VISUAL SIMULATION LABORATORY

# A Novel Optimization & Application of Order-Independent Transparency

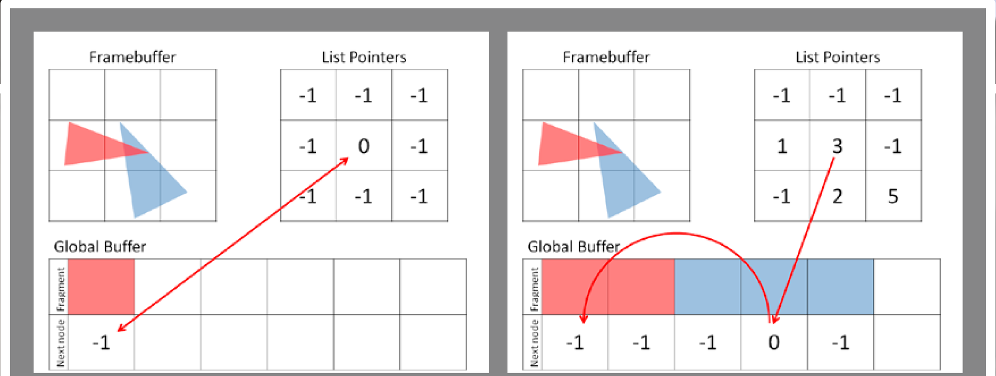
**Ethan Kerzner**
Department of Computer Science
University of Iowa

**Chris Wyman**
Department of Computer Science
University of Iowa

**Lee A. Butler**
U.S. Army Research Laboratory

## Overview

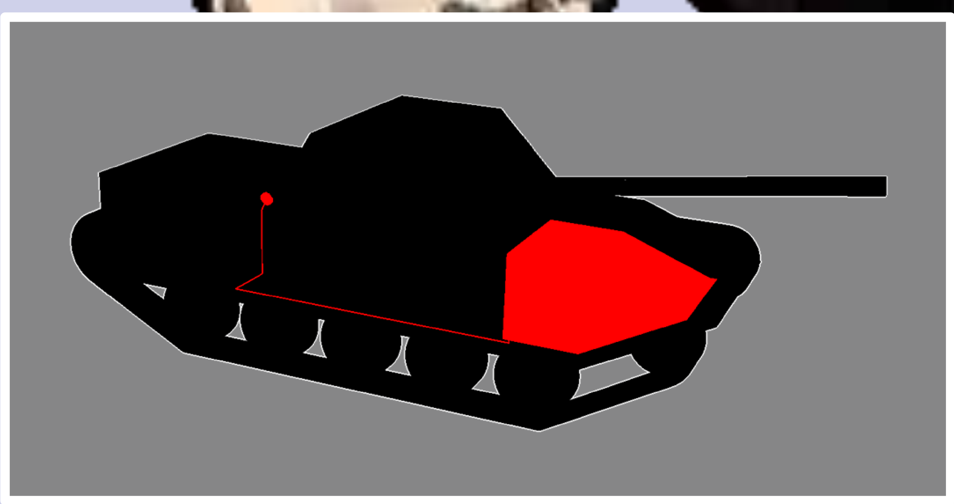
The fast and accurate rendering of transparent objects is an open problem in computer graphics as it necessitates expensive fragment sorting on the GPU. We present initial findings in optimizing existing order-independent transparency (OIT) algorithms by reducing costly global thread synchronization. We leverage these improvements in a novel application of OIT to ballistic simulations used in vulnerability/lethality (V/L) analysis software.

