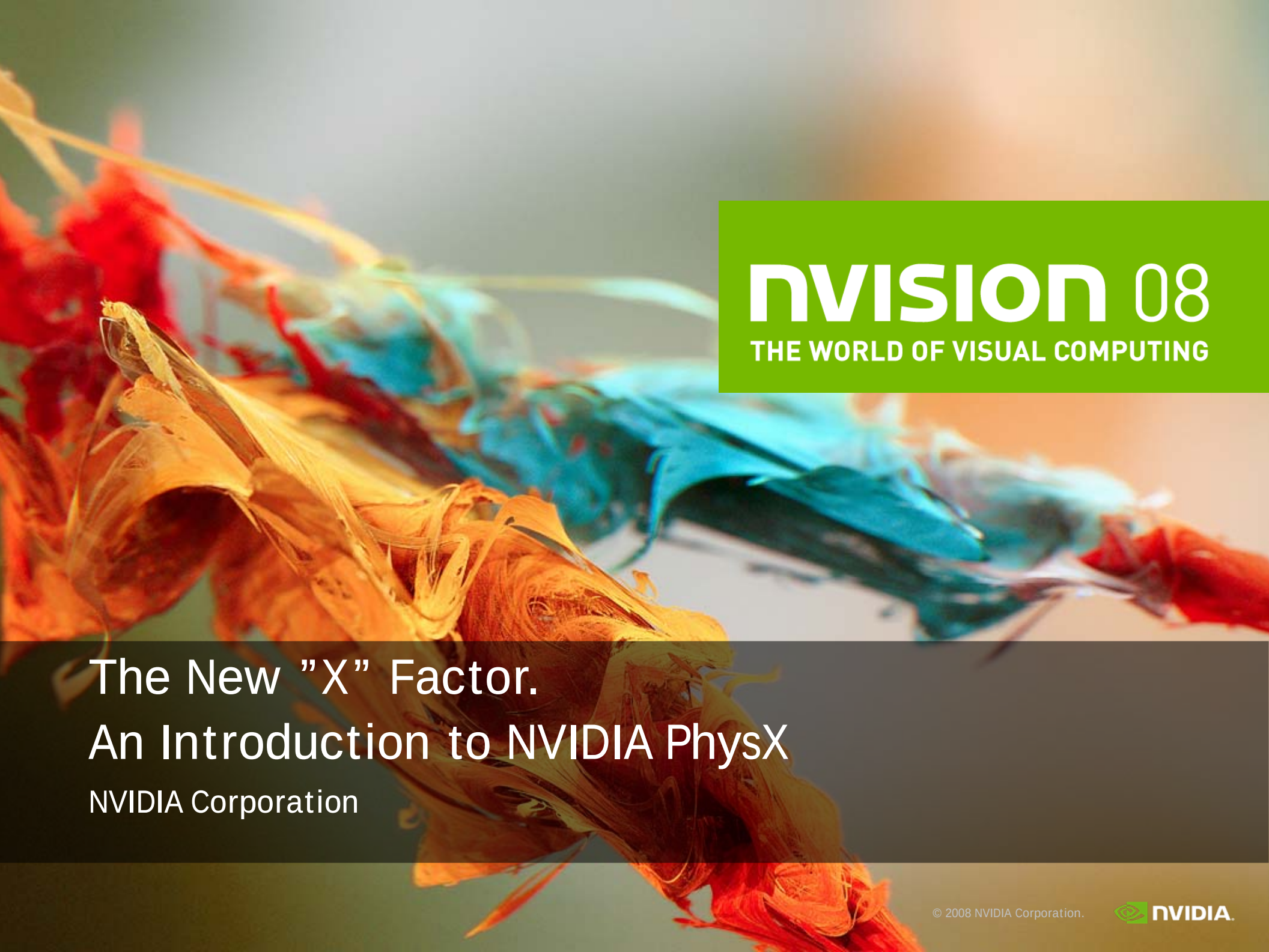
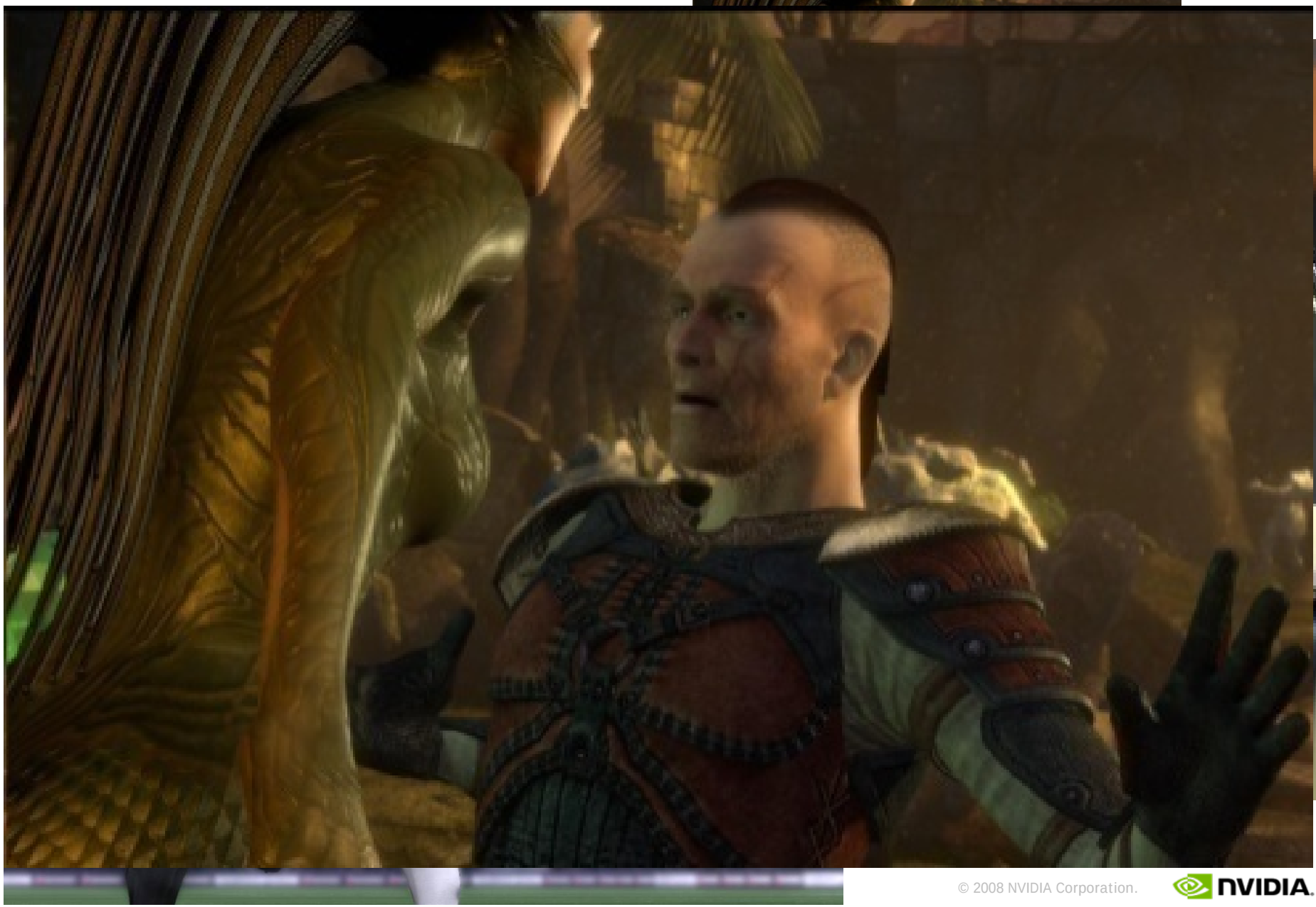


The New "X" Factor.
An Introduction to NVIDIA PhysX

NVIDIA Corporation

NVIDIA

What is Gaming Physics?



What is Gaming Physics?

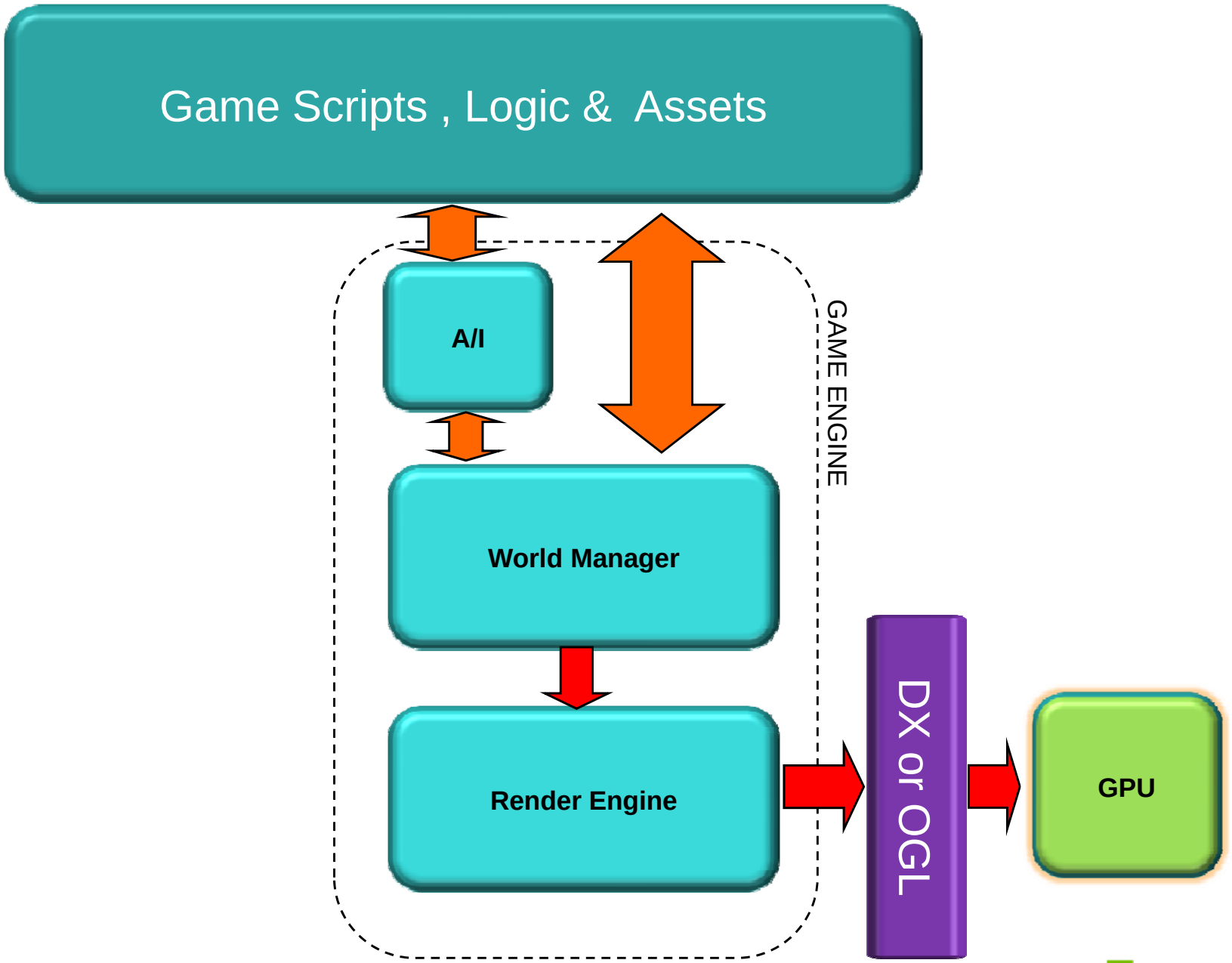
What is Gaming Physics?

