VSL
VISUAL SIMULATION LABORATORY

Visually Programming GPUs in VSL

Jefferson Amstutz
Applied Technology Operation
SURVICE Engineering

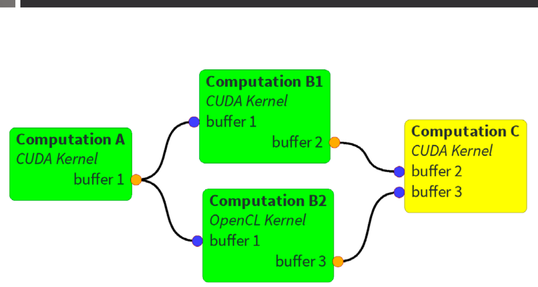
Scott Shaw
Applied Technology Operation
SURVICE Engineering

Lee Butler
U.S. Army Research Laboratory

Overview

The Visual Simulation Laboratory (VSL) is an ongoing, open source framework developed by the U.S. Army Research Laboratory and its collaborators to bring modern hardware and software to a variety of DoD application domains. VSL is designed to transform legacy workflows into immersive, physics-based simulation and analysis tools.

Modern simulation codes can be very complex and difficult to understand by both developers and end-users. VSL addresses this problem by using a visual programming interface (VPI) for constructing and running simulation pipelines. This interface provides two useful benefits for both developers and users alike. First, insight into what a simulation code is computing on behalf of the user helps to de-mystify results. Second, it is coupled tightly with the rest of VSL's 2D/3D visualization capabilities which makes it easy to develop and run additions to existing simulation pipelines.



Pipeline framework

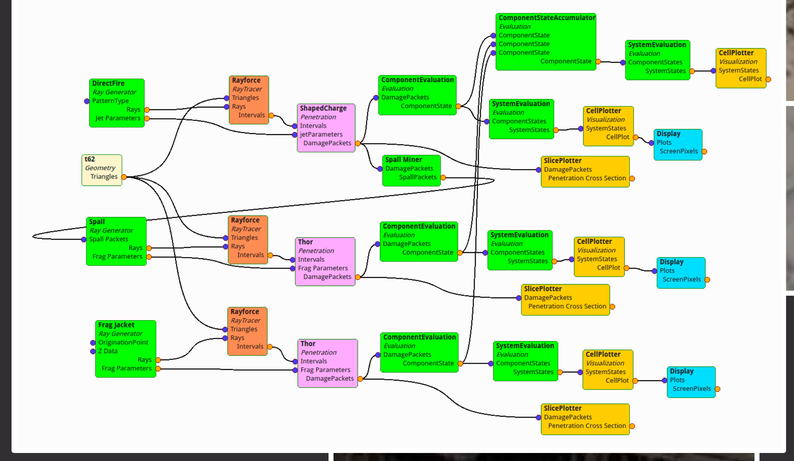
The pipeline model and VPI are implemented within VSL's existing framework on top of Qt. VSL's pipeline model contains the relationship of computational nodes to be executed as a directed graph. Each node along the pipeline represent a single, modular element of the computation. Connections between nodes represent intermediate data buffers passed to different stages of the computation. The view VSL implements is an interactive diagram of the pipeline that can be modified by the user at run time.

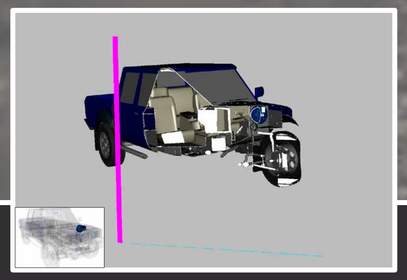
Computational nodes

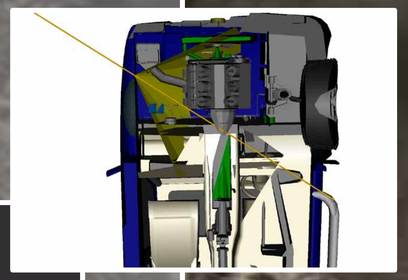
The Node class itself is very flexible and easy to work with. The four basic elements of a Node are:

1. input data buffer(s),
2. (optional) parameters that can be used as additional input,
3. the computation on the input data, and
4. the resulting output data buffer(s).

The advantage of the generality of Nodes is that they can implement operations in many domains on any available hardware platforms. For example, a node that ray traces geometry for a buffer of intersection points could be implemented using a CUDA based ray tracer or implemented with an OpenCL based ray tracer. This makes it easy to substitute either node in a large pipeline for performance comparison or CUDA enabled hardware availability, without the need to modify any of the other existing pipeline nodes.





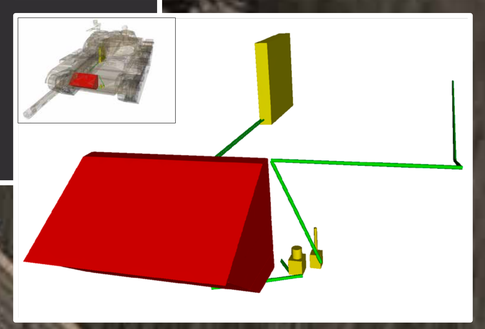


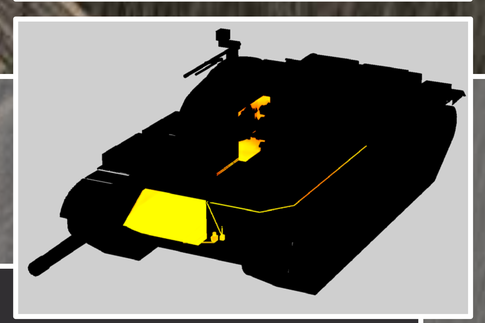
Example implementation

VSL currently has nodes that implement a traditional ballistic simulation pipeline, which include nodes that use CUDA to compute:

1. multi-hit ray intersection points,
2. threat penetration,
3. component damage assessment,
4. system fault-tree evaluation, and
5. result visualization.

Each node in pipeline executes a single, specific GPU kernel, but allows them to be connected together on demand at runtime. Some nodes have additional input parameters that allow the execution of nodes to behave differently, depending on the demands of the simulation.





Future work

While this work has seen recent success for flexible acceleration of ballistic simulations, there are improvements we are seeking to implement. Such improvements include optimization of data movement between GPUs and the host, view based node encapsulation in the UI, and expanding node implementations to domains beyond ballistics.

NVIDIA CUDA · OpenCL · intel inside Xeon Phi

Source code available via http://www.vissimlab.org

ARL SURVICE

NVIDIA CUDA RESEARCH CENTER