Unlocking the Full Potential of the Cray XK7 Accelerator

Mark D. Klein*, and John E. Stone†
*National Center for Supercomputing Application, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA
†Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

***Abstract*—The Cray XK7 includes NVIDIA GPUs for acceleration of computing workloads, but the standard XK7 system software inhibits the GPUs from accelerating OpenGL and related graphics-specific functions. We have changed the operating mode of the XK7 GPU firmware, developed a custom X11 stack, and worked with Cray to acquire an alternate driver package from NVIDIA in order to allow users to render and post-process their data directly on Blue Waters. Users are able to use NVIDIA's hardware OpenGL implementation which has many features not available in software rasterizers. By eliminating the transfer of data to external visualization clusters, time-to-solution for users has been improved tremendously. In one case, XK7 OpenGL rendering has cut turnaround time from a month down to to just one day. We describe our approach for enabling graphics on the XK7, discuss how the new capabilities are exposed to users, and highlight their use by science teams.**

***Keywords*-GPU computing; High-performance computing; Scientific visualization; Parallel rendering;**

I. Introduction

The Cray XK7 includes NVIDIA GPUs for accelerating high performance computing applications, but the standard Cray-supported system software currently limits their use to compute-specific tasks, effectively locking out acceleration of OpenGL and other graphics-specific tasks and functions. We have changed the operating mode of the XK7 GPU firmware, developed a custom X11 stack, and worked with Cray to acquire an alternate driver package from NVIDIA in order to allow Blue Waters users to render and post-process their data directly on the system. Users of Blue Waters are able to take advantage of NVIDIA's full hardware-accelerated OpenGL implementation directly on the XK7 compute nodes. NVIDIA's OpenGL provides many features not available in software rasterizers such as Mesa, including interoperability APIs for efficient in-place access to GPU-resident data by CUDA, OpenCL, the OptiX ray tracing framework, and the XK7 GPU's on-board NVENC video encoding and decoding accelerator hardware. By enabling local XK7 rendering and eliminating the bulk transfer of data to external visualization clusters, the workflow for visualizing the results of petascale simulations becomes much simpler and the time-to-solution for users has been greatly reduced. These advancements have proven critical for the creation of advanced visualizations of the HIV-1 virus capsid and other large biomolecular complexes with VMD[1] [1], [2], [3], [4], and for high fidelity movie renderings with the HVR volume rendering software.[2] In one example, the total turnaround time for an HVR movie rendering of a trillion-cell inertial confinement fusion simulation [5] was reduced from an estimate of over a month for data transfer to and rendering on a conventional visualization cluster down to just one day when rendered locally using 128 XK7 nodes on Blue Waters. The fully-graphics-enabled GPU state is currently considered an unsupported mode of operation by Cray, and to our knowledge Blue Waters is presently the only Cray system currently running in this mode.

Historically the analysis and rendering of large data sets has often required visualization clusters separate from the systems used to generate the data. Blue Waters does not currently provide a separate GPU-accelerated visualization cluster for users as part of the same facility, so in cases where GPU-accelerated visualization was needed, users were required to transfer their data to visualization systems external to Blue Waters and NCSA. With the extremely large size of data generated by the Blue Waters system, such off-site transfers and post-processing can take weeks or longer. After aforementioned modifications were made to the Blue Waters system software and to the XK compute nodes to unlock their full potential, users of the Blue Waters system are now encouraged to use the 4,224 Tesla K20x GPUs not only for computation and generation of their science data, but also to perform large scale analysis and visualization on it. By keeping these visualization tasks in-house, network traffic is greatly reduced and results are delivered to users much more rapidly. In many cases, users can combine their analysis and visualization into a single-pass, more efficiently operating on massive datasets, which has the added benefit of avoiding redundant I/O, providing a benefit even if a facility also has separate co-located visualization clusters available.

By fully enabling the XK7 accelerators for both graphics and compute usage, new opportunities are created for performing high-performance hardware video encoding and decoding for remote visualization. This also allows for efficient encoding of high-resolution "4K" resolution movies and various types of image processing and computer vision algorithms that exploit OpenGL rasterization hardware and software. These capabilities are not available when the GPUs are run in the compute-only mode as shipped by Cray.

