Game Physics on the GPU with PhysX 3.4

Kier Storey
PhysX 3.4 Features

• GPU Rigid Bodies
• Improved threading and performance
• New CCD mode
• Low-level immediate mode
• Enhanced Determinism
• Faster, more robust convex hull cooking
• Faster mid-phase structure
• Serializable scene query trees for level streaming
• Split-sim
• Improved vehicles
Basic PhysX Rigid Body Pipeline

- Broad Phase
- Pair Filtering
- Narrow Phase
- Island Gen
- Solver
- CCD (Optional)
- Fetch Results
Pipeline Stages

- **Broad Phase**
  - Produces set of candidate pairs that are potentially interacting
  - Quickly rejects non-overlapping pairs
  - Uses approximate bounds (e.g. AABBs or spheres)

- **Pair Filtering**
  - Apply application-rules to permit/disallow pairs to be processed by narrow phase or solver

- **Narrow Phase/Contact gen**
  - Processes the set of pairs produced by broad phase
  - Determines if the geometries are actually interacting, in which case generates contacts.
Main Pipeline stages cont.

• Island Management
  • Groups bodies into islands
  • Island = collection of bodies interacting via contacts or constraints
  • A given object can be a member of only 1 island unless that body is static or kinematic

• Constraint Solver
  • Solves islands
  • Produces constraints from the set of contacts and joints
  • Computes new velocities and transform for rigid bodies that satisfy constraints.

• Fetch Results
  • Buffering
  • Fire user callbacks
  • Update Scene Query structures
GPU vs CPU

• GPU
  • Massive FLOPS and memory bandwidth.
  • 1000s of compute cores
  • Lower clock frequencies
  • Longer-latency instruction pipeline
  • Highly-sensitive to memory access patterns and branching
  • Algorithms must scale to 1000s of threads.

• CPU
  • Lower FLOPS and memory bandwidth
  • Small number of cores
  • Higher clock frequencies
  • Lower-latency instruction pipeline
  • Tolerant to memory access patterns branching
  • Executes sequential and parallel algorithms well
GPU Rigid Body Goals

• Easy to integrate
• Same semantics and behavior as CPU PhysX
• Support full PhysX feature-set
• Must be fast!
• Minimize latency to access results
• Gameplay-effecting simulation

Plan:
  • Port broad phase, narrow phase and solver to GPU
  • Leave rest of pipeline on CPU
Potential performance gains?

• Moving pipeline stages from CPU to GPU can yield significant performance gains

• It can also introduce additional overhead
  • Memory transfer
  • Kernel dispatch overhead

• Amdahl’s Law applies
  • The serial stages of the pipeline will become a bottleneck as the number of cores processing the parallel stages increases
A PhysX 3.3 CPU simulation frame

PhysX 3.3 per-stage breakdown (13824 convexes)

- Broad Phase: 26%
- Narrow Phase: 30%
- Island Gen: 11%
- Post Broad Phase: 5%
- Fetch Results: 6%
- Update AABBs: 0%
- Solver: 22%
What if GPU stages were faster?

100x Faster
10x Faster
5x Faster
Orig

- Update AABBs
- Broad Phase
- Post Broad Phase
- Narrow Phase
- Island Gen
- Solver
- Fetch Results
Performance in PhysX 3.3

• Broad Phase, narrow phase and solver ~70-80% of total simulation time
  • Meaning maximum speed-up is limited to 3.3-5x
• Not enough!
• Serial stages of pipeline quickly become bottleneck!
• Either migrate more to GPU or optimize CPU code
An Improved Physics Pipeline!

• PhysX 3.3 pipeline too serial
• New pipeline parallelizes more stages
• Optimized parallel interaction framework to scale to 1m+ pairs
• New incremental island management
• New sim controller and AABB manager
  • Shares common information between broad phase, narrow phase and scene query to avoid redundant work
• Optimized CPU contact generation and constraint solver
An Improved Physics Pipeline!

• Improved memory footprint and cache coherence
• Decouple and overlap pipeline stages so CPU and GPU can both be busy at the same time
  • Also provides better multi-core CPU performance
• New split fetchResults API to enable application to parallelize callbacks
  • Callbacks can potentially become a bottleneck!
• New split sim API
GPU Rigid Bodies in PhysX 3.4

• Hybrid CPU/GPU rigid body simulation

• Execute the following Rigid Body pipeline stages on GPU
  • Broad Phase, Narrow Phase, Solver
  • Miscellaneous state management, bounds computation etc.

