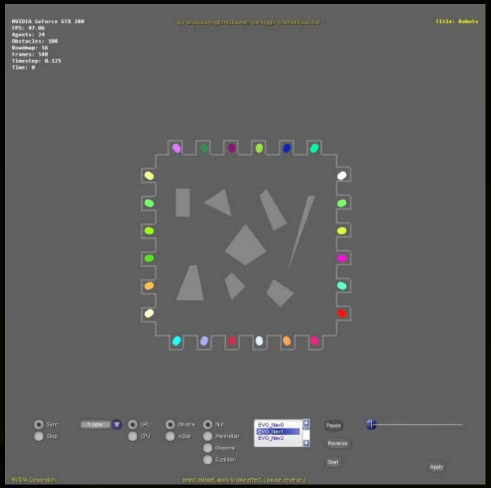
Multi Agent Navigation on GPU

Avi Bleiweiss

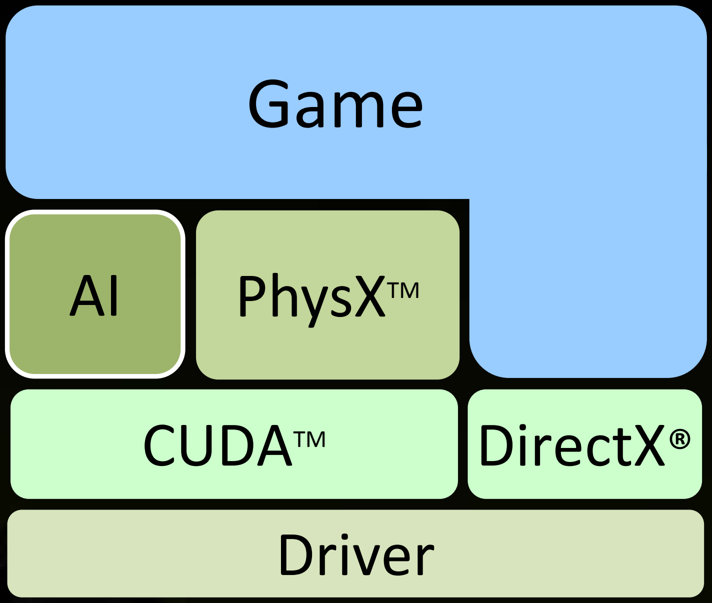
Reasoning

- Explicit
  - Script, storytelling
  - State machine, serial
- Implicit
  - Compute intensive
  - Fits SIMT architecture well
- Navigation planning
  - Collision avoidance



Motivation

- Computational intelligence
  - On CUDA platform
- Alternative pathfinding
  - Intuitive multi threading
  - Flat, nested parallel
- Scalable, real time
  - Dense environments



Problem

Planner

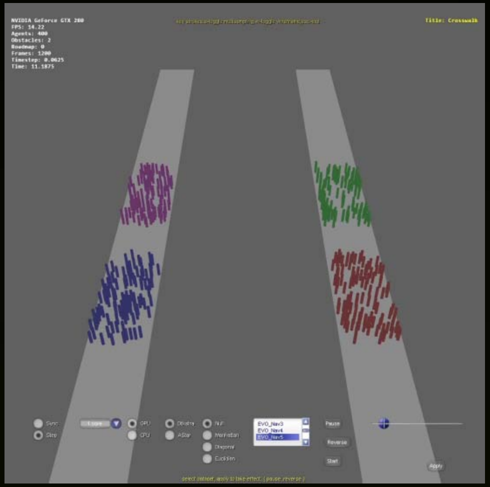
- Searches a global, optimal path
  - From start to goal
- Locally, avoids collisions with
  - Static, dynamic objects
- Exploits autonomous sensing

Simulator

- Visually compelling motion
- Economical memory footprint
- A subset of compute units
- Linear scale with # characters

Solution

- Multi agent model
- Pre-computed roadmap
- Extended Velocity Obstacles
  - Global path integration
  - No explicit communication
- GPU specific optimization
  - Nearest neighbors search



Outline

- Algorithm
- Implementation
- Results
- Takeaways

Paper: Bleiweiss, A. 2009. Multi Agent Navigation on GPU

Algorithm

Visibility

- Two sets of edges
  - Visible roadmap node pairs
  - Goal to unblocked nodes
- A* search, shortest path
  - From goal to any node
- Line segment obstacles
  - Efficient sweep line method

A point is visible from another point -

If the connecting line doesn't intersect any static obstacles.