[1] http://www.ks.uiuc.edu/Research/vmd/

[2] http://www.lcse.umn.edu/MOVIES

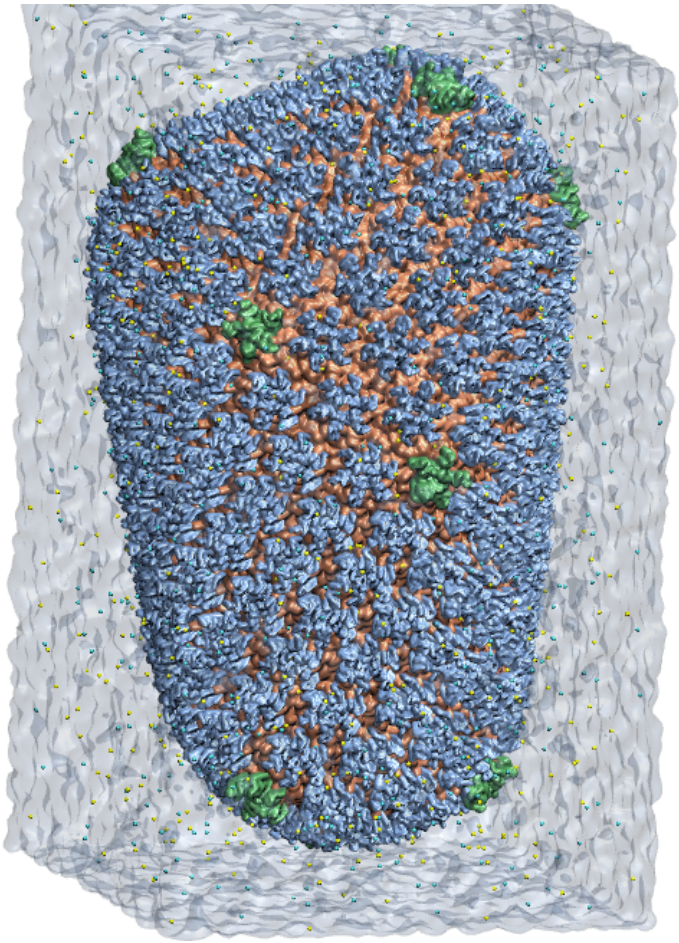

Figure 1. VMD GPU-accelerated OpenGL rendering of the 64-million-atom HIV-1 capsid simulation [2] performed on the NVIDIA Tesla K20X GPUs of the Cray XK7 compute nodes. This view shows the extent of the molecular dynamics simulation unit cell with solvent, ions, and multiple surface representations highlighting components of the HIV-1 capsid. The OpenGL rendering used GLSL programmable shading for pixel-rate lighting and direct ray casting-based rasterization of spheres used to represent atoms.

Below we describe all of the steps required to change the operating mode of GPUS and the X11 modifications needed to enable OpenGL acceleration on the Cray XK compute nodes. While these changes help accelerate time-to-solution and create additional research opportunities; one must also be wary of any additional load, instability, or overhead these changes might introduce to the system, particularly anything that would affect traditional compute-only codes that do not make use of OpenGL rasterization or other graphics-specific GPU hardware or software features. We address these concerns by reporting our experiences operating Blue Waters XK7 nodes in the graphics-enabled state for over a year, and note the lack of impact on performance, system reliability, power consumption, and other dimensions of potential concern.

II. Blue Waters System Software

A. System Overview

Blue Waters is a hybrid Cray XE/XK system consisting of 22,640 XE6 and 4,224 XK7 compute nodes. Each of the 4,224 XK7 compute nodes contains an NVIDIA Tesla K20X GPU for acceleration of computational codes. The stock Cray system configuration enables the Tesla K20X GPU accelerators used for computational purposes, but does not enable the GPUs to be used for OpenGL rasterization despite the fact that GPU hardware is designed principally for this purpose. The desire to enable applications to use native hardware-accelerated OpenGL on the XK7 GPUs led to a number of system software changes that we describe below.