PhysX Makes Graphics Come Alive

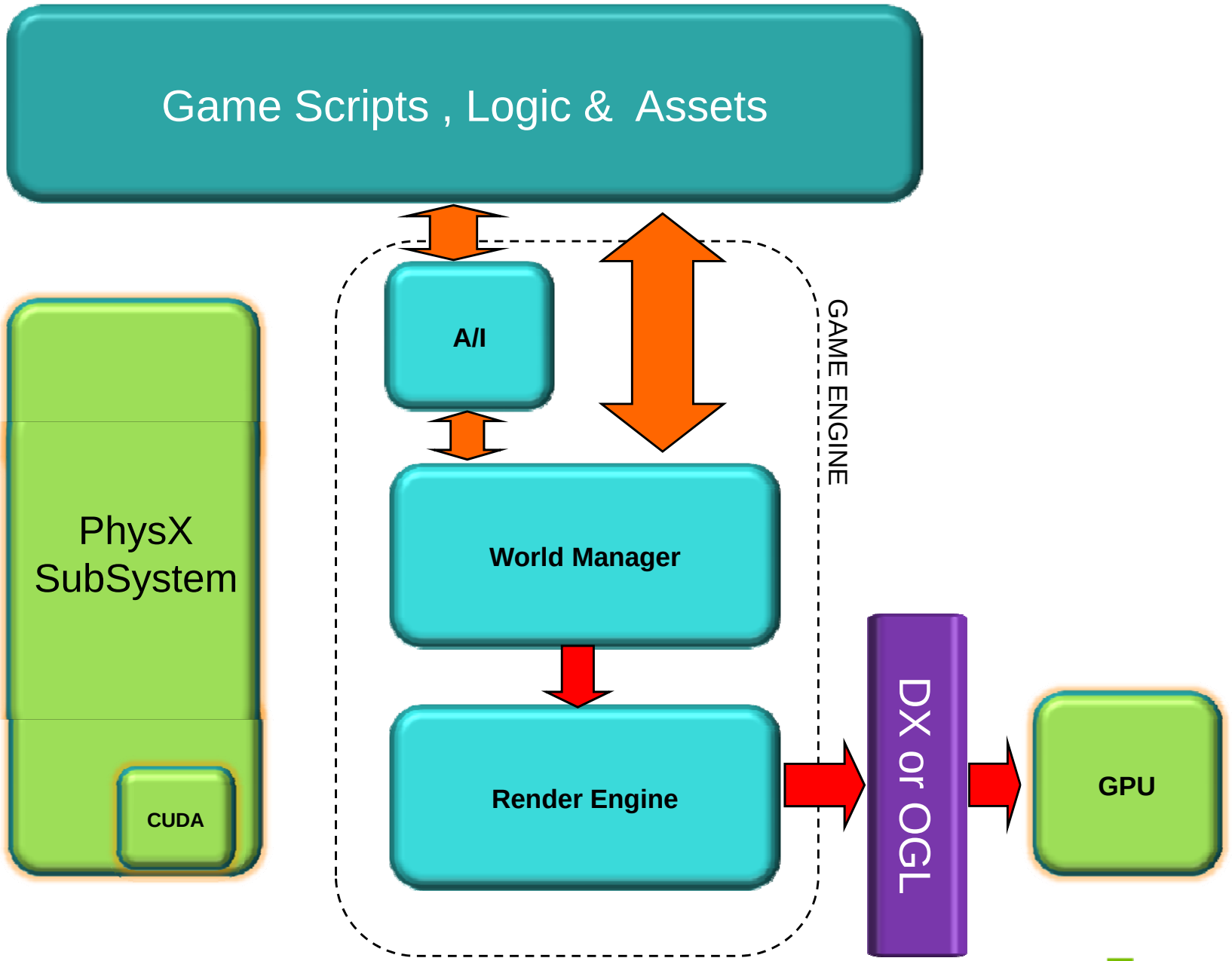
What is Gaming Physics?

PhysX Makes Games Come Alive

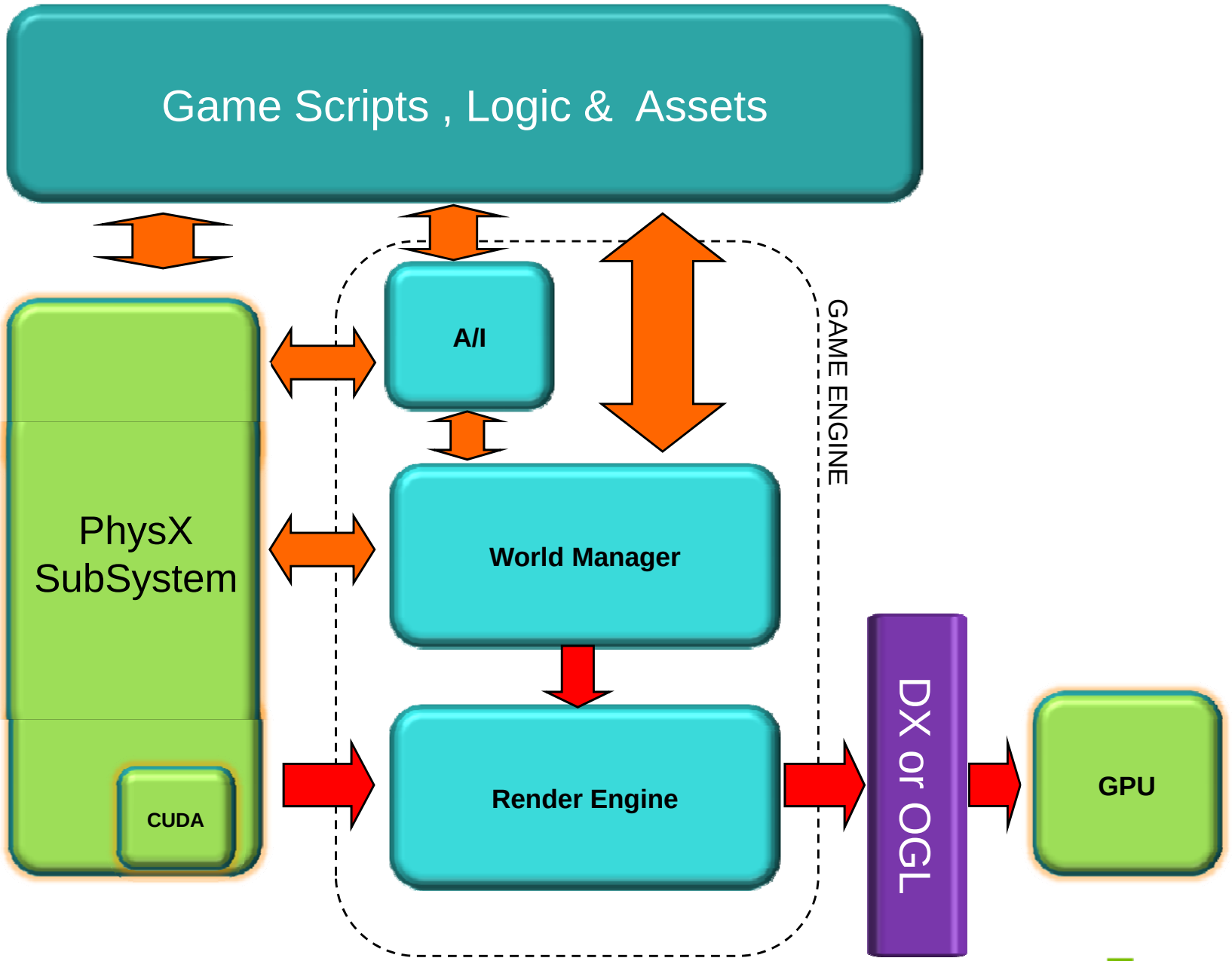
Simplified Game Pipeline



Simplified Game Pipeline



Simplified Game Pipeline



PhysX™
by NVIDIA

PLAYSTATION 3

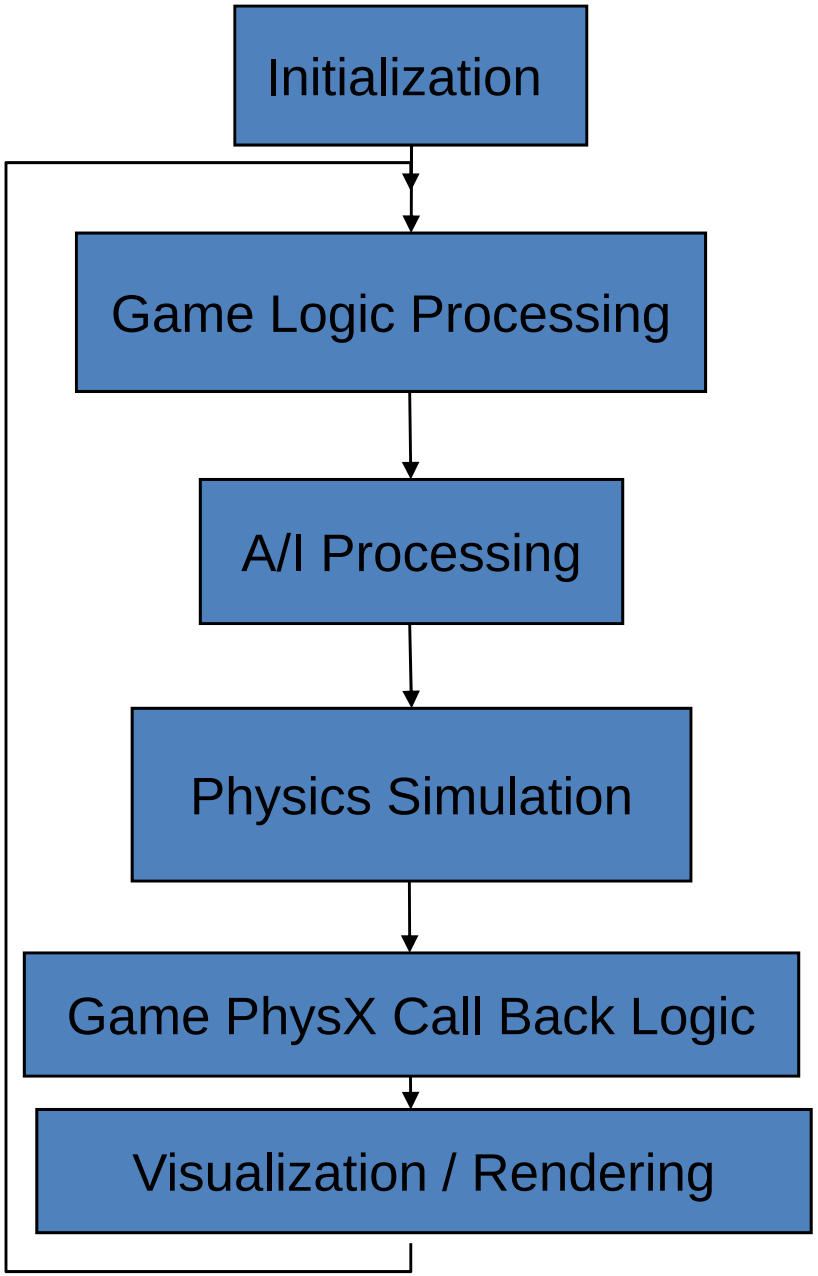
nVISION 08
THE WORLD OF VISUAL COMPUTING

NVIDIA

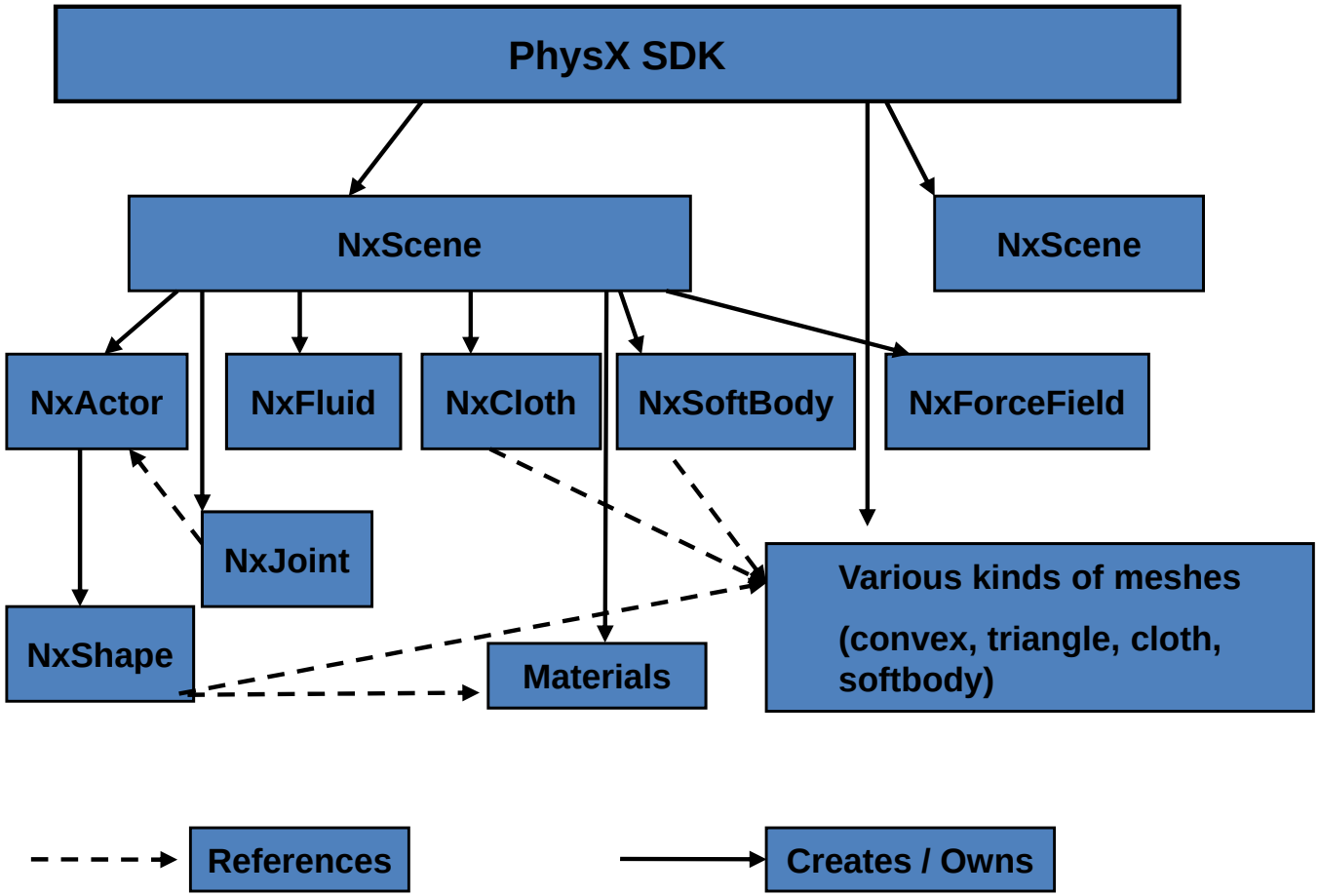
NVIDIA PhysX SDK Overview

- PhysX SDK is a complete Physics Solution
  - Comprehensive API
  - Library of auxiliary methods (Cross platform)
  - “Cooking” Library
  - Development Tools
  - Multi-Media Documentation
  - Multi-Tiered Support
  - Extensive Industry Eco-System