Velocity Obstacles

- Avoidance velocity set for
  - Dynamic agents among
  - Passively moving obstacles
- Prone to oscillations
- Reciprocal Velocity Obstacles
  - Identical, collision free mind
- Complement set
  - Admissible agent velocities

Velocity Obstacles:
[Fiorini and Shiller 1998]

$$VO_B^A(v_B) = \{ v_A \mid \lambda(p_A, v_A - v_B) \cap B \oplus -A \neq \emptyset \}$$

Reciprocal Velocity Obstacles:
[Van Den Berg et al. 2008]

$$RVO_B^A(v_B, v_A) = \{ v'_A \mid 2 v'_A - v_A \in VO_B^A(v_B) \}$$

Simulation

- Simulator advances until
  - All agents reached goal
- Path realigned towards
  - Roadmap node or goal
- Agent, velocity parallel

```
VO = velocity obstacle
RVO = reciprocal velocity obstacle
do
    hash
        construct hash table
    simulate
        compute preferred velocity
        compute proximity scope
        foreach velocity sample do
            foreach neighbor do
                if OBSTACLE then VO
                elseif AGENT then RVO
            resolve new velocity
    update
        update position, velocity
        resolve at-goal
while not all-at-goal
```

nested

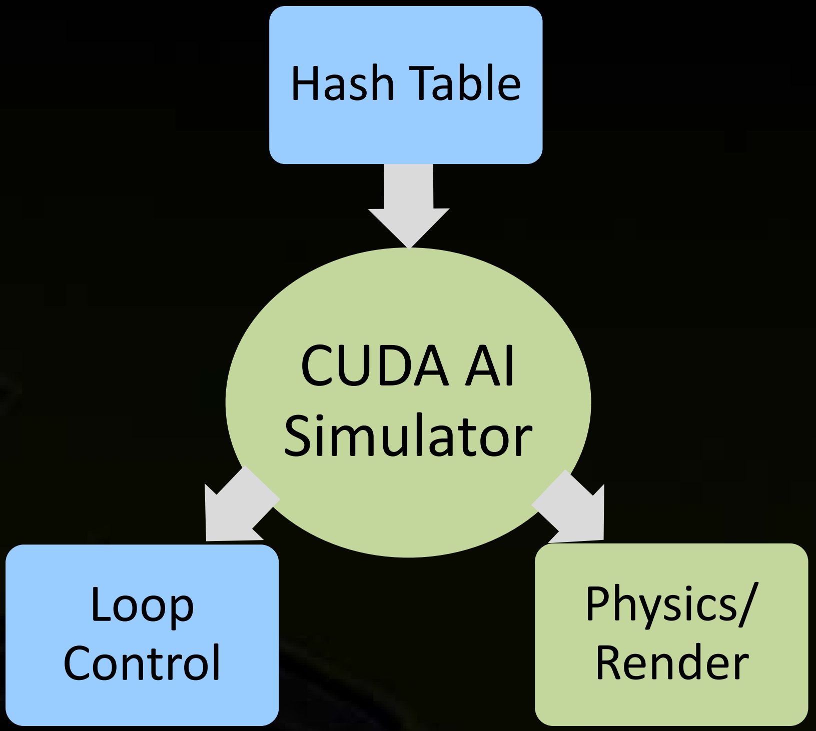
per frame

flat

Implementation

Workflow

- CUDA kernel pair
  - *simulate* and *update*
- Deterministic resources
  - Allocated at initialization
- Per frame output
  - At-goal, path waypoints
- Split frame, multi GPU
  - Device-to-device copy