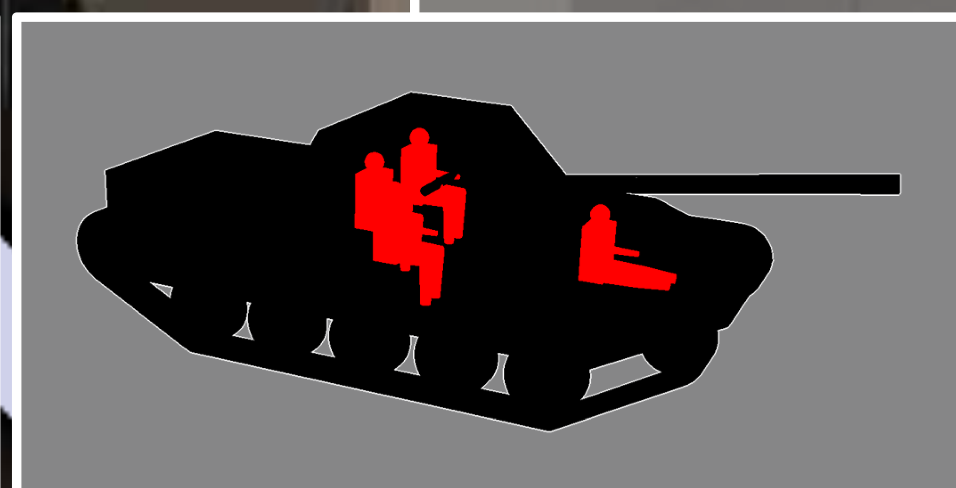



**Left:** Shaders store fragments in a global buffer. **Right:** Once all fragment information has been stored, it can be sorted for shading computations.

## Per-Pixel Linked Lists for OIT

New API features allowing for atomic memory operations and arbitrary memory access in shaders enable GPU-optimized dynamic data structures such as per-pixel linked lists (PPLL) [Yang et al. 10]. The figure above illustrates the construction of PPLL, which uses a single global atomic counter to allocate memory in the global fragment buffer.