Simplified Game Flow With PhysX

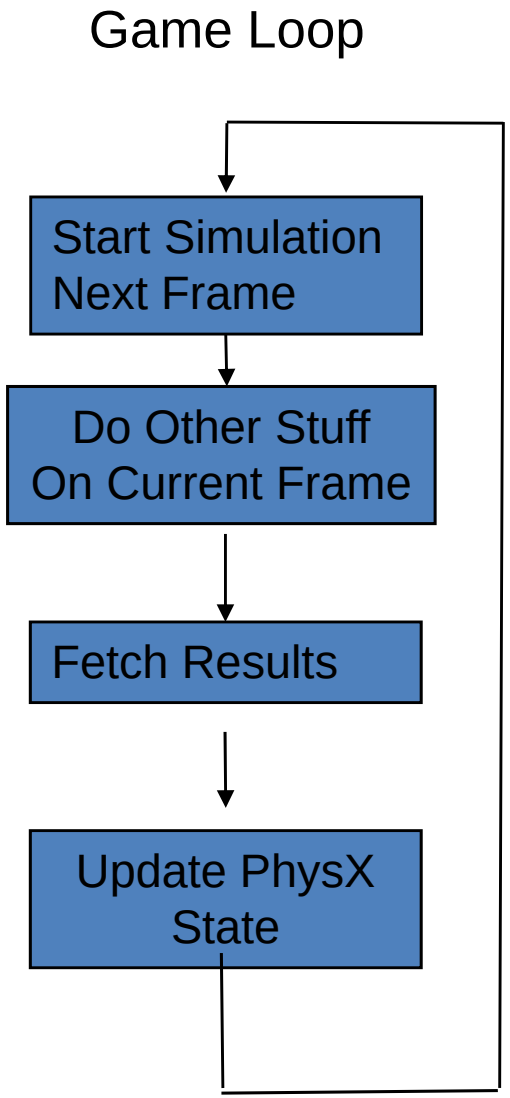


PhysX Object Model



PhysX Runtime

- Asynchronous Simulation Core
  - Rigid Body
  - Fluid
  - Cloth
  - Soft Body
  - Force Field
- Additional Functions
  - Scene Query
  - Character Controller
  - Vehicle Controller

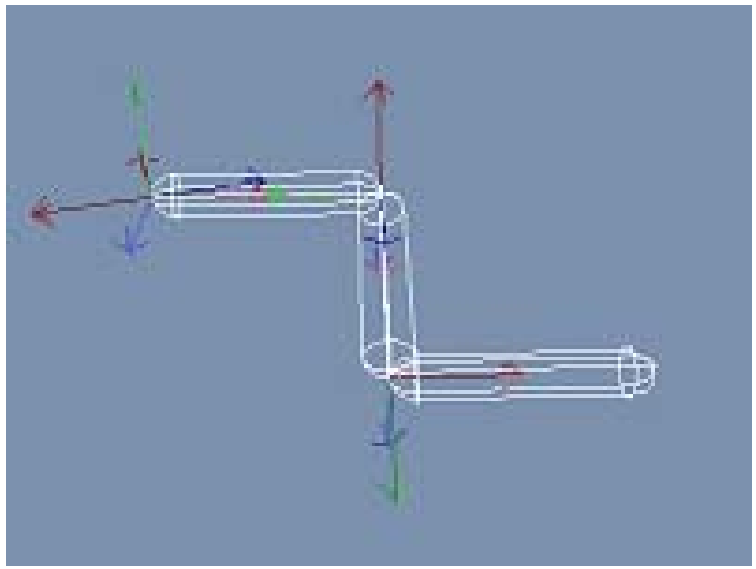


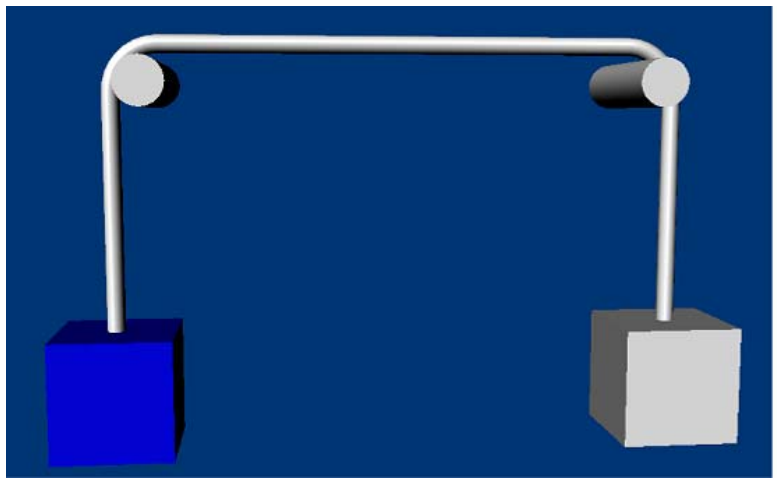
Rigid Body

- **Actor**
  - Static – fixed in world space
  - Dynamic – and have ‘body’
  - One or more Shapes
    - Geometry
    - Relative Transform
    - Material
- **Dynamic - Body**
  - Velocity (linear & angular)
  - Mass properties
  - Sleep properties
  - Can form mechanisms

Rigid Body

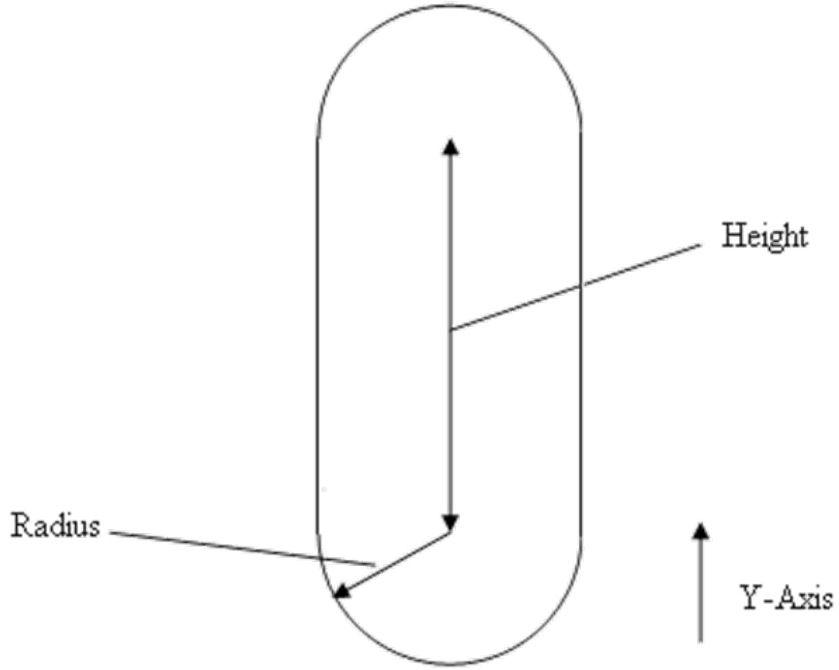
- Joints
  - a frame in each actor
  - constraints on those frames
- D6 and 8 additional types
- Includes support for
  - Motors
  - Springs
  - Limits
  - Pulleys





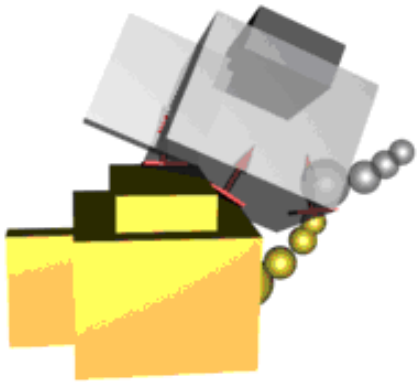
PhysX Shapes

- Single or Meshes
  - Capsule
  - Sphere
  - Box
  - Convex Mesh
  - Triangle Mesh



Physics Shape

Graphics Representation



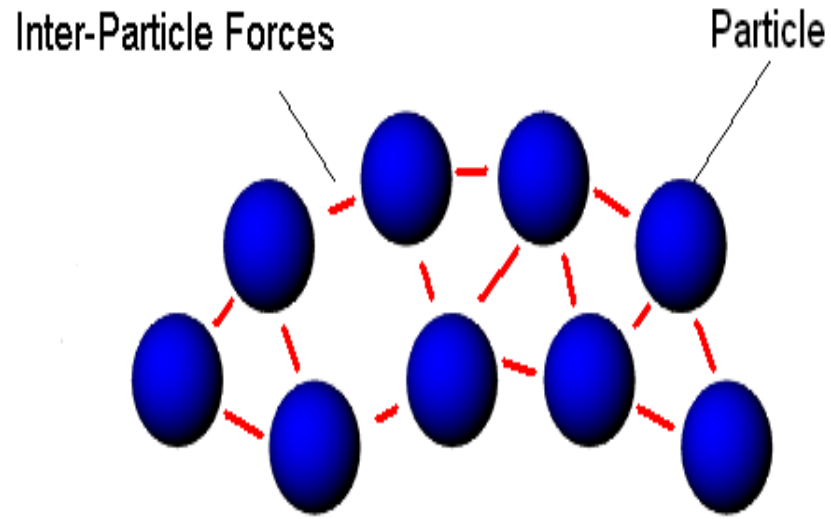
Bounding Box Representation

The Kinematics of any rigid body can be represented by a tensor and point.

Physics representation does not need to be same of the graphical mesh.

Fluid

- Colliding particle system
  - Position, velocity, lifetime, density, …
- Interaction Modes
  - SPH
  - Non-interacting
  - Mixed

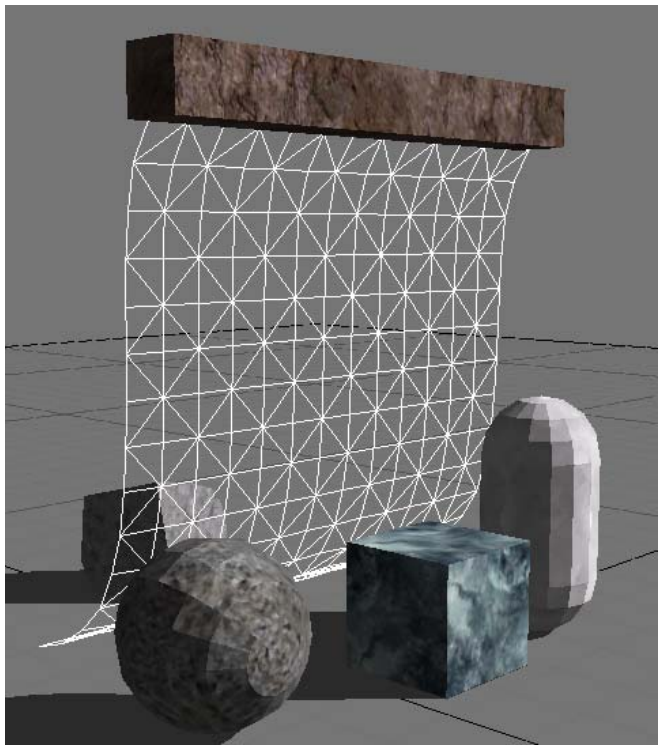


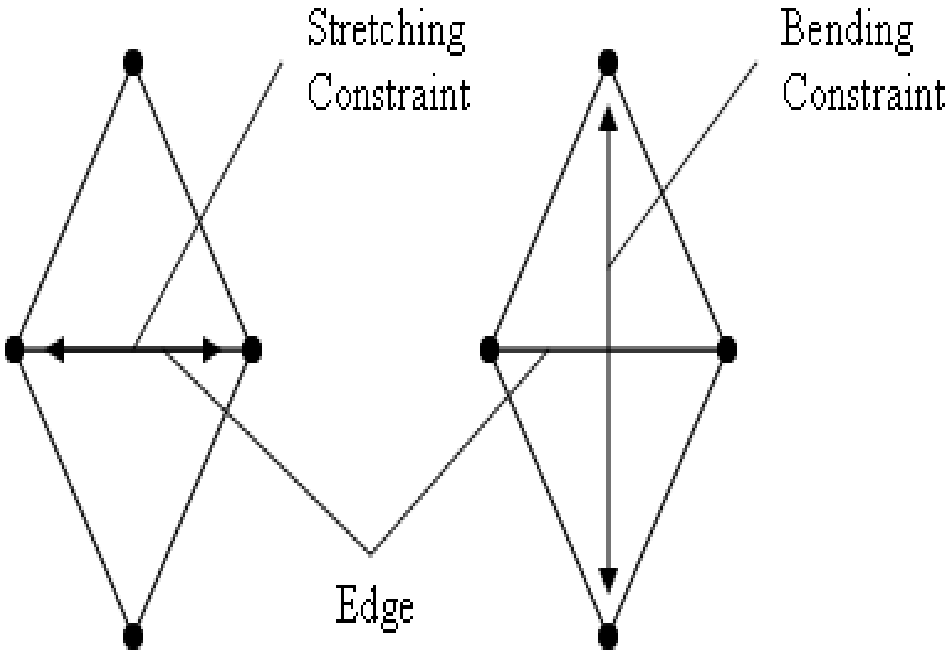
Fluid

- Particle Manipulation
  - add, update, remove particles
- Packet-based culling
- Emitter (Source)
  - can attach to shapes
- Drain (Sink)
  - NX_SF_FLUID_DRAIN flag on shape
- Event Notification

