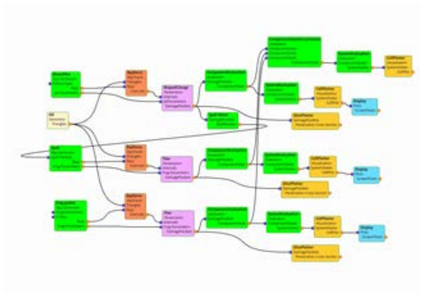
Visually Programming GPUs in VSL

Jefferson Amstutz[*] Scott Shaw
Applied Technology Operation
SURVICE Engineering

Lee Butler
U.S. Army Research Laboratory


(a) *Visual Programming Interface*

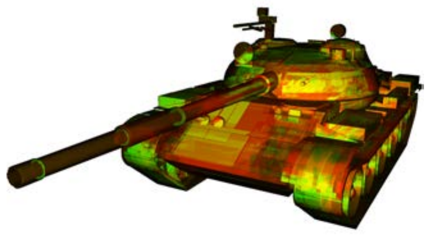

(b) *Ballistic Simulation*

Figure 1: (a) Visual Programming Interface. *VSL uses a visual programming interface (VPI) for constructing and running simulation pipelines on massively parallel hardware.* (b) Ballistic Simulation. *VSL currently has nodes that implement a traditional ballistic simulation pipeline. The above image (b) shows one of the visuals VSL can create to assess vehicle vulnerability.*

1 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 work flows into immersive, physics-based simulation and analysis tools.

2 Motivation

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 demystify 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.

3 Initial Implementation

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.

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

1. **Ray Tracing**: Find multi-hit ray intersection points
2. **Penetration**: Propagate a threat along the ray intersections
3. **Damage Assessment**: Calculate the component probability of kill given damage
4. **Fault-Tree Evaluation**: Evaluate the probability of kill for interested systems of components

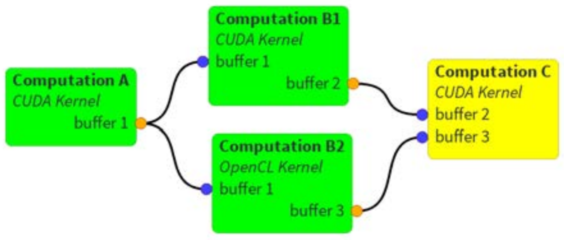


Figure 2: Computation Nodes. *Nodes can easily be connected at runtime to create multi-stage computations on-the-fly. Nodes can be implemented with any available compute device or compute environment, such as CUDA and OpenCL.*

5. **Visualization**: Create a visualization of the results

4 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 are optimization of data movement between GPUs and the host, view based node encapsulation in the UI, and expanding node implementations beyond ballistics.

Acknowledgments

This work is funded in part by the US Army Research Laboratory.

[*]e-mail: jeff.amstutz@survice.com