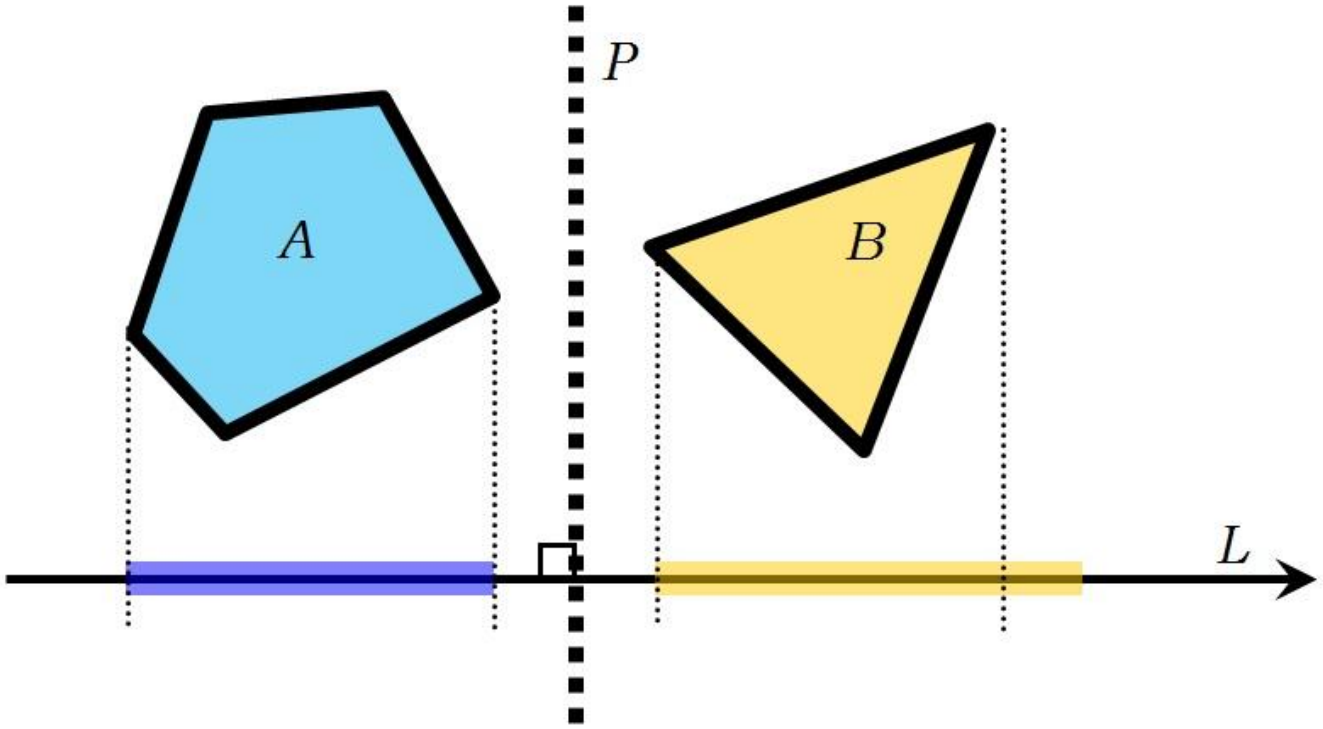
- Input: overlapping pairs, collision shapes
- Output: contact points

Closest point algorithms

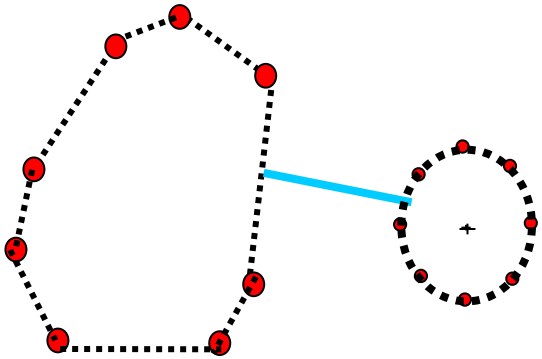
- Algebraic closed forms, ie. sphere-sphere
- Separating axis theorem for convex polyhedra
  - Only computes overlap, no positive distances
- Gilbert Johnson Kheerthi (GJK) for general convex objects
  - Doesn't compute overlap, needs a companion algorithm for penetrating case, for example the Expanding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