Cloth

- Mesh of particles
- Stretching and bending constraints





Cloth Parameters

NxCloth class parameters

- *Bending Stiffness*
- *Stretching Stiffness*
- *Density*
- *Thickness*
- *Damping*
- *Solver Iterations*
- *Attachment Response Coefficient*
- *Collision Response Coefficient*
- *Friction*
- *External Acceleration*
- *Wind Acceleration*
- *Valid bounds*

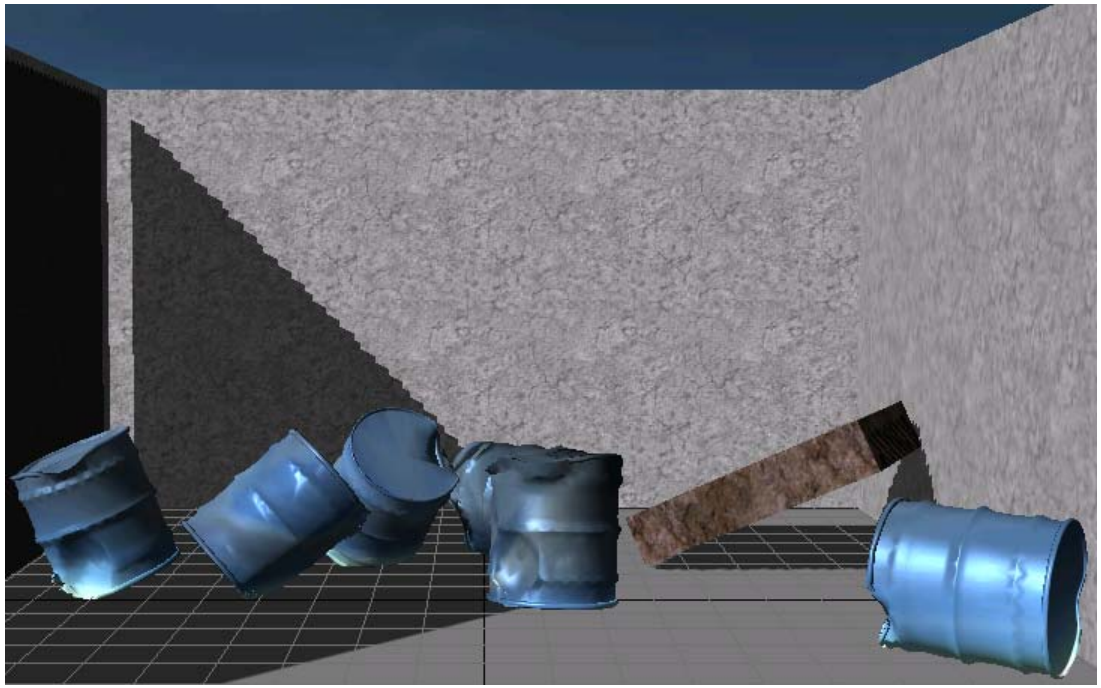
- *+ Selection of Flags to enable various affects*

Cloth

- Attachment
  - Attach vertex to fixed points or shapes
  - Detach vertex
- Tearing
  - automatic or explicit
  - tearable attachments
- Pressure
  - Closed meshes only
- Collision
  - Self-collision
  - Collision with rigid bodies

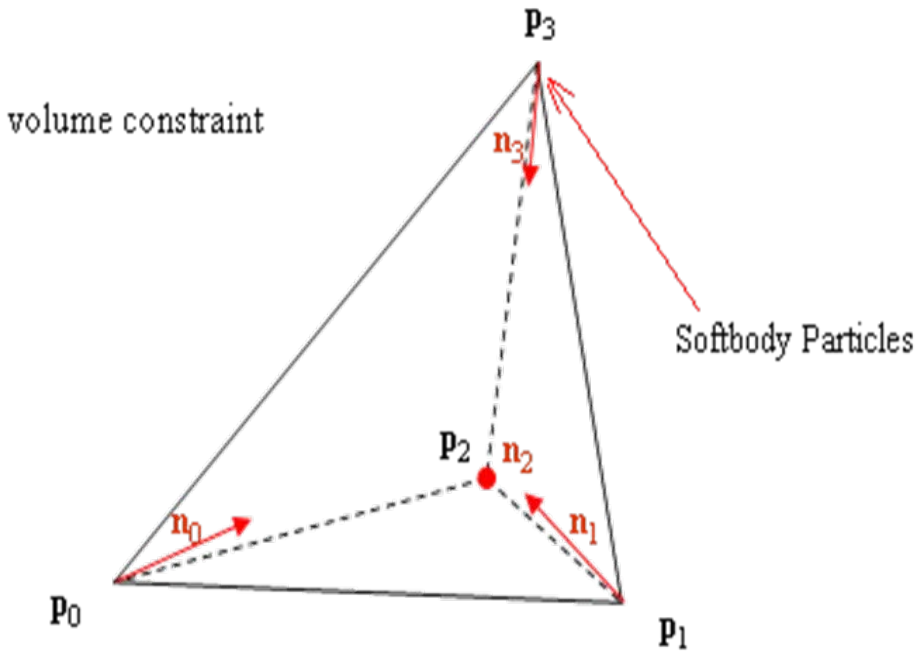
Metal Cloth

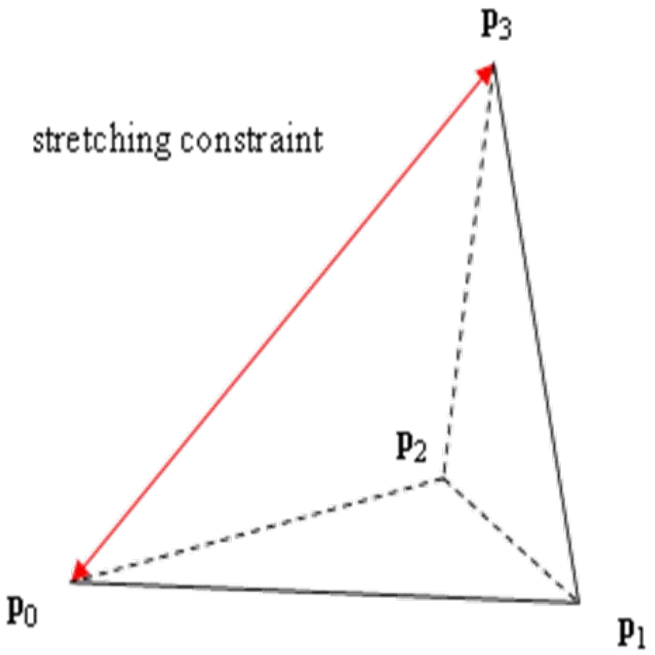
- Derivative of Cloth Feature
  - Plastic deformation of sheet metal
  - Cloth mesh around a rigid body core
  - On impact deform the mesh & adjust RB collision



Soft Bodies

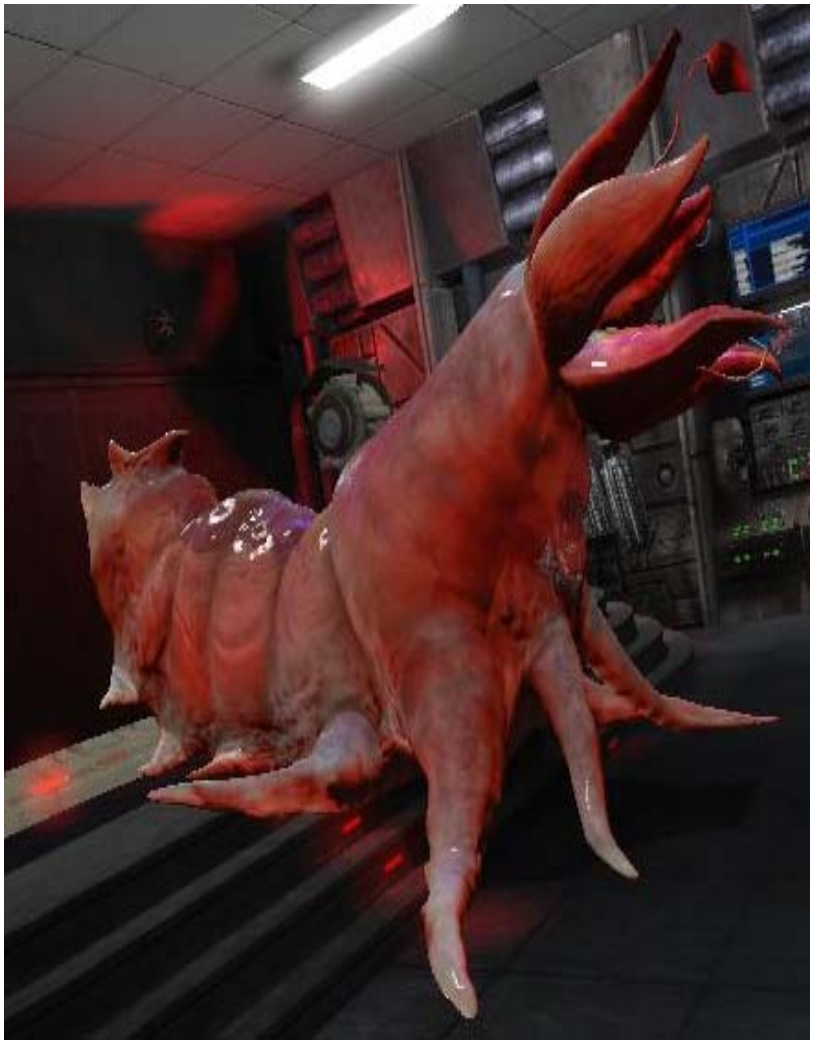
- Mesh of particles
- Volume-preserving tetrahedral constraints
- Uses cloth Solver





Soft Bodies

- Creation
  - Requires Tetrahedralization
  - Tetra-maker in PhysXViewer!
- Attachments & Tearing



Volumetric Force Fields

- Enables procedural insertion of energy into the system
- Activity volume
  - ‘Include’ & ‘Exclude’ volumes
- Data-driven Fixed-function Kernel
- Custom Kernel
  - Procedural shaders compiled at compile time and runtime

Tornado example FF provided in SDK

Other SDK Functionality

- Scene Query
  - Raycast
  - Swept Volume: Box/Capsule/Actor
  - Batching & Sweep Cache for performance
- Character Controller
  - A box- or capsule- shaped actor
  - Sweep tests for ease of walking
  - Ships with source
- And More …

Cooking

- Offline Asset Preprocessing
  - AABB Trees for Triangle Meshes
  - Convex Hull from point set, plus acceleration cube map
  - Cloth from tri-mesh, soft body from tet-mesh
- All assets cook to binary stream
- Conditioning
  - Vertex welding
  - Optional inflation and edge beveling for thin/sharp objects

PhysX Development Infrastructure

- APEX
- AgPerfmon
- PhysX Visual Debugger (VRD)
- SoftImage PlugIns
- Max PlugIns
- Maya PlugIns
- PhysXViewer
- TetMaker
- Video Tutorials

APEX Overview

NVIDIA Corporation

NVIDIA.