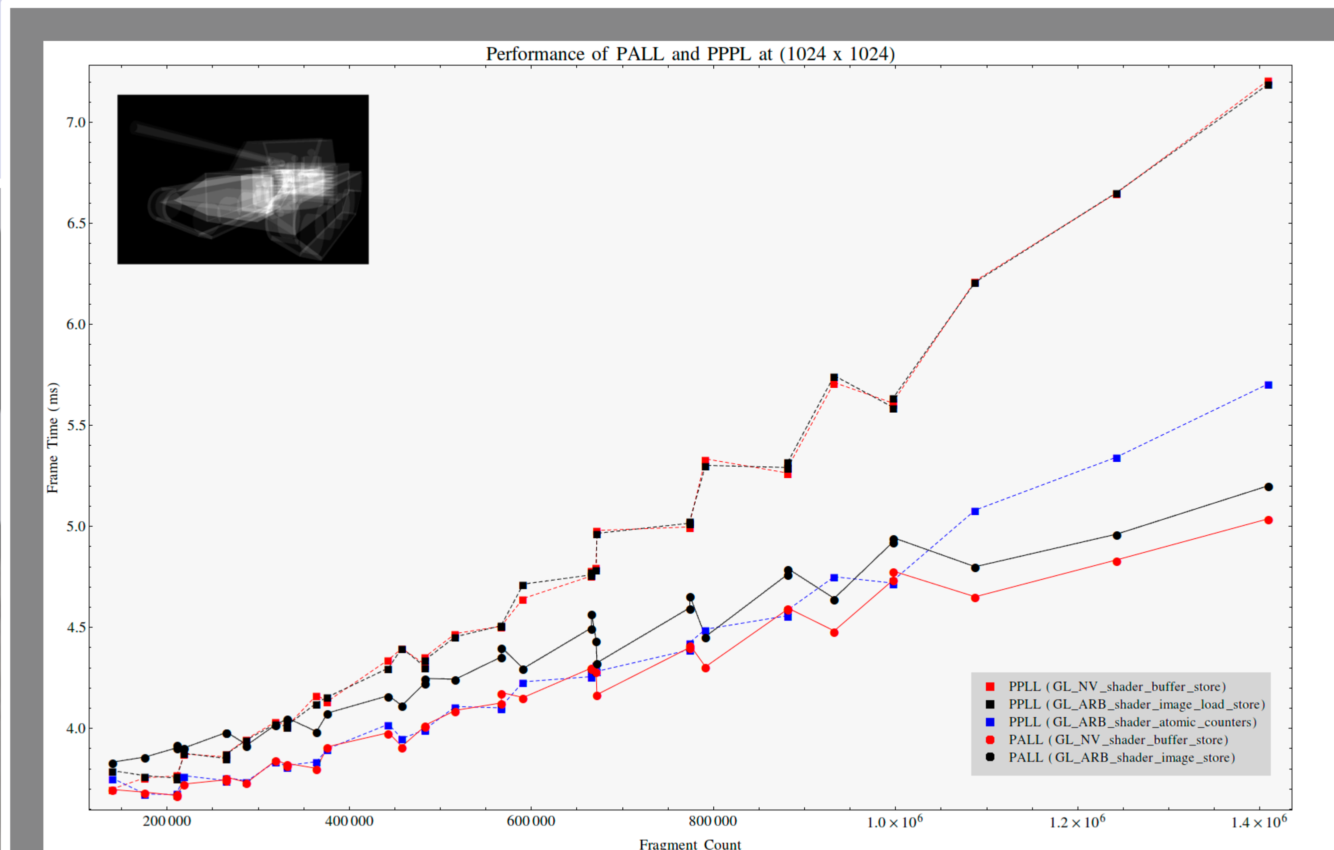




## V/L Analysis

V/L analysis uses ballistic simulations to evaluate threat-target interactions and plan live-fire testing. Although ballistic simulations were previously too costly for real-time applications, by applying state-of-the-art OIT algorithms from computer graphics, real-time ballistic simulations are now within reach.

In particular, optical transparency computes the light absorbed as photons pass through the environment, whereas ballistic simulation computes the energy absorbed as projectiles pass through an object. In each case, absorbance is a function of object thickness. Optical transparency, for instance, computes absorbance using Beer's Law [Suffern 07]. Ballistic penetration also follows a similar type of exponential decay; however, the penetration equations are derived empirically and are typically functions of material properties and threat parameters including size, weight, and velocity [Butler and Stephens 07].



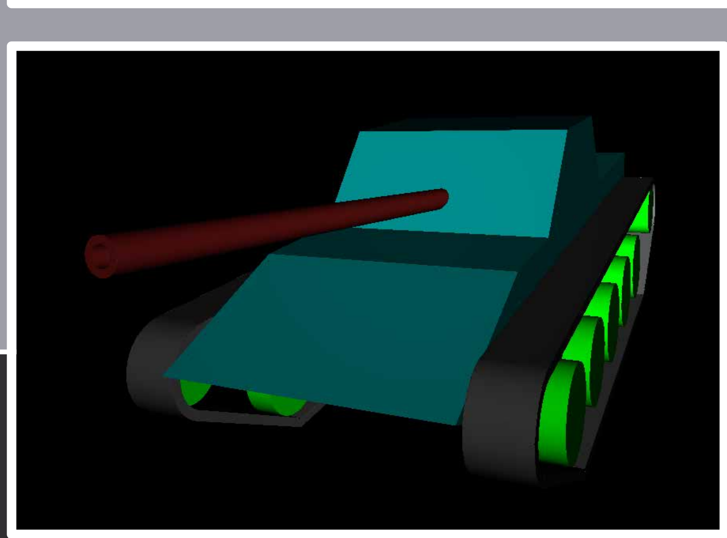

The performance of PALL and PPLL on an NVIDIA GeForce GTX 690 using OpenGL 4.3. When PALL and PPLL use the same counters, PALL is consistently faster than PPLL. Furthermore, PALL performance is on par with PPLL when PPLL uses the fastest atomic counters. Unfortunately, PALL cannot use the fastest atomic counters as they do not provide atomic add operations, only atomic increment.