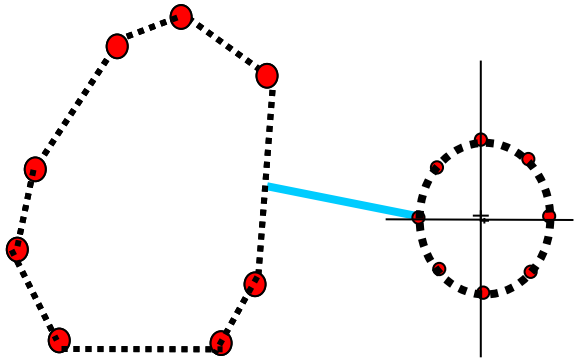
Separating axis test

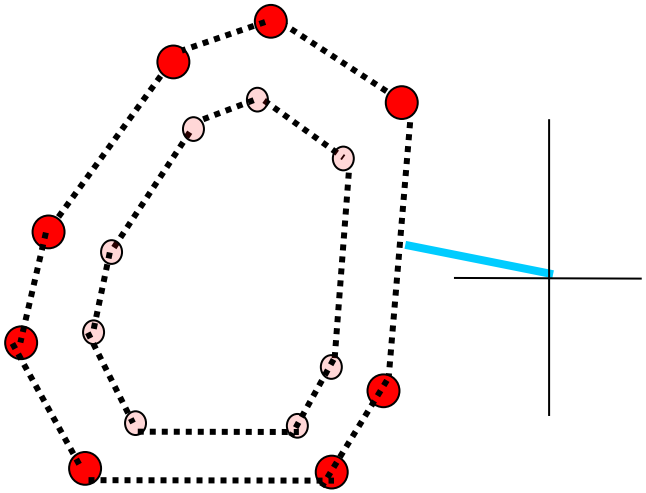
- Test all axis in parallel, use cube map