APEX Mission:
Solve 3 Big Game Physics Problems

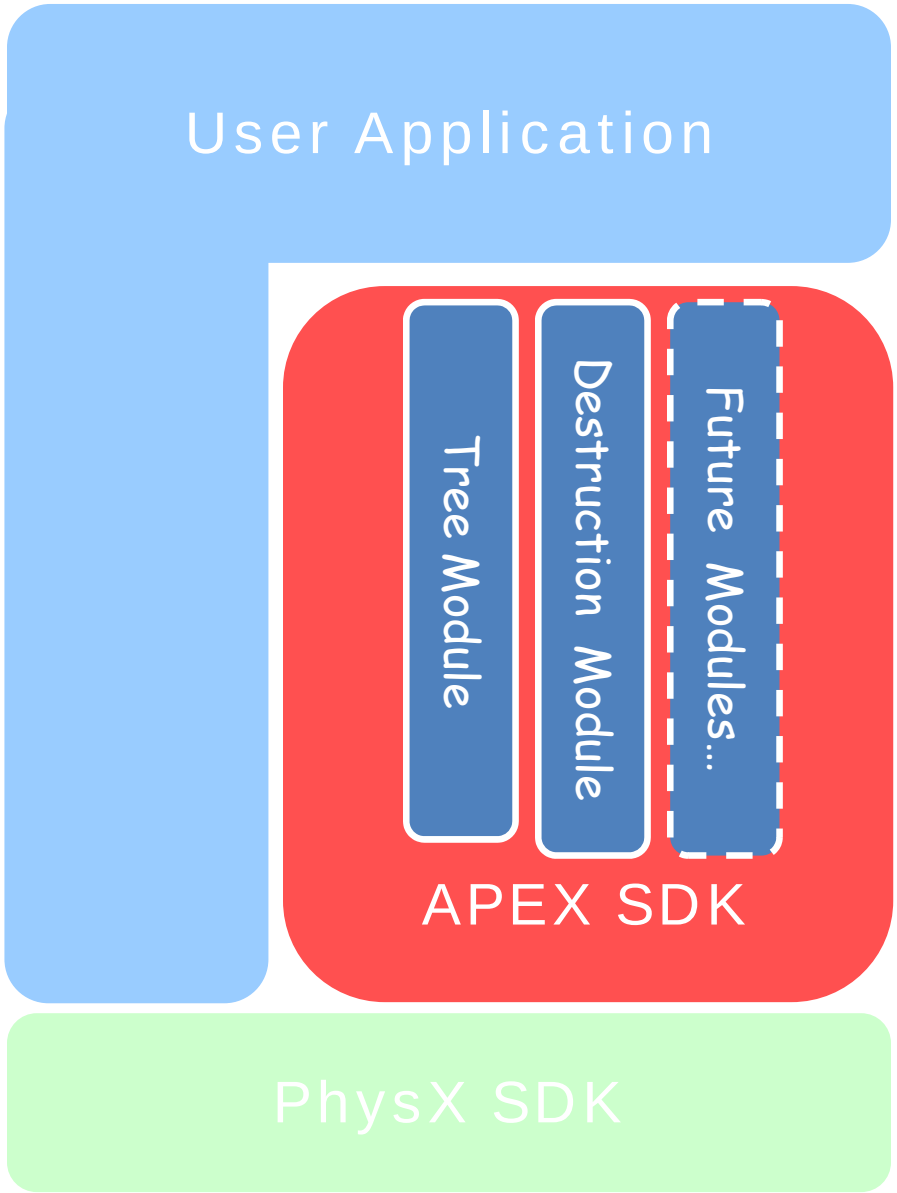
1. Significant programmer involvement

2. Content designed to “min spec”

3. Game engine limitations

Basic Design

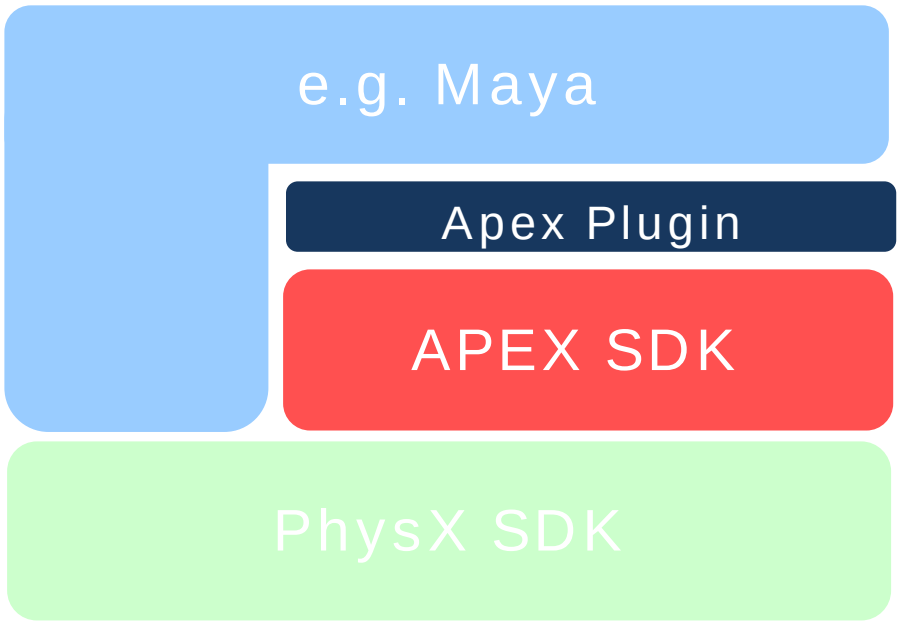
- A collection of “modules”
- Shared interfaces
- Built on top of the PhysX SDK
- User Application:
  - Game (runtime)
  - Authoring tools
  - Level editor

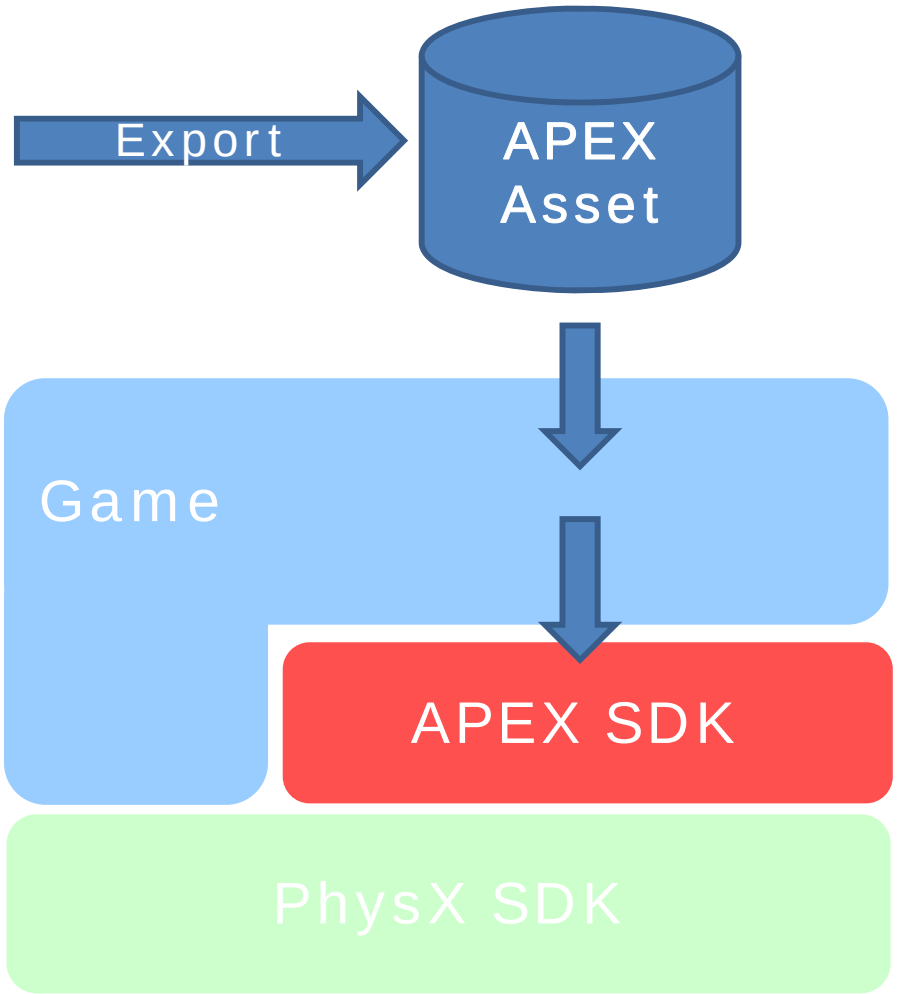


Modules

- **Implement intuitive, specific purpose, high-level physics technology**
- **Level of abstraction appropriate for content creators**
  - **Low SDK level of abstraction: rigid body, joint, cloth, fluid**
  - **High APEX level of abstraction (modules): vegetation, character, destructible mesh**
- **Manage multiple physics elements and simulation types**

Authoring





- **Modules are provided with authoring tools**
- **SDK has asset file serialization support built-in (load/store)**

Problem 1

- **Significant programmer involvement**
- **APEX Solution: Provide a “high-level” interface to the artists**
  - **Reduces the need for programmer time**
  - **Add automatic physics behavior to familiar objects**
  - **Leverage multiple SDK features**

Scaling

- Scaling graphics was easy…
  - Scaling physics, not so much!
- All modules provide load-time and/or run-time scalability mechanisms
  - E.g. Number of APEX objects to have active, total number of debris to keep around
- APEX assets are authored once
  - … to reduce work
  - But the developer still has control over scaling and LOD

Problem 2

- Content designed to “min spec”
- APEX Solution: scalable modules
  - Variable physics “quality” for each physical system
    - Static: hardware capability, player preference
    - Dynamic: visibility, distance from player, etc.

Interface to Rendering System

- **APEX has a unified API for sending data directly to the rendering engine**
  - **Shared by all modules**
  - **Bypass game logic whenever possible**
  - **Efficient, but flexible**

- **Application implements a few interfaces**
  - **APEX objects ask the game to allocate buffers**
  - **APEX streams rendering data to the buffers**
  - **Application renders the buffers with appropriate materials, shaders, etc…**

Problem 3

- Game engine limitations
- Solution: Create a rendering “fast path”
  - Bypass the inefficient, fully generic path
  - PhysX objects in an APEX asset are scriptable / networkable / etc. as a group, not individually

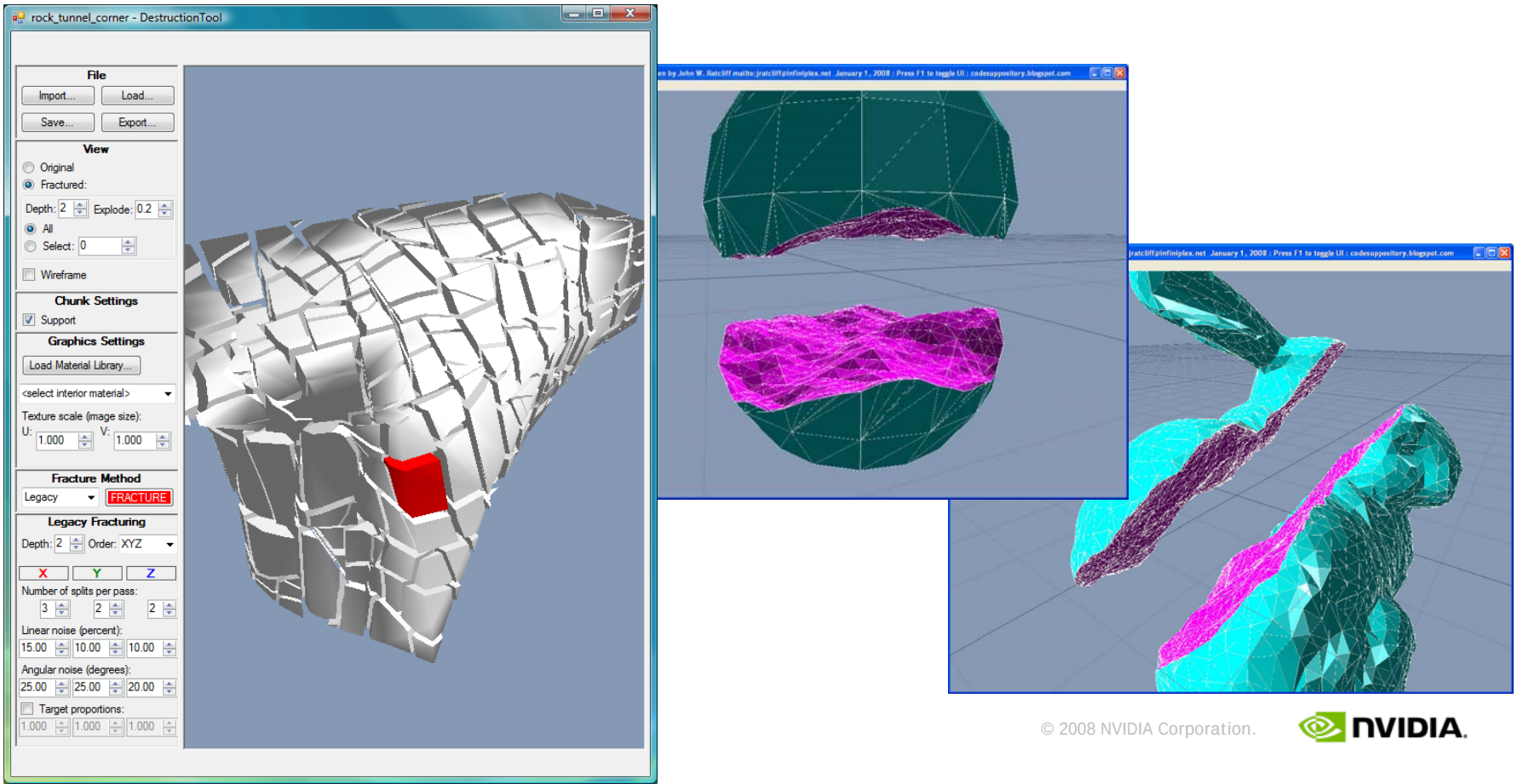
Destruction Module

- Arbitrary meshes are pre-fractured at authoring time
- In real time as they take damage, pieces get blasted away
- Meshes can be initially static or dynamic
- “Support” system for static meshes
- Automatic small debris effect using particles


0
1st / 0
+100
100

Destruction Authoring

- **Standalone tool, easily integrated into 3dsmax, Maya, or XSI**
  - **Hierarchical splitting with random fracture surface generation**



Destruction Scalability

- At authoring time:
  - # of pieces the original asset is split into
- At runtime:
  - Whether to fracture down all the way to the finest level of pre-fracture, or only to a coarser level
  - Option to scale amount of particle meshes
  - Size adjustable global debris FIFO
- The game can change runtime APEX parameters based on LOD.