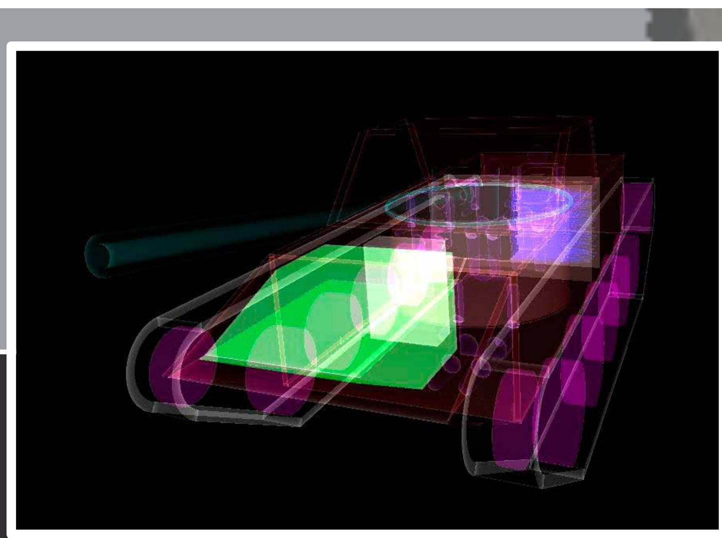
## Reducing Global Contention

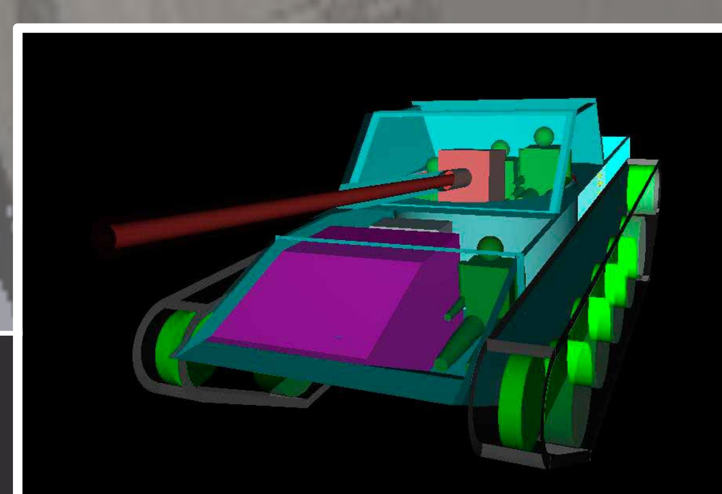
We reduce the need for global atomic synchronization by assigning regions of memory for each primitive rather than for each fragment. In our Primitive Allocated Linked List (PALL), the geometry shader allocates memory for each primitive. Specifically, the screen space bounds of each primitive provide a maximum number of fragments that can be generated. Fragment shaders then construct the linked list while using a memory address passed from the geometry shader. As a result, *this approach distributes the cost of atomic operations across the fragments generated by each primitive.*

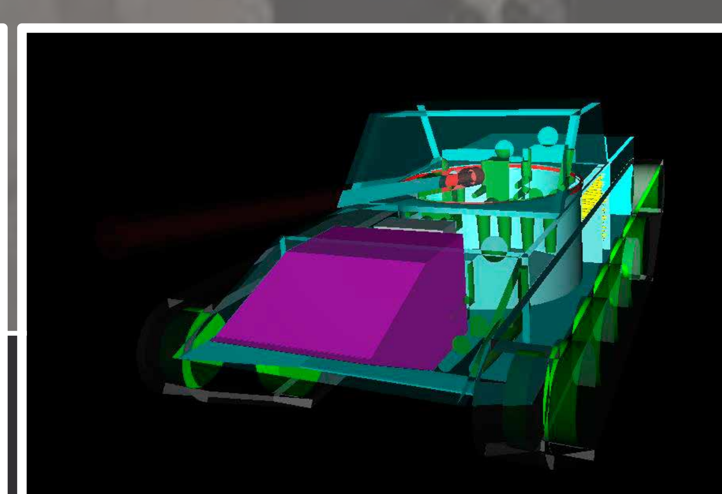
## Future Work

As we move toward our goal of real-time ballistic simulations, we will perform a comprehensive analysis of modern OIT algorithms in this context. Additionally, we will compare the performance and accuracy tradeoffs between raster-based transparency and ray tracing before moving our work from research prototype to production-grade software.









THE UNIVERSITY OF IOWA

ARL

### References

[Butler and Stephens 07] L. Butler & A. Stephens. Bullet Ray Vision. In *IEEE Symposium on Interactive Ray Tracing*, 2007.

[Suffern 07] K. Suffern. Ray Tracing From the Ground Up. A K Peters, Ltd. Wellesley, MA, 2007.

[Yang et al. 10] J. Yang, J. Hensley, H. Grün, & N. Thibieroz. Real-Time Concurrent Linked List Construction on the GPU. *Computer Graphics Forum*, 29:1297–1304, 2010.