B. System Changes

Three issues block applications that use OpenGL from running by default on the Cray XK GPU nodes. The following steps will enable OpenGL to run on the XK GPU nodes with full GPU hardware acceleration.

1) OpenGL requires an X Server: The NVIDIA OpenGL implementation currently requires an X server to be running, even when rendering to an off-screen OpenGL pbuffer. This stems from a reliance on the use of X server infrastructure for bootstrapping the graphics side of the GPU driver software, and because there is currently not an NVIDIA driver stack that allows an OpenGL context to be obtained except by the use of the GLX API, which was designed to be tied to an X11 windowing system. Sun Microsystems developed a cluster-oriented OpenGL rendering extension called “GLP” which provided a means of creating GPU-accelerated OpenGL contexts without running an X server, but to our knowledge this extension was unique to Sun Microsystems hardware and software, and it has not been implemented by other vendors. It is possible that the use of the so-called “EGL” embedded OpenGL API may be a good starting point for off-screen rendering without a windowing system dependency, as NVIDIA has recently begun to support EGL on Linux platforms, but this currently remains an area of future work.

The requirement for an X server creates a problem on the Cray system as the compute nodes run incredibly minimalistic and stripped-down system software. All unneeded kernel drivers, kernel modules, and other software components are removed from the kernel and Linux environment to make the compute node system software lean, reliable, and performant. The major obstacle to starting X on an XK7 compute node is the lack of Virtual Terminal (CONFIG_VT) support in the compute node kernel which is required in the Linux initialization routines. There appeared to be two ways to get around this: recompiling a kernel with the VT support enabled, or modifying X to work without Virtual Terminal support.

Building a custom kernel, while technically possible, creates too many uncertainties with package updates, and

may cause performance effects beyond just enabling the X server. This idea was evaluated and scrapped, and the Xorg 1.12.4 release was downloaded and the source code was examined. It turns out that bypassing the VT calls in the X Server's initialization routines is incredibly easy, because there are only a handful of VT-related calls in the X server initialization code. The VT-related calls are not needed when running in a headless configuration, so for the needs of the Cray user community these VT-related calls can safely be removed. The functions `xf86OpenConsole()`, and `xf86CloseConsole()` simply need to return early[3] and everything will start correctly. An entire Xorg 1.12.4 release was recompiled from source [4] and put into `/usr/local/X11R7.7` which allowed us to avoid conflicts with existing libraries on the system. A headless `xorg.conf`[5] was created to load the NVIDIA driver.

2) OpenGL requires additional driver components: The next missing piece in the puzzle arose from a missing hardware-accelerated NVIDIA driver. In a normal distribution, the `nvidia-drv.so` X driver and associated libraries are included with the kernel module.[6]

These files were missing from the Cray distribution as they are unneeded except for X. The NVIDIA kernel driver version on a Cray is custom to the Cray and in the case of a driver/kernel module version mismatch, the driver will not load. A request was filed with Cray to provide the missing files, and they were quickly provided. All future NVIDIA updates on our system have included the previously missing driver components, allowing us to keep the driver in sync with the kernel module through a simple script to update links to version dependent files. [7]

3) Cray accelerators run in a non-standard GPU Operation Mode: Finally, even with the above pieces in place, the Xorg server still refused to start. The error message received was "The GPU Operation Mode for this GPU disallows graphics". This error indicated that the nodes are shipped by Cray in a state where the graphics capabilities of the GPUs are disabled.[8] The NVIDIA supplied `nvidia-smi` tool can be used to adjust the GPU Operation Mode to set the GPU back to the default to "ALL_ON". This change requires a system reboot to take effect. With the GPU Operation Mode bit flipped, and the system rebooted, our modified Xorg 1.12.4 server started on a compute node. It took a few attempts to get all of the accelerated libraries moved into the proper paths and to set environment variables, but a test application[9] then ran successfully.

[3]http://raw.github.com/mdklein/XonXK/XonXK.patch

[4]http://www.x.org/wiki/Building_the_X_Window_System/

[5]http://raw.github.com/mdklein/XonXK/xorg.conf

[6]Refer to the "installed components" section of the NVIDIA GPU driver documentation.