Tree Module

- Lets us create trees with physical behaviors
- Works with SpeedTree trees
- Full physical interactions
- Tree destruction
- Leaf dropping effect

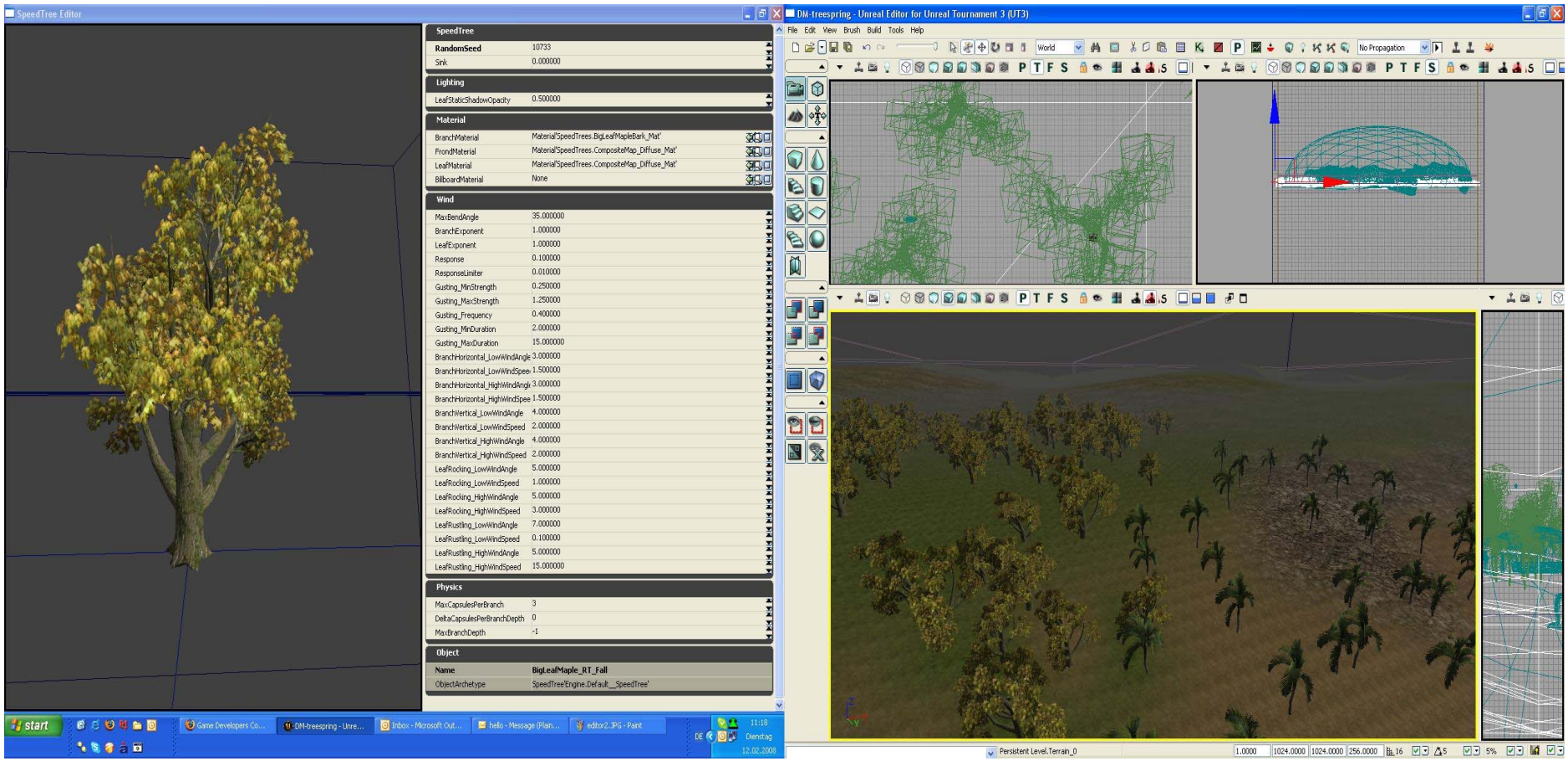
Tree Physics

- Trees skeletons are automatically generated
- Skeletons are only simulated when trees are being interacted with (LOD)
- Emitters are automatically created to spawn leaves


www.fraps.com
0
1st / 0
+100

Tree Authoring

- **Authoring process same as for normal trees, except for a few simple physics parameters**
  - **Capable of directly loading SpeedTree format files**



Tree Scalability

- **At load / authoring time:**
  - **A particular tree asset can have variable detail RB skeleton generated**
- **At runtime:**
  - **A large forest can have more or less of its trees become physically active at any one time, in response to interactions**
    - **Tree actor FIFO for fast activation**
  - **Leaf emitters can emit more or less leaves**

More modules…

- **Actively seeking great ideas**
  - **and partners to work with…**
- **What would you like to see?**

AgPerfMon & VRD

NVIDIA Corporation

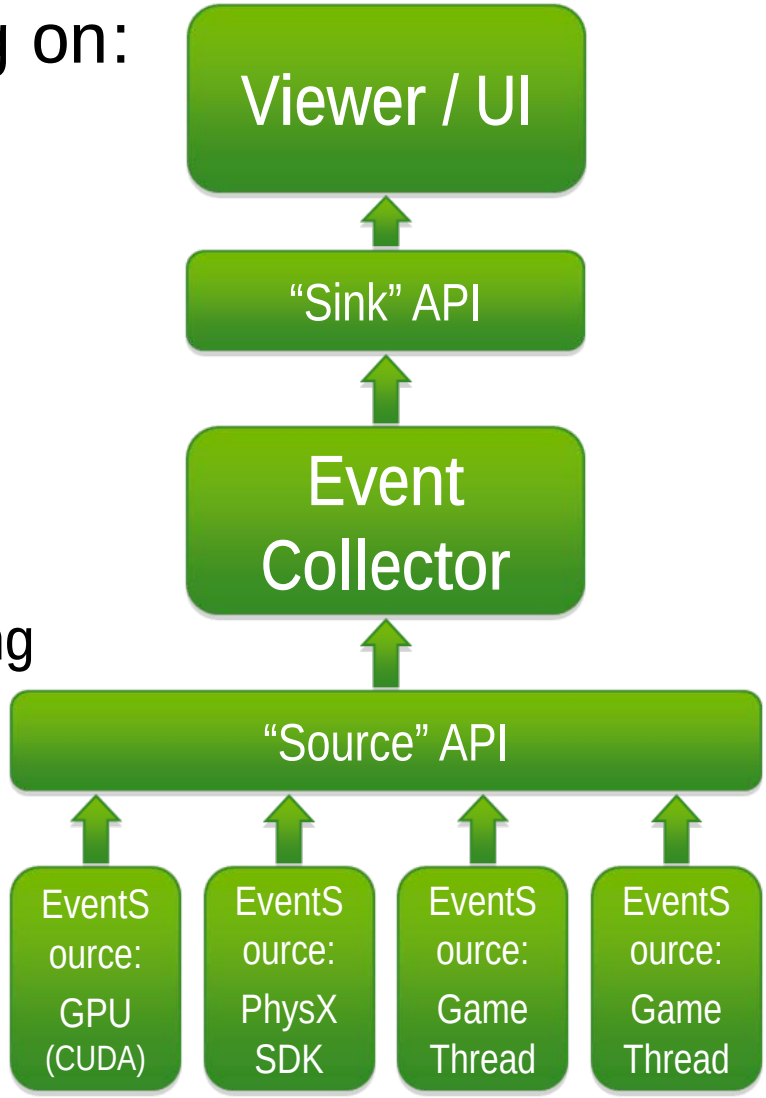
NVIDIA

Problem Statement

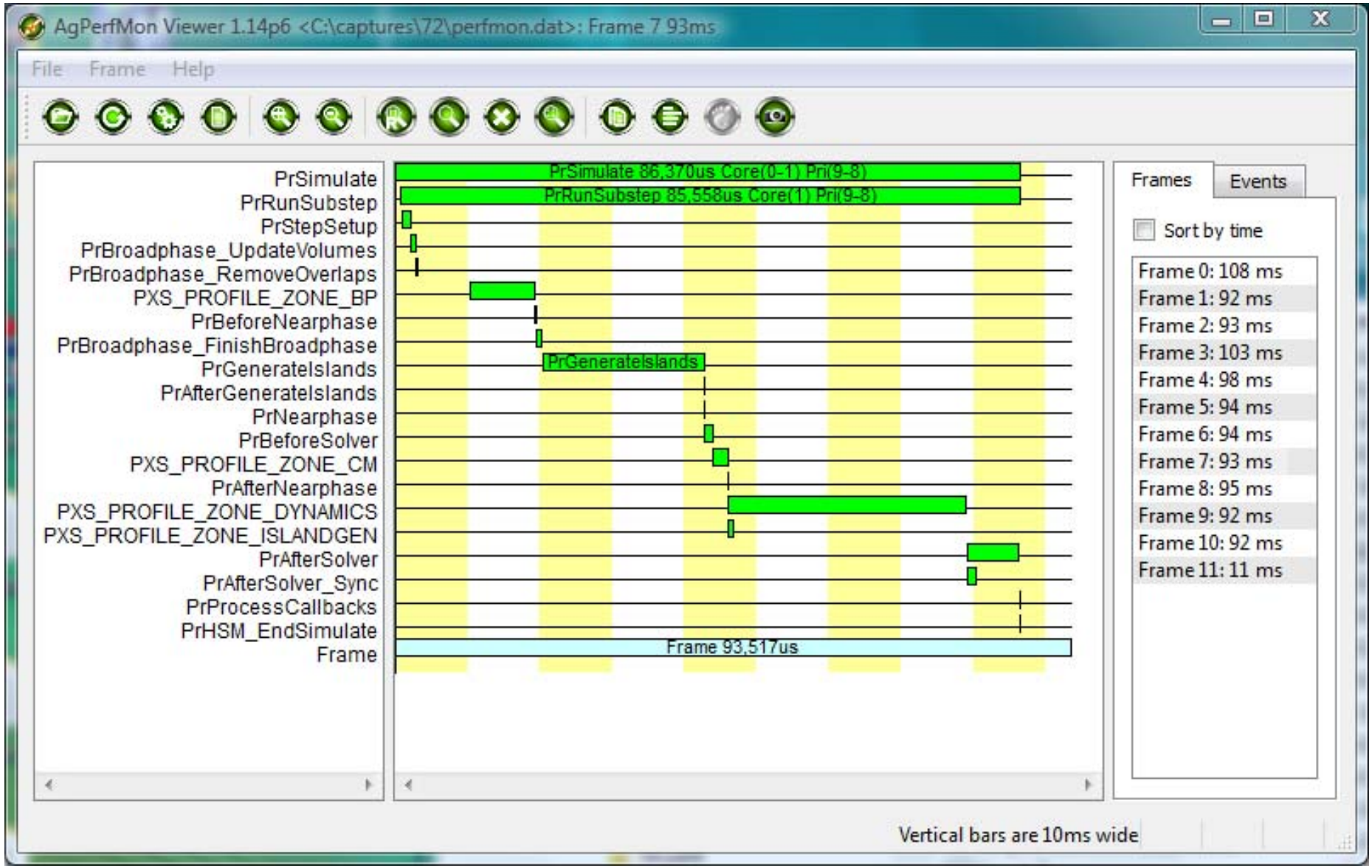
- Identifying system-wide bottlenecks is very challenging
  - Many asynchronous processes running in parallel on different processing cores
  - Can't just analyze who's using the most cycles: dependencies matter
  - Visualizing "what happens when" is critical