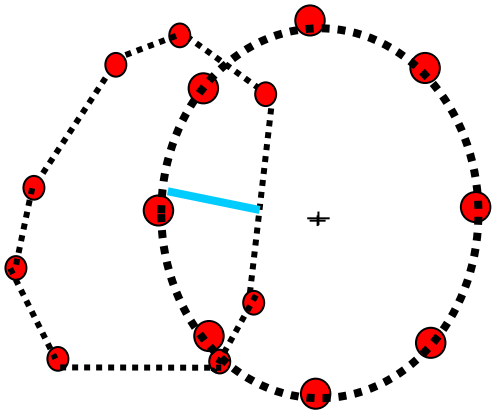


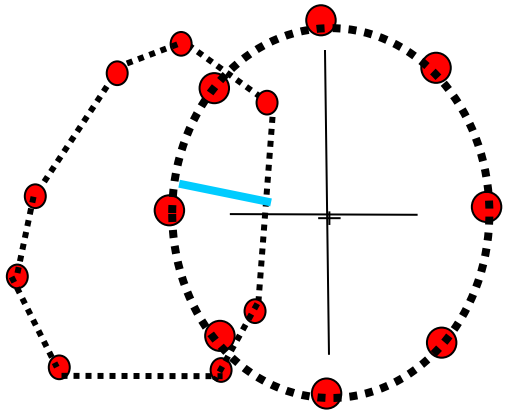
GJK or EPA

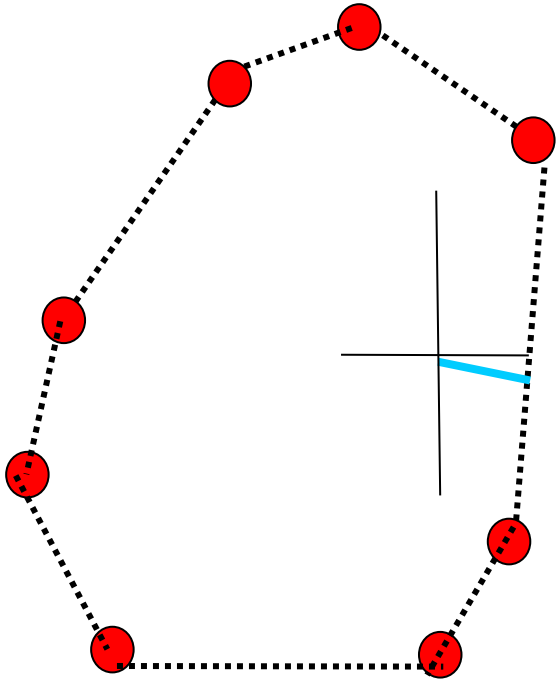






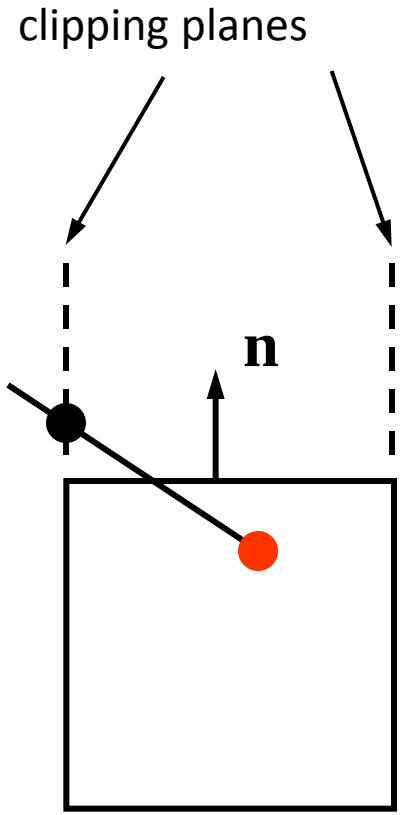






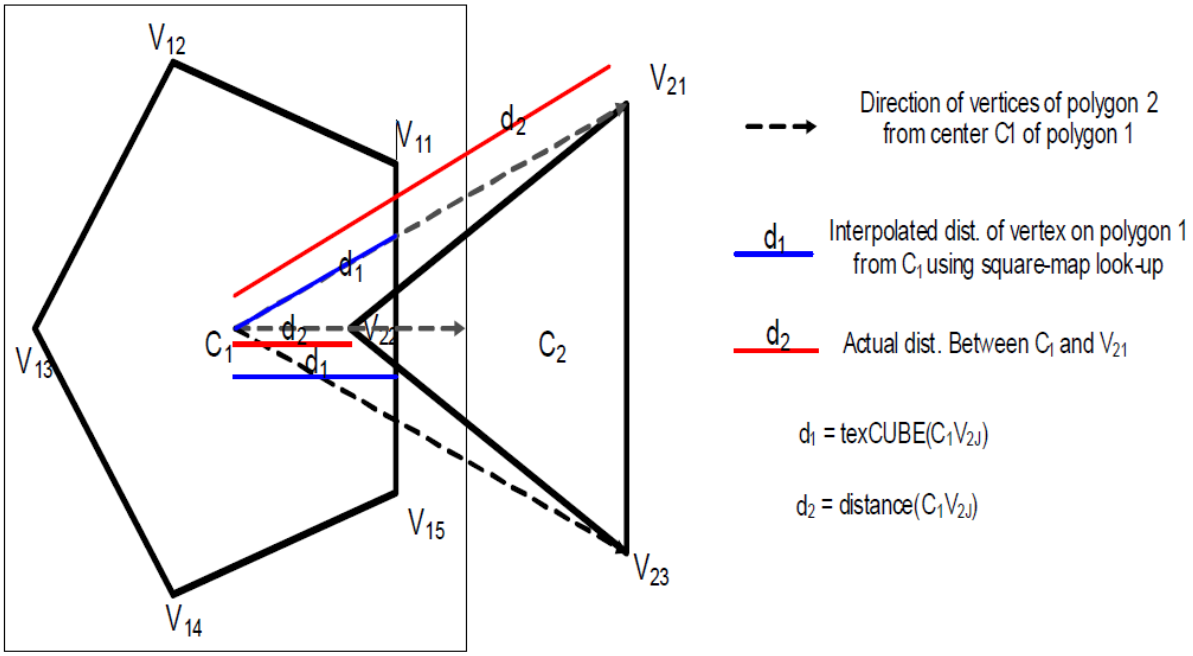
Sutherland–Hodgman clipping

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



Cube map

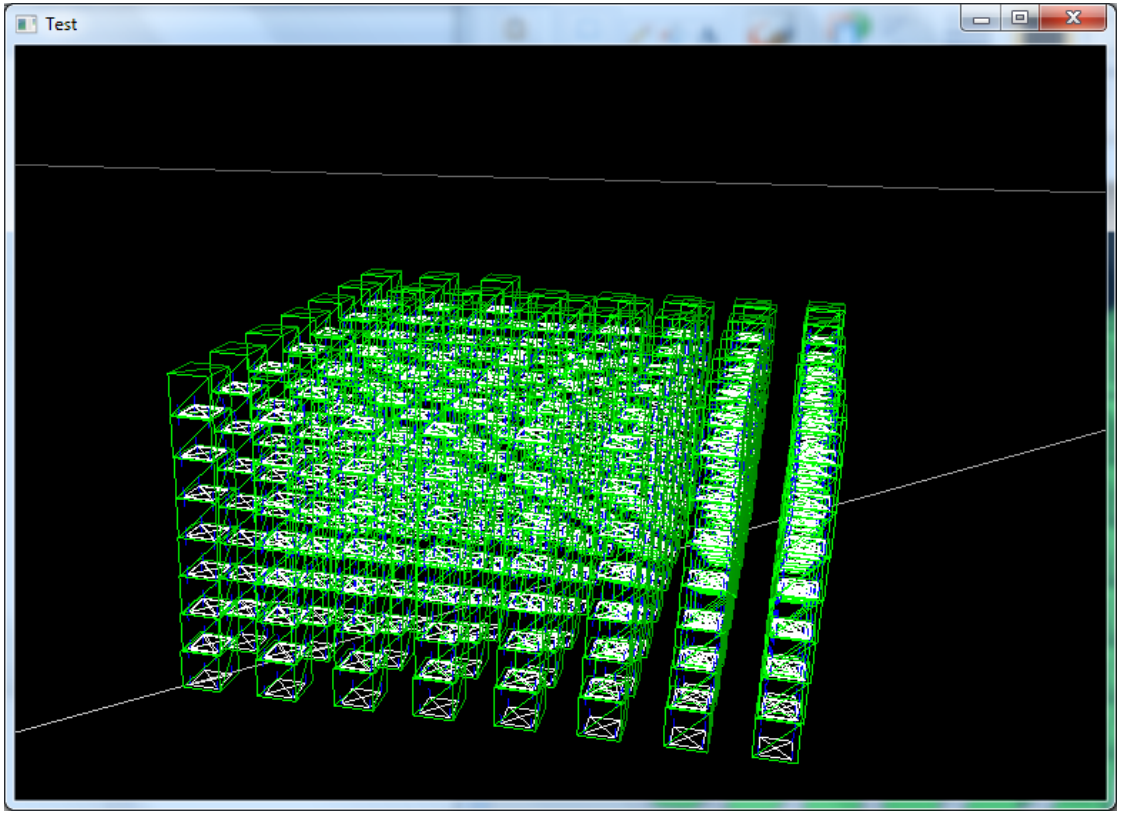