[7]http://raw.github.com/mdklein/XonXK/fixnvdriver.sh

[8]Refer to NVIDIA documentation for the `nvidia-smi` GPU Operation Mode setting.

[9]http://raw.github.com/mdklein/XonXK/testglx.cxx

This is technically the only modification that was made to the general operation of the system as it changes the operating mode of a hardware component. The other two changes were all just modifications to the software environments. NVIDIA and Cray were contacted to investigate this change to make sure it was safe for our hardware. NVIDIA reported a possible power draw increase of around 2 Watts per GPU while running in the OpenGL mode, which was an acceptable number. Cray agreed that the change to the GPU Operation Mode would not harm the hardware, but this pushed the system into the "unsupported" designation. If problems with GPUs turned up after switching the GPU Operation Mode, we accepted that we might need to switch the GPU Operation Mode back to "COMPUTE_ONLY" to debug the system. Since enabling this mode on March 13, 2013, we have had zero issues relating to the change, and we have never needed to change it back.

C. Scheduler Modifications

All of the above work was done running manually as root on a compute node. For users to benefit from the work, a method needed to be developed that worked with our job scheduling system. OpenGL applications require an X server to be running, but having Xorg running all the time would negatively affect users that had no use for it. Various methods for starting X were investigated with the aim of finding solutions that would be easy to use, but that would only affect users that wanted the X server.

Blue Waters uses Torque and Moab for its resource manager and workload scheduler, respectively.[10] The way CCM sets up the environment was examined, as it did things similar to those that we needed for X server initialization, such as starting up SSH servers on the compute nodes. This was done by the Torque prologue. We chose to create a *generic resource* in Moab we called "viz". We then added some logic to the prologue of Torque specifying that if a viz resource was requested, `pcmd` from the Cray `nodehealth` package would start up Xorg on the XK nodes. A 10 second sleep was added when starting viz jobs to ensure the Xorg server was completely started before an `aprun` was attempted. The prologue runs as root before the job starts completely outside of the `alps` scheduler, and `pcmd` allows for backgrounding jobs making the X server initialization completely transparent to the user. The epilogue was also modified to kill the X server so nodes are cleaned up properly after jobs were finished. This generic resource method allows for accelerated OpenGL applications in both ESM and CCM modes, allowing flexibility to meet needs of the science teams.

The goal of this project was to make hardware-accelerated OpenGL support completely optional, but also easy to use by those that want it. A loadable *module*[11] was created to easily