AgPerfMon Overview

- Supports simultaneous profiling on:
  - CPU Threads: PhysX / APEX SDKs, game / application threads
  - GPU: CUDA Kernels, Warps
- Event-logging architecture
  - Synchronized time stamps
  - Configurable source-based filtering
  - Configurable event triggers
  - Periodic event generation
  - Source & sink API's
  - VERY lightweight



Event Viewer



Source API – Initialization

YourCode.cpp:

```
#include “AgPerfMonEventSrcAPI.h”

// Initialization code
AgPerfUtils *gPerfUtils = new AgPerfUtils;
```

```
// Shutdown code
delete gPerfUtils;
```

Source API – Event Generation

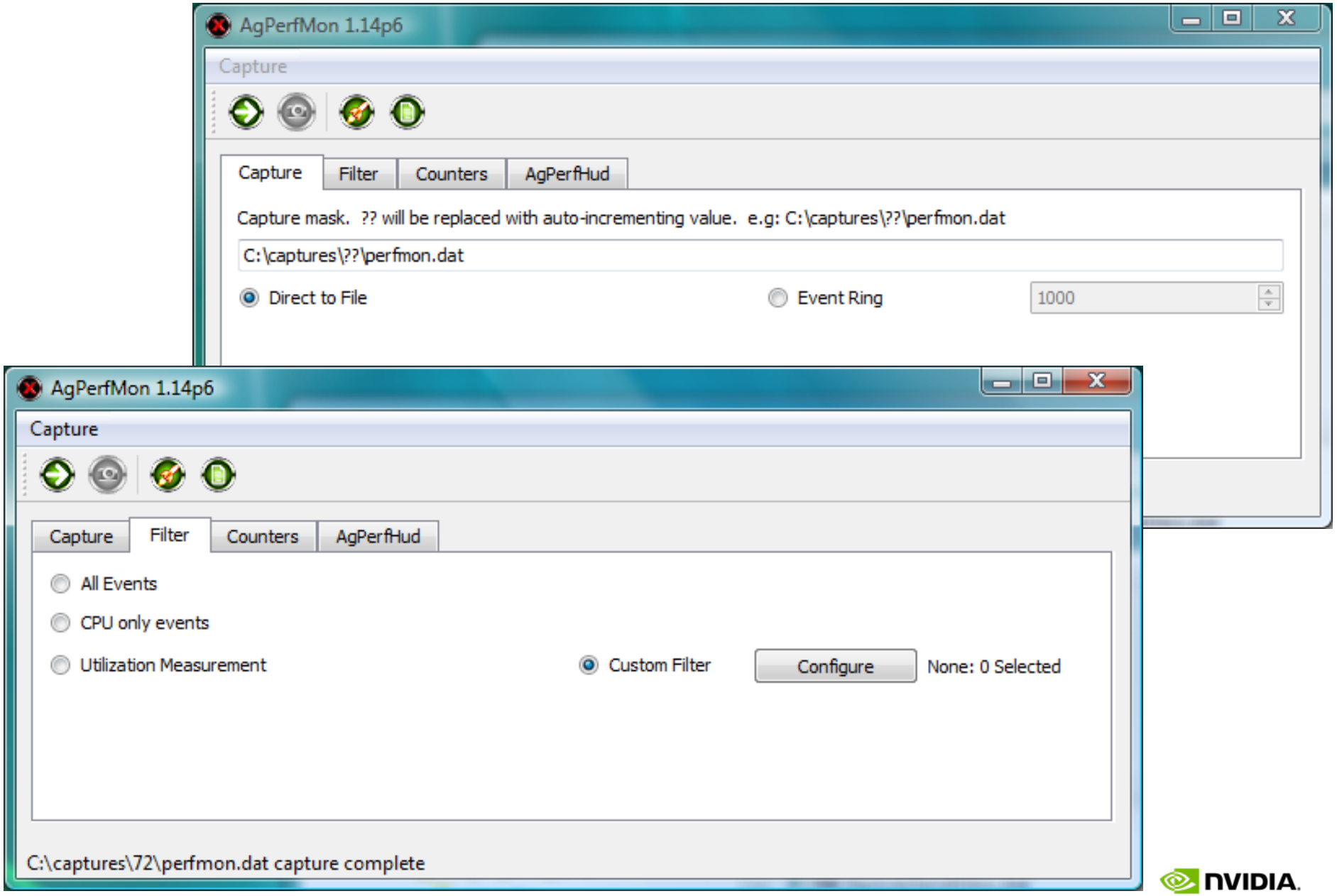
AgPerfMonEventDefs.h:

```
DEFINE_EVENT(Foo)
```

YourCode.cpp:

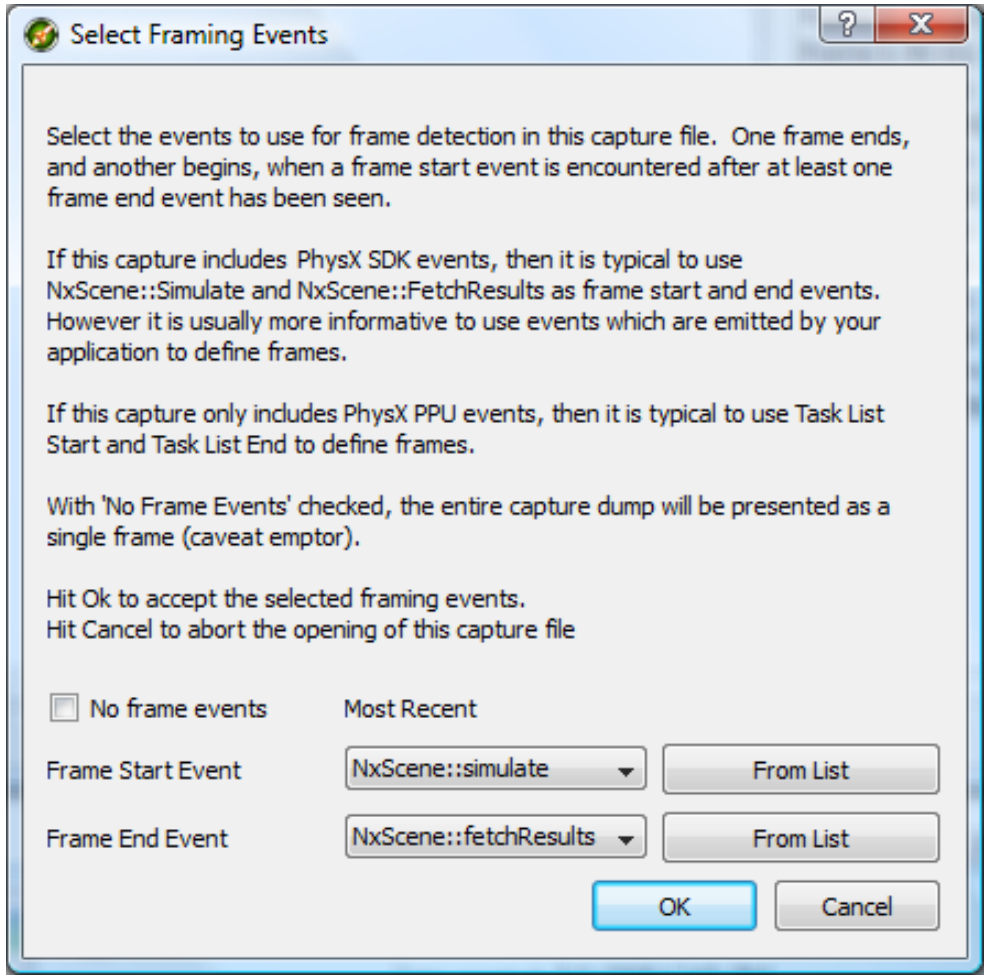
```
{
   AgPerfScope s(Foo);
   // do something
}
```

AgPerfMon GUI

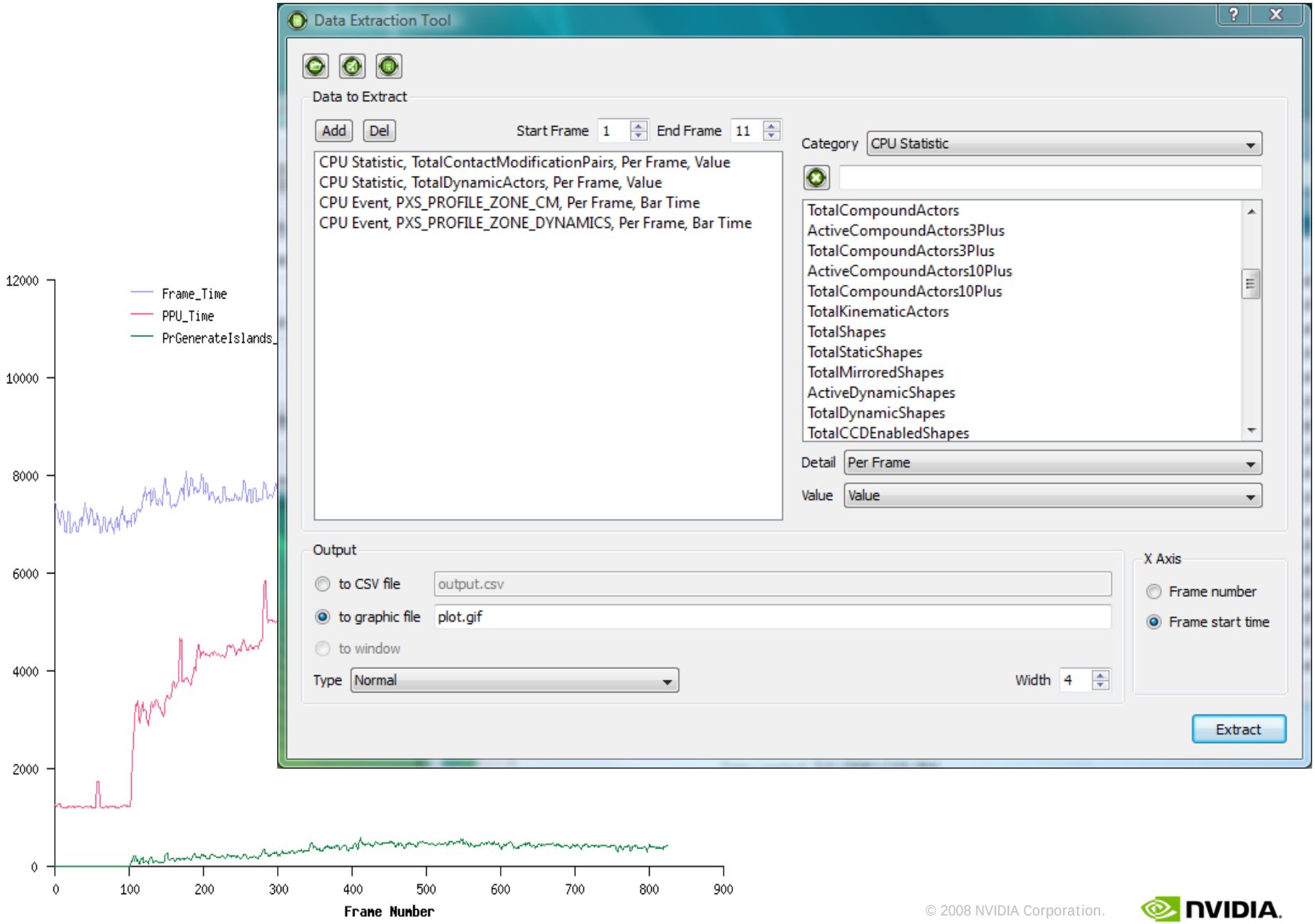


AgPerfMon Viewer

- Frame events
  - Define frame boundaries
    - Frame Start Event marks the start of frame…
    - … only if a Frame End Event has been seen since last start of frame
  - No frame events
    - View entire file as a single frame
    - Only works for small files
  - Index cache
    - Viewer builds an index cache to accelerate finding frames in large data files (*.fdx)