• Execute the following stages on the CPU
  • Island Management
  • Shape filtering and interaction management
  • CCD
  • Triggers
  • User callbacks
  • Updating scene query structures
GPU Broad Phase

• Two-phase incremental broad phase algorithm
• Produces only delta pairs
  • New or lost pairs since last time BP was run
  • Significantly reduces data transfer between CPU and GPU
• Highly-scalable
• Often orders of magnitude faster than commonly-used CPU sweep and prune approaches.
• Can be enabled without enabling the rest of the GPU pipeline
• PxAggregates are partially handled on CPU
  • PxAggregate is usually not beneficial if using GPU broad phase
GPU Narrow Phase

• PCM-based
  • Supports boxes, convex hulls, meshes and heightfields
  • Convex hulls must have <= 64 verts and <= 32 verts per-face
  • Meshes and convex hulls need extra cooked data

• CPU processes
  • Incompatible shape pairs (sphere, capsule, plane, complex convex)
  • Pairs with contact modification enabled

• Contacts generated on CPU are automatically transferred to GPU to be processed by the solver
• Contacts generated on GPU are automatically transferred back to CPU as needed

• Trigger pairs are processed on CPU
  • Trigger behaviour can be emulated on GPU using touch found/lost events
GPU Constraint Solver

• Hybrid PGS/MS constraint solver
• Provides equivalent behaviour to PhysX CPU solver
• Extracts and exploits massive levels of parallelism from within islands
• Utilizes an efficient lazy algorithm to determine dependency chains
  • Cost is proportional to how much connectivity changes rather than the complexity of the graph itself
• Solves all contacts and joint constraints
  • Native support for D6 joints (full pipeline executed on GPU)
  • Other joint types have joint shaders execute on CPU and results transferred to GPU for processing
GPU Constraint Solver continued

• Supports most features supported by CPU
  • Force reports and force thresholding
  • Breakable joints
  • Applies all modifiable properties
    • Limiting contact/constraint force, target velocity, max de-penetration velocity, dominance and local mass modifications

• Doesn’t currently support articulations

• Designed to provide good performance while using as few GPU compute resources as possible.
GPU Simulation Controller

- Body and shape state management
- Manages pair and constraint states
- Controls actor sleeping
- Handles user state modifications to actors and pairs
  - Efficiently keeps CPU and GPU view of current body/shape/pair states up-to-date by lazily updating states as required
- Buffers external/internal states to minimize per-frame data transfers between CPU and GPU.
Ease of Integration

Basic scene initialization

```cpp
PxSceneDesc sceneDesc(gPhysics->getTolerancesScale());
sceneDesc.gravity = PxVec3(0.0f, -9.81f, 0.0f);
sceneDesc.cpuDispatcher = PxDefaultCpuDispatcherCreate(4);
sceneDesc.filterShader = PxDefaultSimulationFilterShader;

gScene = gPhysics->createScene(sceneDesc);
```

Basic scene initialization with GPU rigid bodies

```cpp
PxSceneDesc sceneDesc(gPhysics->getTolerancesScale());
sceneDesc.gravity = PxVec3(0.0f, -9.81f, 0.0f);
sceneDesc.cpuDispatcher = PxDefaultCpuDispatcherCreate(4);
sceneDesc.filterShader = PxDefaultSimulationFilterShader;

PxCudaContextManagerDesc cudaContextManagerDesc;
gCudaContextManager = PxCreateCudaContextManager(*gFoundation, cudaContextManagerDesc);

sceneDesc.gpuDispatcher = gCudaContextManager->getGpuDispatcher();
sceneDesc.flags |= PxSceneFlag::eENABLE_GPU_DYNAMICS;
sceneDesc.broadPhaseType = PxBroadPhaseType::eGPU;

gScene = gPhysics->createScene(sceneDesc);
```
Performance Results

• Windows 10 64-bit
• I7-5930k
• 32GB RAM
• GTX 1080
13,824 Convex Objects
Hallway Destruction
Arena Demo Destruction
GRB Demo (Kapla Tower) 20,000 convexes
700 Ragdolls
N convex objects Complexity Scaling

www.gameworks.nvidia.com
N convex objects Complexity Scaling Cont.
Scene Query Performance (Raycast Mesh)
Box Sweep vs Mesh

PhysX 3.3.4

PhysX 3.4
Convex sweep vs Convex

![Graph showing comparison between PhysX 3.3.4 and PhysX 3.4 for convex sweep vs convex.

The graph illustrates the performance difference, with PhysX 3.3.4 having a significantly higher result than PhysX 3.4.]}
Convex Cooking Speed Improvements

- 16 Verts
- 32 Verts
- 64 Verts
- 128 Verts

Comparison between PhysX 3.3 and PhysX 3.4.
Conclusions and Future Work

• PhysX 3.4 - Full CPU source available NOW!

• Significantly faster to PhysX 3.3 across-the-board with lots of cool features
  • If you use PhysX 3.3 - you should upgrade ASAP 😊

• GPU rigid body simulation available on Windows and Linux (Kepler and above)

•GPU rigid body Future work
  • Further performance improvements
  • Improve simulation quality
  • Make feature complete
Questions?