[10]https://bluewaters.ncsa.illinois.edu/running-your-jobs

[11]http://modules.sourceforge.net/

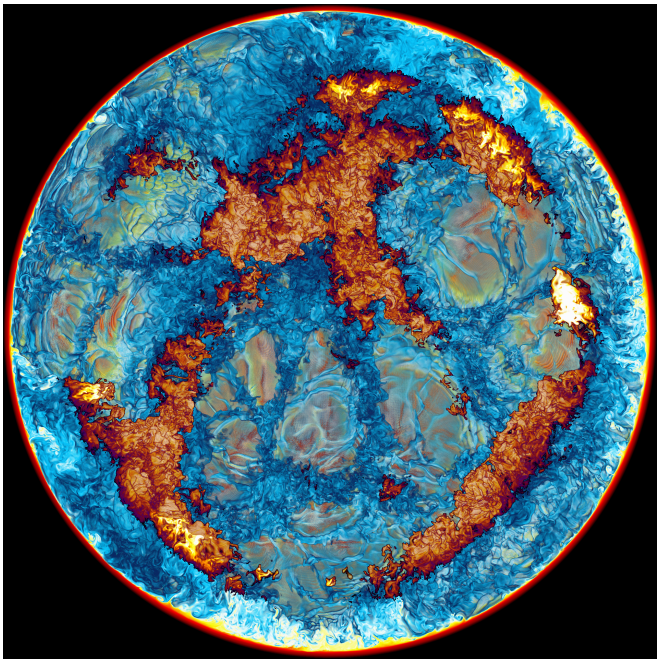

Figure 2. In this image from the PRAC team led by Paul Woodward at the University of Minnesota, team member Mike Knox has visualized the process of entrainment of hydrogen-rich gas into the helium shell flash convection zone of the very late thermal pulse star called Sakurai's object. The front half of the star has been cut away, and the central degenerate carbon-oxygen core which will ultimately become a white dwarf star, has been made transparent. Just above this carbon-oxygen core, helium burns in a shell at a radius of about 9000 km, generating about 40 million solar luminosities. This drives vigorous convection of the helium and carbon mixture above the helium burning shell extending up to the top of this convection zone at about 18000 km. The helium-carbon mixture of the convection zone has also been made transparent. Only mixtures of this gas with entrained hydrogen-helium gas from above the convection zone is made visible in this volume rendering. Concentrations of entrained gas from large to small range in color from red to yellow, white, aqua, and finally dark blue. Superposed on this image of the entrained gas concentration is an image of the rate of energy release from burning of the entrained hydrogen. This burning proceeds only at depths in the convection zone where the temperature is high enough, between radii of about 15000 to 12500 km. The burning gas is shown with very dark blue representing slowest combustion, then red, yellow, and finally white for most rapid combustion at the greatest depths. The localized nature of this energy release is apparent from this image. It ultimately leads, as the combustion causes the entrainment rate to increase, to global oscillations of the shell hydrogen ingestion and burning, to which this team gives the acronym GOSH. A movie showing this hydrogen ingestion phenomenon can be found on the Web: http://www.lcse.umn.edu/MOVIES

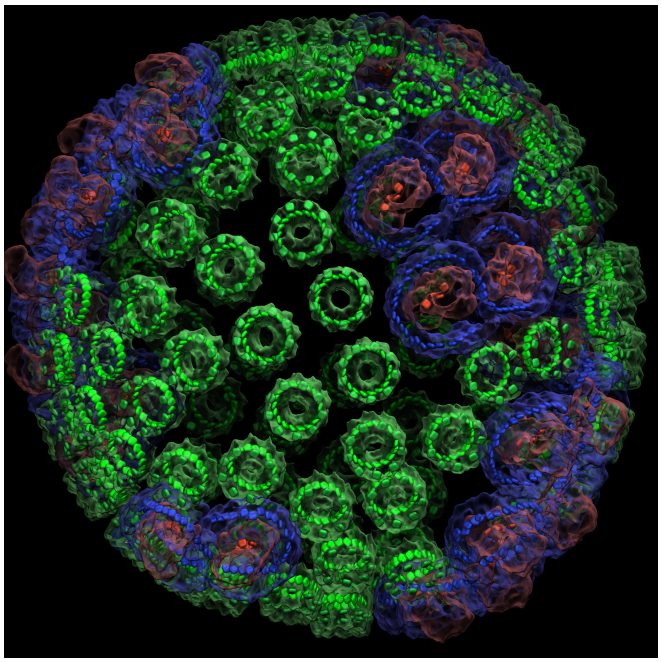

Figure 3. Spherical chromatophore from purple photosynthetic bacterium *Rhodobacter sphaeroides* showing the placement of light harvesting complexes as determined by microscopy and computational modeling [6]. This VMD rendering of the chromatophore structure used GPU-accelerated OpenGL rendering on Blue Waters and incorporated custom GLSL shaders for pixel-rate lighting, angle-dependent transparency, and depth cueing. The use of sophisticated GLSL shaders is a particular area where the NVIDIA GPU-accelerated OpenGL implementation significantly outperforms software rasterizers such as Mesa, both in terms of speed and often also in terms of compatibility with a wide range of OpenGL shading language versions and features. By performing rendering on Blue Waters itself, researchers can completely eliminate transfers of large datasets to other sites. The performance gain achieved just as a result of eliminating large data transfers can often dwarf all other considerations.

set up the paths to the libraries, and all needed environment variables. To compile with the OpenGL accelerated libraries, one simply loads the `opengl` module.[12] If the user wants to use the included Mesa software rasterization library provided by Cray, they can simply skip loading the new `opengl` module.