Challenges

Hiding memory latency

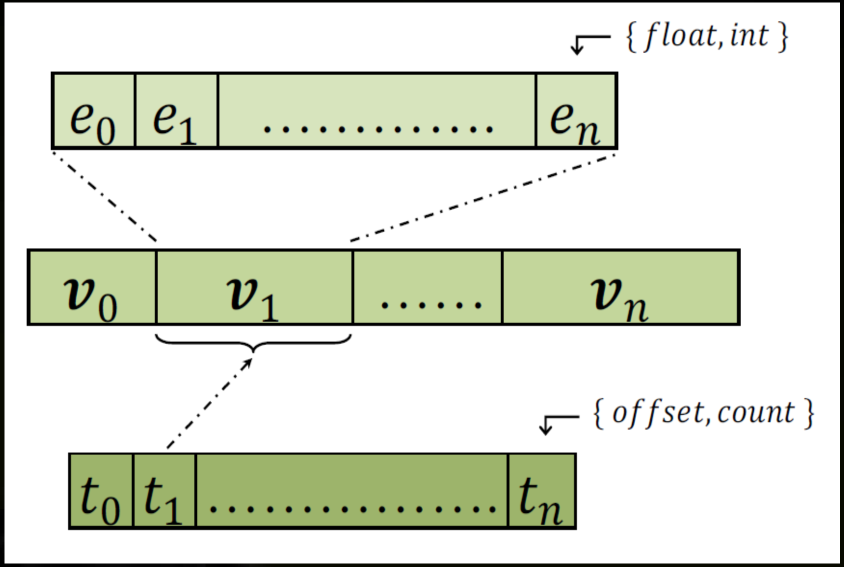
Divergent threads

Hash construction cost

Thread safe RNG

Data Layout

- Persistent resources
  - Reside in global memory
- Static, read-only data
  - Texture bound, linear
- Thread aligned data
  - Better coalescing
- Consistent access pattern
  - Improves bandwidth



Nearest Neighbors Search

- Naïve, exhaustive scheme
  - $O(n^2)$ total running time
- Spatial hash based
  - 3D point to a 1D index
  - Signed distance rule
- Logarithmic traversal time
- Per frame construction
  - Current agent's position

For each agent:

- Select random, 3D position samples

For each sample:

- Hash position
- Compute distance
- Insert, sort distance

Execution Model

- 1D grids and blocks
- Static shared memory
- Hide ALU ops latency
  - 10–12 cycles FMA
- Lessen memory latency
  - Independent math ops
- Per agent RNG

Kernel	Registers	Shared (B)	Local (B)	Constant (B)
simulate	32	116	244	208
update	14	60	0	56

Property	Kernel	
	simulate	update
Threads / Block	128	128
Warps / Multiprocessor	16	32
Occupancy	50%	100%

Nested Parallel

- Flat parallel limited
  - Nested more scalable
- Thread grid hierarchy
  - Independent child grids
  - All running same kernel
  - Grid global atomic sync
- Threads exceed HW max
  - No added memory

```
__global__ void
candidate(CUAgent* agents, int index,
              CUNeighbor* neighbors)
{
   float3 v, float t;
   CUAgent a = agents[index];

   if(!getThreadId()) v = a.prefvelocity;
   else v = velocitySample(a);
   t = neighbor(a, agents, neighbors, v);

   float p = penalty(a, v, t);
   atomicMin(a.minpenalty, p);
   if(p == a.minpenalty)  a.candidate = v;
}
```

Results

Methodology

Environment

- Vista 32 bits, CUDA 2.1
- Simulation-only
- Flat parallel
- Copy to/from device included

Property	GTX280	X7350
Vendor	NVIDIA	Intel
Core Clock (MHz)	601	2930
Memory Clock (MHz)	1107	1066
Global Memory (MB)	1024	8192
Multiprocessors	30	4
Total Threads	500-20000	16