- GPU friendly convex versus convex



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

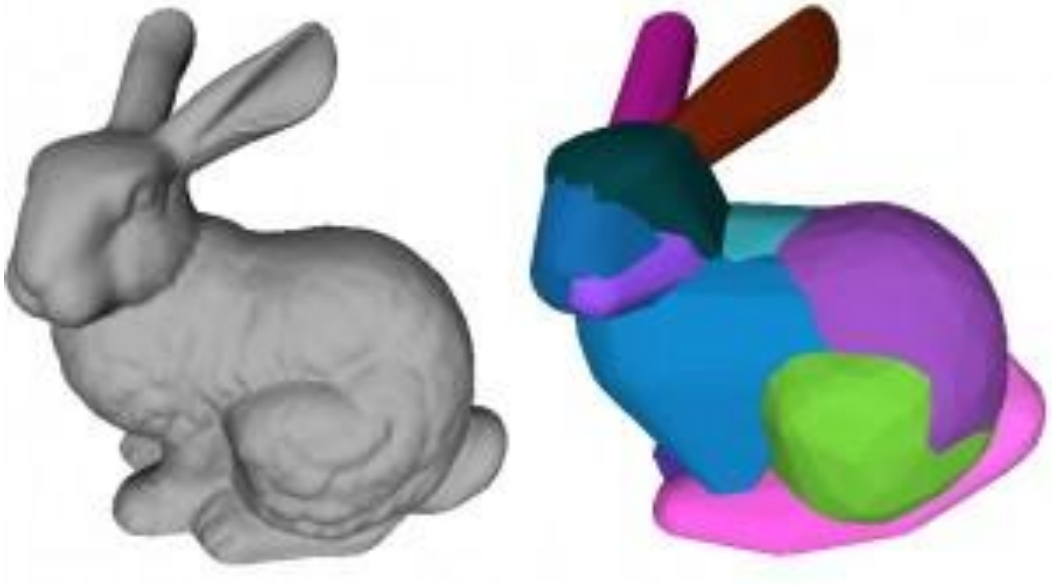
Contact reduction

- Keep only 4 points



Concave shapes

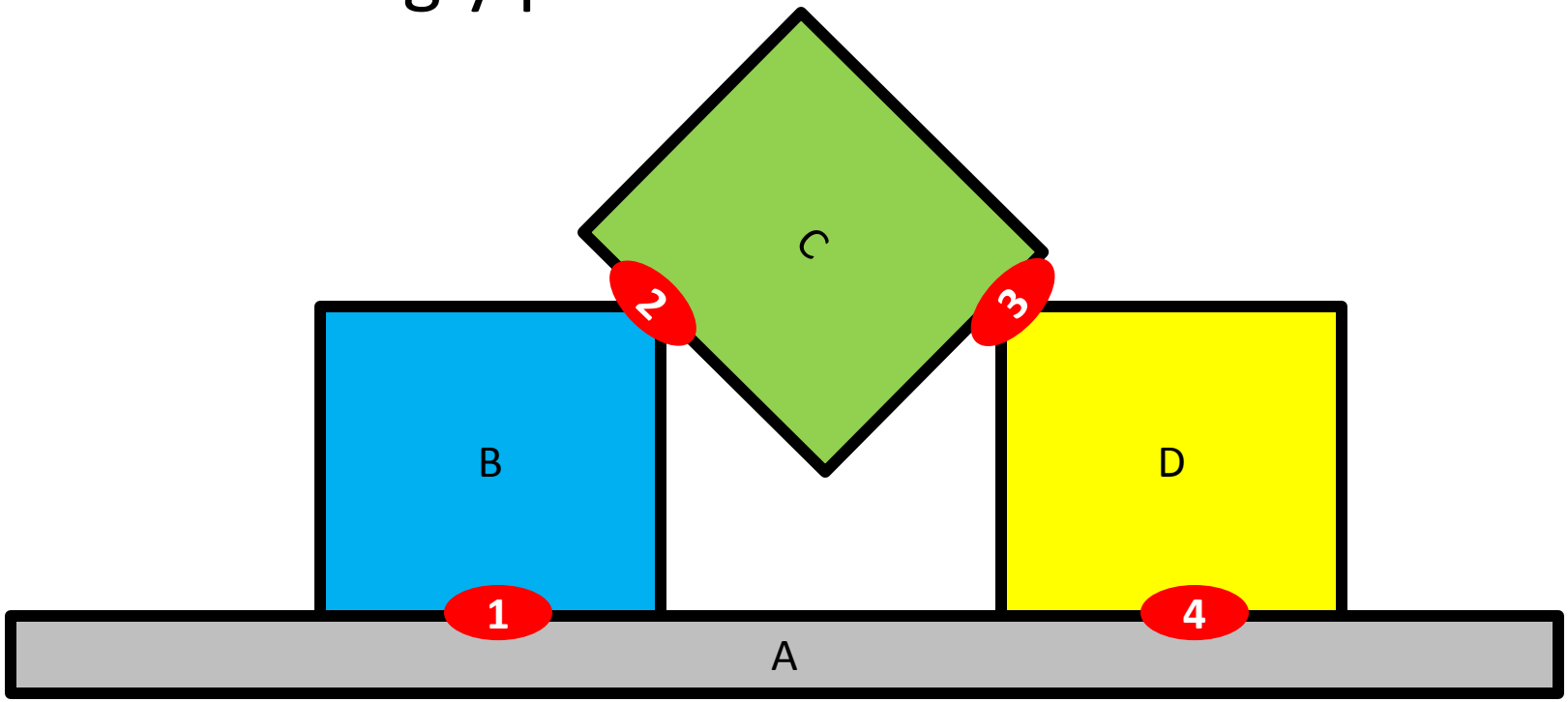
- Hierarchical approximate convex decomposition (ICIP 2009 proceedings. Khaled Mamou)



- http://sourceforge.net/projects/hacd

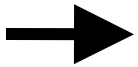
Parallelizing Constraint Solver

- Projected Gauss Seidel iterations are not embarrassingly parallel

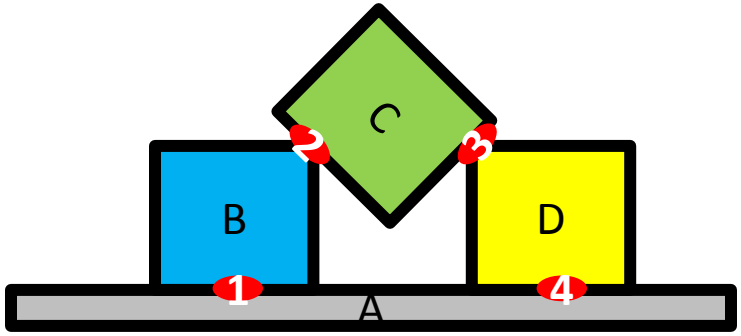


Reordering constraint batches

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



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



Thanks!

- For more information: http://bulletphysics.org
- erwin.coumans@amd.com