III. Applications

Two NSF PRAC science teams have begun to use their applications with the hardware-accelerated OpenGL capability made available on Blue Waters by the techniques described above. The first two applications to use the new capability were HVR[13], a sophisticated volume renderer used for visualizing a wide variety of computational fluid dynamics simulations such as the visualization of stellar combustion [7], [8] shown in Fig. 2, and VMD[14] [1], a tool for preparing, visualizing, and analyzing molecular dynamics simulations such as the HIV-1 capsid [2] shown in Fig. 1, and the chromatophore structure shown in Fig. 3.

VMD has been adapted to support off-screen OpenGL rendering for high-throughput visualization of petascale molecular dynamics simulations [3], such as the HIV-1 capsid [2] shown in Fig. 1, and a chromatophore [6] shown in

[12] http://raw.github.com/mdklein/XonXK/opengl.module
[13] http://www.lcse.umn.edu/hvr/hvr.html
[14] http://www.ks.uiuc.edu/Research/vmd/

Fig. 3. The process of modifying an existing OpenGL application for off-screen rendering is relatively straightforward, and involves the use of a special set of GLX APIs to create and manage off-screen pbuffers. VMD is written in C++ using a set of `DisplayDevice` and `OpenGLRenderer` subclasses that adapt specialize the rendering subsystem for a variety of OpenGL rendering hardware and windowing systems [9]. With the existing VMD `DisplayDevice` class hierarchy in place, the development of a new subclass for off-screen OpenGL pbuffer rendering required only about three hours of work to write and test. With the new off-screen rendering mechanism added to VMD, we immediately began to evaluate its performance, and added support for it within the parallel movie rendering tools included in VMD [3].

From the beginning it was clear that the availability of GPU-accelerated OpenGL would be beneficial for speeding up large scale VMD movie rendering tasks on Blue Waters, both for daily visualization tasks and as a rapid method for previewing renderings and movies to be created more computationally costly ray tracing and ambient occlusion lighting techniques. An unexpected benefit of GPU-accelerated OpenGL is that the rendering speed is so fast relative to higher quality methods that a user can render images and movie frames using both OpenGL and ray tracing with a negligible increase in rendering time as compared with ray tracing alone. A secondary benefit of using GPU-accelerated OpenGL, as compared with software rasterizers such as Mesa, is that interoperability APIs can be used to improve efficiency of data exchange between compute kernels written in CUDA and OpenCL, OpenGL-based visualization algorithms, high quality ray tracing with OptiX, and hardware-accelerated video encoding and decoding with NVENC hardware present on the XK7 GPUs. By using interoperability APIs, data can remain resident in GPU global memory between computation and visualization, thereby eliminating costly copy operations.

Another important benefit of support for the NVIDIA GPU-accelerated OpenGL implementation and the associated X server usage is that they allow a wide range visualization tools to be developed using roughly the same approach that would be used for a conventional desktop workstation, thereby eliminating a potential barrier to the use of many tools on petascale Cray systems such as Blue Waters. The availability of full OpenGL greatly eased the development of the OptiX-based GPU-accelerated ray tracing features in VMD on Blue Waters [4]. The OptiX libraries contains internal references to OpenGL GLX APIs such as `xcb_glx_set_client_info_arb()`, which are unresolved symbols on the standard Cray XK7 system software. While it has subsequently been possible to work around the missing OpenGL GLX symbols by creating no-op stub functions within VMD, this is clearly an undesirable situation. The stub function approach had to be used to support OptiX-based GPU-accelerated ray tracing on the Titan machine at Oak Ridge National Laboratory, and on the Big Red II machine at Indiana University. The use of stub functions is fragile since OptiX or any similar library might incorporate additional internal OpenGL dependencies in future versions, and there could be dependencies on OpenGL or GLX APIs that would be too difficult to emulate effectively. By providing a complete OpenGL and X11 software stack for the XK7 nodes, it is much easier to compile existing visualization tools and to exploit the full performance potential offered by the XK7 GPUs.