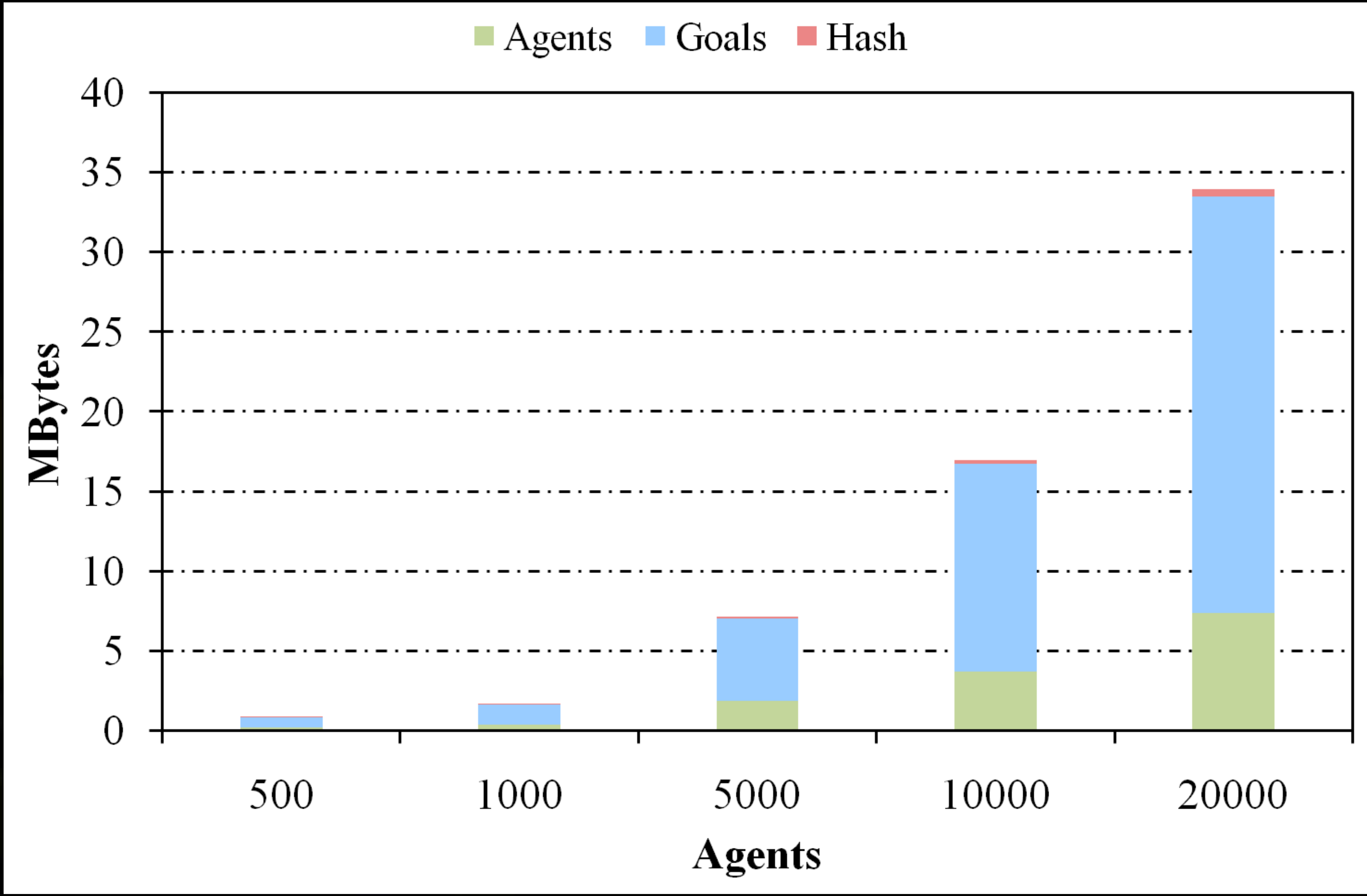
Experiments

Timestep	Proximity		Velocity Samples	Frames
	Neighbors	Distance		
0.1	10	15	250	1200

Dataset	Agents	Thread Blocks
Evacuation	500	4
	1000	8
	5000	40
	10000	79
	20000	157