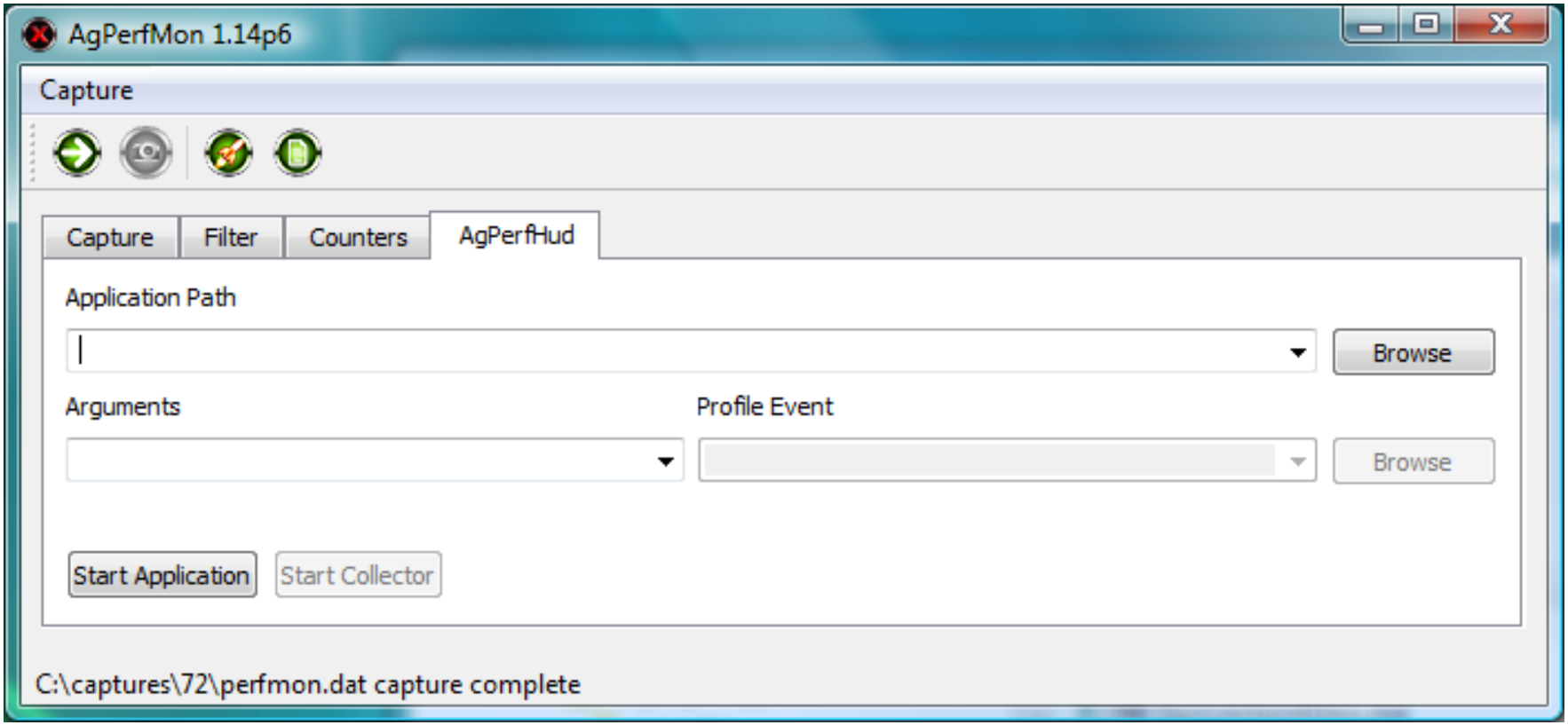


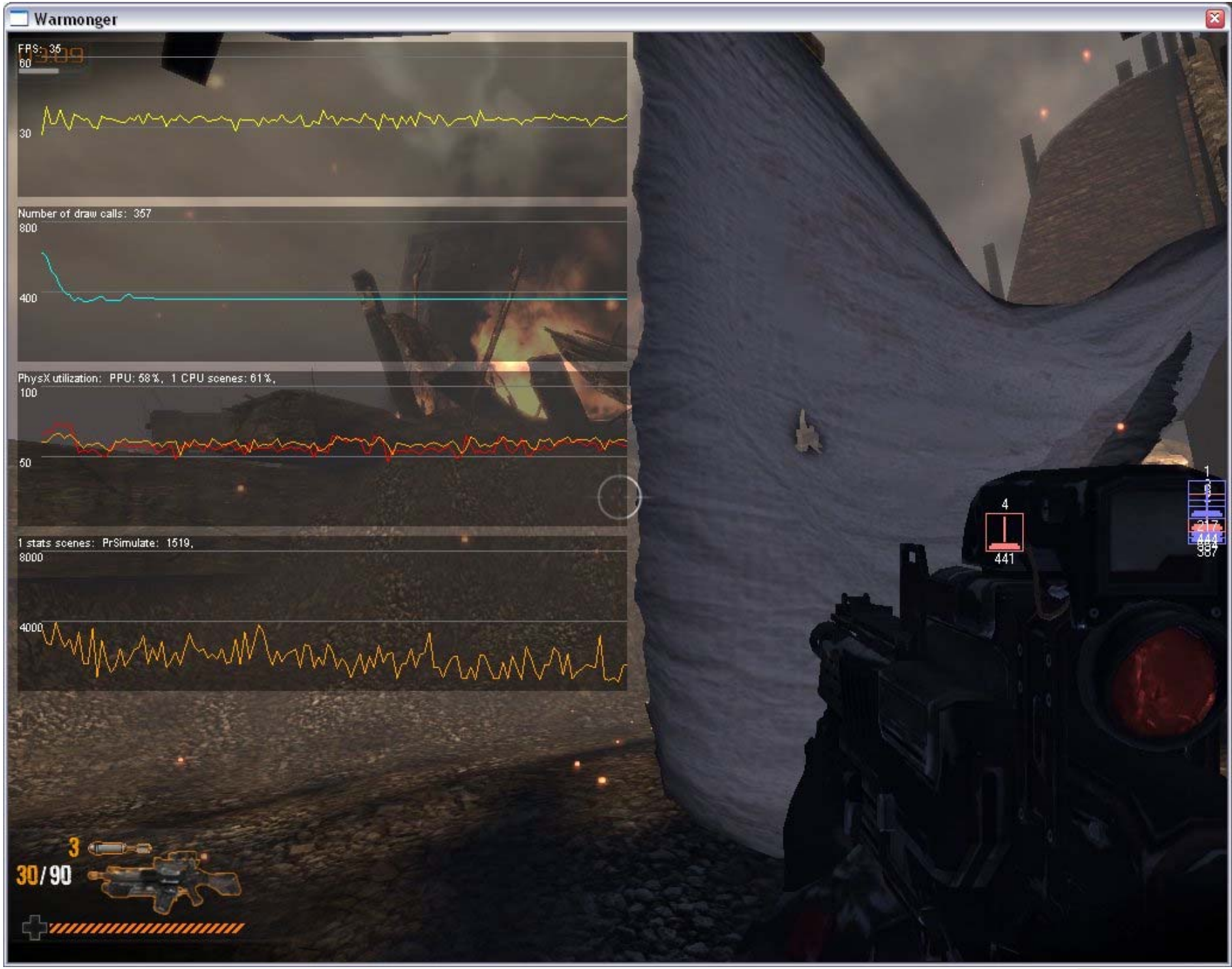
Data Extraction Tool



AgPerfHud

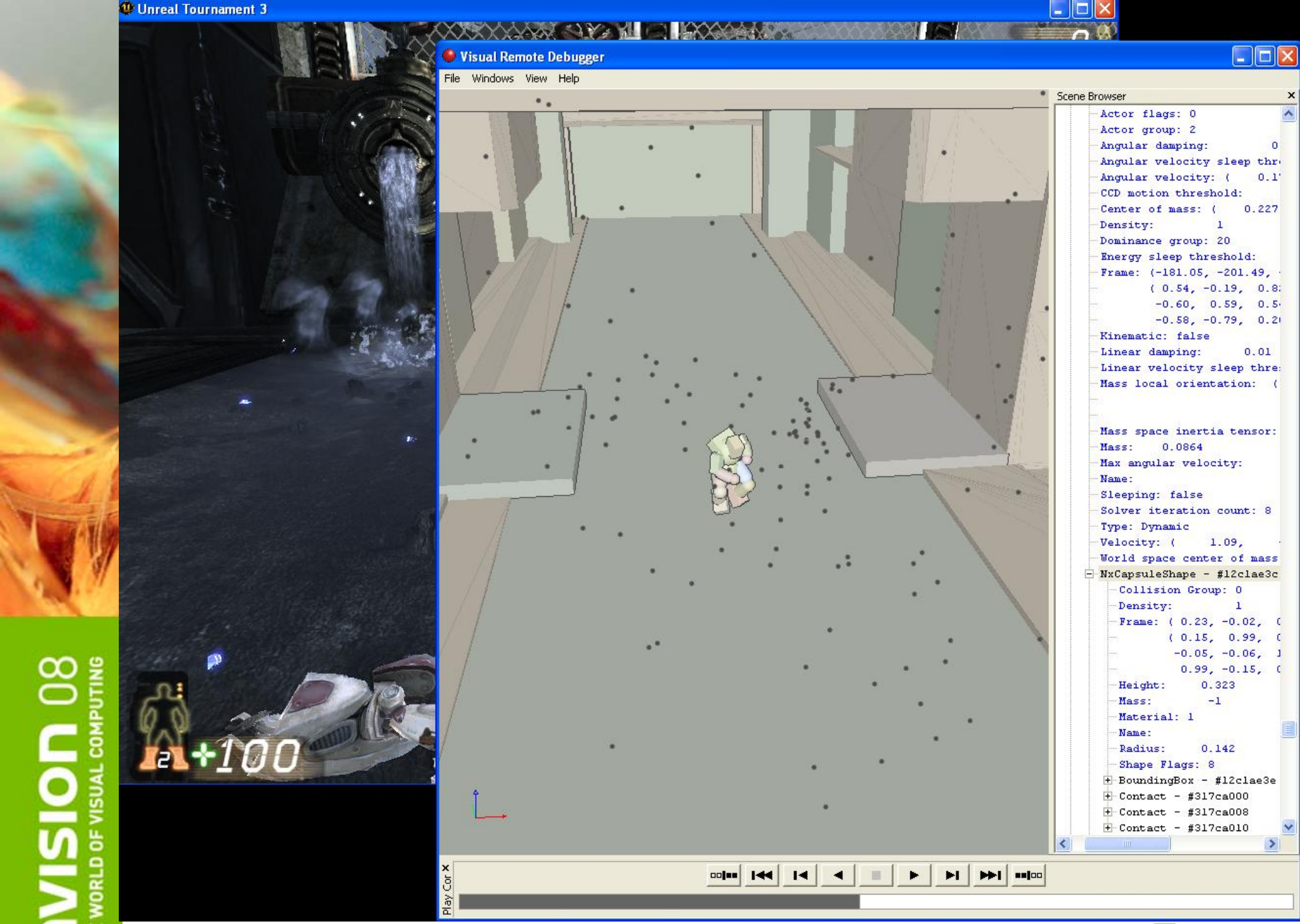
- Simple utility to overlay AgPerfMon performance data on any D3D application window





Visual Remote Debugger (VRD)

- Connects to a running PhysX SDK through a network socket
- Provides access to all simulation state data
  - Live visualization
  - Record / playback / save to file
- Full visualization of physics scene
  - Shapes / bounding boxes / particles
  - Contacts / joints
  - Velocities
- Data editing
  - Click on an object to view / modify state in “Scene Browser” window



PhysX SDK Licensing and Support

NVIDIA Corporation

NVIDIA

Platform Coverage

	PhysX™ by NVIDIA	Competitor's Solution
PC Support - CPU only	YES Multi-Threaded Support for Multi-Core x86 Products	YES
PC Acceleration	YES! GPUs (CUDA): Mobile + Desktop	NO
Xbox 360	YES	YES
Playstation3	YES	YES
Wii	YES*	YES

PhysX License Fees

Per Game License Fee	PC[3]	PS3[1]	Xbox 360	XBLA	Wii[2]	Wii-Ware
Binary	Free	Free	Free	Free	Free	Free
Source	**$50K**	**$50K**	**$50K**	**$50K**	**$50K**	**$50K**

1 The PS3 PhysX SDK has been maintained and supported by Sony. If you are a PS3 registered developer, you can find the PhysX SDK on Sony's online download site. NVIDIA will soon take direct ownership of licensing.

2 Wii SDK in Beta

3 Linux Driver Support Available

PhysX License Fees

Binary SDK

- SDK – Unified PhysX API for both PC and Console Platforms
- PC Binary SDK Free for both Commercial **AND** Non-Commercial use
  - No License Fee Required
  - Over 30,000 Downloads
- Console SDK's Free for Registered Developers for both Commercial **AND** **Non-**Commercial use
  - EULA covers terms

Source SDK

- Individual Game License for each Platform
  - Multi-title flexibility on terms
- Source Code SDK includes HL source code to facilitate debugging process

Developer Support

- **Multi-language documentation on the way:**
  - Japanese, Chinese, and Korean
- **Two Levels of Support:**
  - Free forum support always available via NVIDIA's developer website

    developer.NVIDIA.com/forums
  - Ticket-based support and staffed in local time-zones worldwide
  - Paid Support Model - $8k annual/game/platform