IV. Conclusions

We have presented an overview of the system software changes that were required in order to enable full GPU-acceleration of OpenGL rasterization on the Cray XK compute nodes containing NVIDIA Tesla K20X GPUs. Our experience has shown that the required change to the GPU Operation Mode to enable graphics has had no observable impact on the reliability of the Blue Waters system, and that GPU power consumption and performance for non-graphical applications remains the same as it was before the change. Several science teams have begun to use the XK7 nodes on Blue Waters for the purposes of high-performance in-place visualization of large datasets on Blue Waters, as opposed to transfer to off-site visualization clusters. The implementation of a new `opengl` software module and the close integration of these features with the Blue Waters scheduler implementation makes it very easy for users to exploit the GPU-accelerated rendering capability without having to compile their own X server software, nor to add complexity to their existing batch scripts. While the changes we have described are currently not a Cray-supported mode of system operation, they have proven very beneficial to Blue Waters science teams, and we expect that more Cray sites will wish to replicate the features we have described here on their own systems.

Acknowledgments

This research is part of the Blue Waters sustained-petascale computing project supported by NSF award OCI 07-25070, and “The Computational Microscope” NSF PRAC award. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications (NCSA). The authors wish to acknowledge support of the CUDA Center of Excellence at the University of Illinois, and NIH funding through grants 9P41GM104601 and 5R01GM098243-02. The authors wish to thank Dave Semeraro of NCSA, and Paul Woodward and Mike Knox of the University of Minnesota for their involvement in developing, using, and evaluating the fully GPU-accelerated OpenGL rasterization capabilities implemented on Blue Waters.

REFERENCES

[1] W. Humphrey, A. Dalke, and K. Schulten, "VMD – Visual Molecular Dynamics," *J. Mol. Graphics*, vol. 14, pp. 33–38, 1996.

[2] G. Zhao, J. R. Perilla, E. L. Yufenyuy, X. Meng, B. Chen, J. Ning, J. Ahn, A. M. Gronenborn, K. Schulten, C. Aiken, and P. Zhang, "Mature HIV-1 capsid structure by cryo-electron microscopy and all-atom molecular dynamics," *Nature*, vol. 497, pp. 643–646, 2013.

[3] J. E. Stone, B. Isralewitz, and K. Schulten, "Early experiences scaling VMD molecular visualization and analysis jobs on Blue Waters," in *Proceedings of the XSEDE Extreme Scaling Workshop*, 2013.

[4] J. E. Stone, K. L. Vandivort, and K. Schulten, "GPU-accelerated molecular visualization on petascale supercomputing platforms," in *Proceedings of the 8th International Workshop on Ultrascale Visualization*, ser. UltraVis '13. New York, NY, USA: ACM, 2013, pp. 6:1–6:8.

[5] P. R. Woodward, J. Jayaraj, P.-H. Lin, M. Knox, D. H. Porter, C. L. Fryer, G. Dimonte, C. C. Joggerst, G. M. Rockefeller, W. W. Dai, R. J. Kares, and V. A. Thomas, "Simulating turbulent mixing from richtmyer-meshkov and rayleigh-taylor instabilities in converging geometries using moving cartesian grids," in *Proceedings NECDC2012*, October 2013.

[6] J. Strumpfer, M. Sener, and K. Schulten, "How quantum coherence assists photosynthetic light harvesting," *J. Phys. Chem. Lett.*, vol. 3, pp. 536–542, 2012.

[7] P. R. Woodward, F. Herwig, and P.-H. Lin, "Hydrodynamic simulations of H entrainment at the top of He-shell flash convection," *Pre-print*, 2013, arXiv: 1307.3821.

[8] F. Herwig, P. R. Woodward, P.-H. Lin, M. Knox, and C. L. Fryer, "Global non-spherical oscillations in 3-D 4p simulations of the H-ingestion flash," *Pre-print*, 2014, arXiv: 1310.4584.

[9] J. E. Stone, A. Kohlmeyer, K. L. Vandivort, and K. Schulten, "Immersive molecular visualization and interactive modeling with commodity hardware," *Lect. Notes in Comp. Sci.*, vol. 6454, pp. 382–393, 2010.