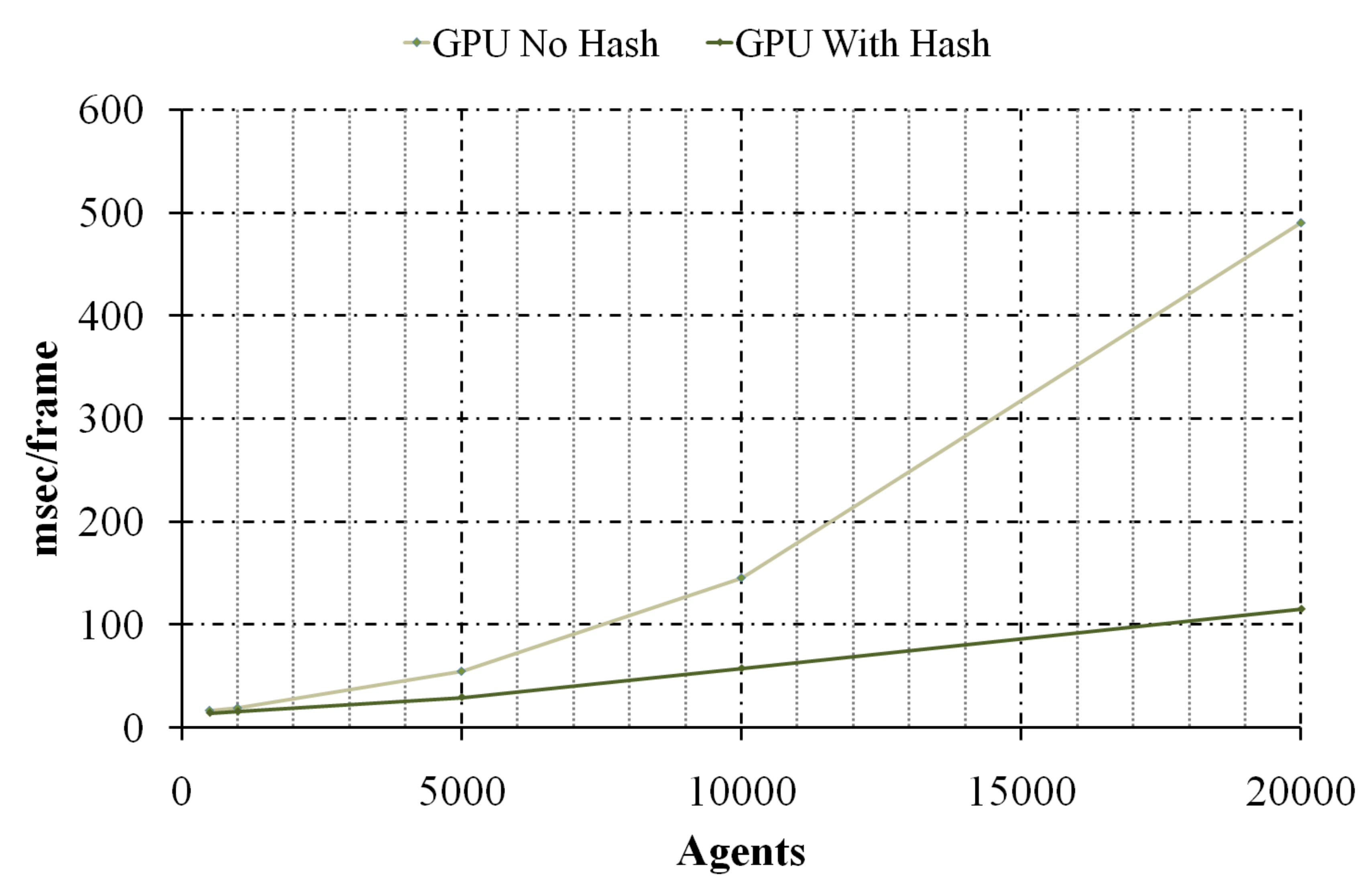
Roadmap:
211 segments
429 nodes

Footprint

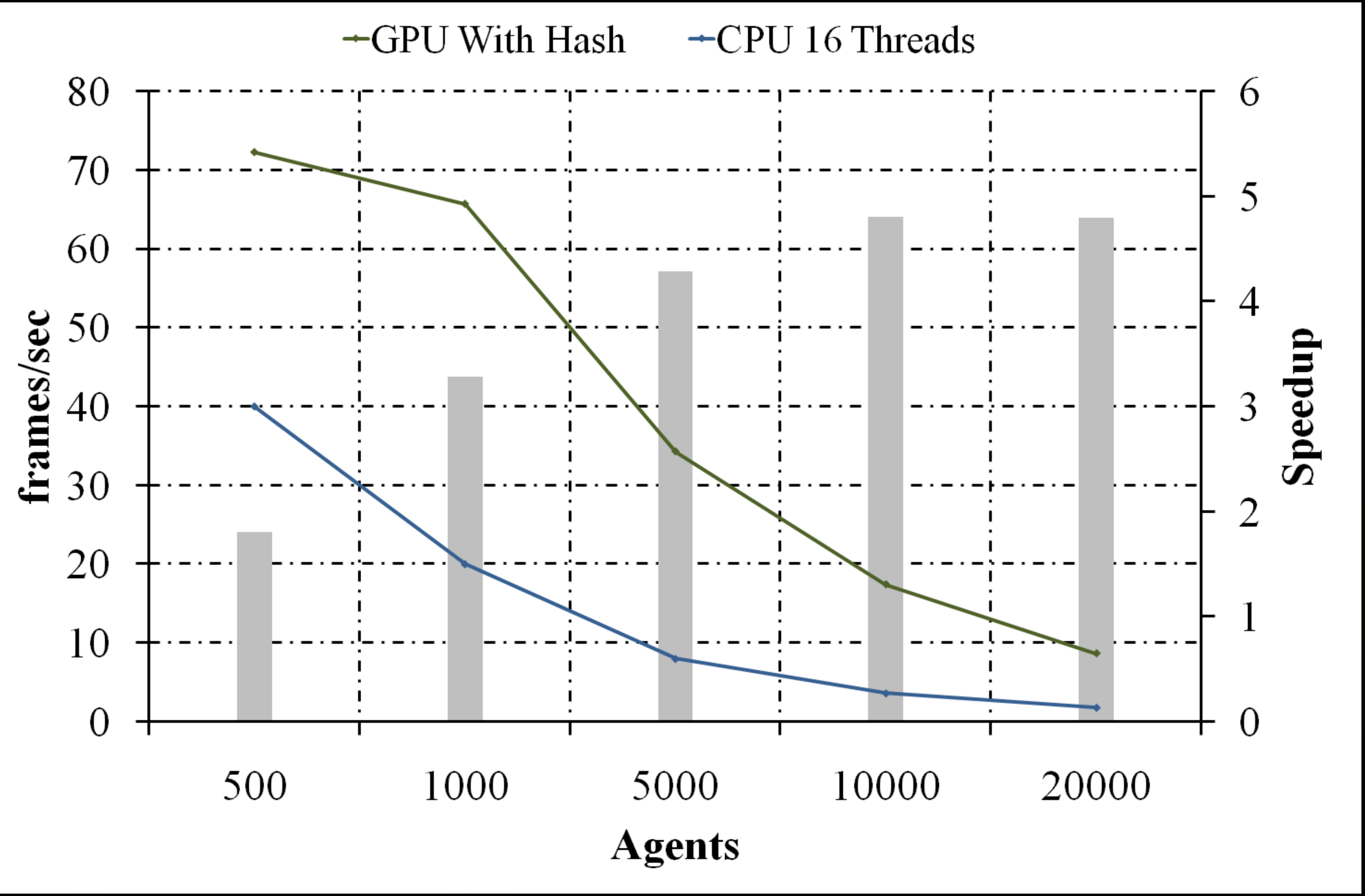


Agents
Goals
Hash
40
35
30
25
20
15
10
5
0
MBytes
500
1000
5000
10000
20000
Agents

Running Time



Frame Rate



GPU With Hash
CPU 16 Threads
frames/sec
80
70
60
50
40
30
20
10
0
Speedup
6
5
4
3
2
1
0
500
1000
5000
10000
20000
Agents

Takeaways

Limitations

- Hash table construction
  - Single threaded
- Thread load imbalance
  - Non, at-goal agent mix
- Hash motion artifacts
  - Area under sampling
- Shared memory SW cache
  - Constraint, 32B per thread

Future Work

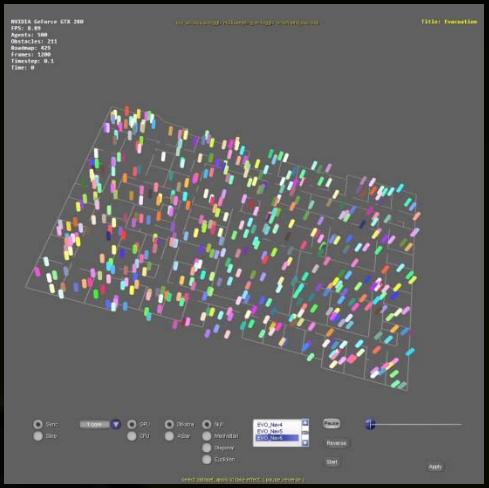
- Exploit shared memory
  - Further hide latency
- At-goal agent extraction
  - Unified thread block
- Up hash sampling quality
- Dynamic obstacles, goals
  - GPU visibility port

Performance

Parameter	NVIDIA GTX280	INTEL X7350
Hash Speedup	4X	Little to None
Simulation Acceleration	Up to 77X	Single thread
	Up to 4.8X	Sixteen threads
FPS (for 10K agents)	18	3.75
Nested vs. Flat	Up to 2X	Difficult to program
Cost ($)	399	2400

Summary

- Computational intelligence
  - Maps well on GPU
- Multi agent solution
  - Compact, scalable
  - Further optimization
- Nested data parallel
  - Multi GPU system
- AI, physics integration



Questions?

Thank You!

How To Reach Us

- Paper:
  - http://tinyurl.com/MultiAgentGPU-paper-2009
- During GDC
  - Expo Suite 656, West Hall
  - Developer Tool Open Chat, 1:30 to 2:30 pm (25th-27th)
- Online
  - Twitter: nvidiadeveloper
  - Website: http://developer.nvidia.com
  - CUDA: http://www.nvidia.com/cuda
  - Forums: http://developer.nvidia.com/forums