Study of Commodity VR for Computational Material Sciences
Katsumi Hagita,*†‡ Shigenori Matsumoto,*‡ and Koji Ota§

†Department of Applied Physics, National Defense Academy, 1-10-20 Hashirimizu, Yokosuka 239-8686, Japan
‡Research & Development Group, Hitachi, Ltd., 832-2, Horiguchi, Hitachinaka, Ibaraki 312-0034, Japan
§Advanced Technology Research & Development Center, Hitachi Chemical Co. Ltd., 48 Wadai, Tsukuba, Ibaraki 300-4247, Japan

ABSTRACT: Recent advancements in virtual reality (VR) devices and software environments make it possible to easily incorporate this technology for many applications, including computational materials science. For studying three-dimensional (3D) structure models and related chemical information, we focused on using a commodity VR device (VIVE) and an authoring tool (Unity). To visualize 3D chemical structures, disturbances like judder due to dropped frames should be eliminated from the VR experience to improve simulations. We propose a simple evaluation method that is straightforward for the nonexpert or novice VR user. We examine the major visualization representations including ball, ball and stick, and isosurface systems. For systematic benchmark measurements, a pendulum from the VR device was used to generate periodic oscillatory motion during measurements of a time series in frames per second (fps). For VIVE with a refresh rate of 90 Hz, judder occurred when less than 90 fps. We demonstrated the system size limitations for the results of molecular dynamics simulations of phase separation of ABA block copolymers and experimental observations of filler morphologies in rubber.

1. INTRODUCTION

Until recently, immersive virtual reality (VR) equipment (e.g., cave automatic virtual environment (CAVE)) was very expensive.1,2 The advent of VR devices for various forms of entertainment3 has decreased cost and led to advancements employed in molecular sciences.4–15 Such devices are now widespread and affordable (e.g., ∼500 USD for HTC VIVE16 or Oculus Rift17). VR devices such as head-mounted displays (HMDs) provide stereoscopic images that follow the movement of the head in real time. Compared to conventional stereoscopic displays, VR provides immersive experiences with walk-through. That is, VR provides a fantastic three-dimensional (3D) voyage in nanoscale space with data from computer simulations and electron microscopies. Furthermore, software environments have been improved by the use of integrated tools such as Unity18 and Unreal Engine.19

These developments in VR can be utilized for training, presentations, and advanced research and development (R&D). In particular, the use of VR in computational materials science (CMS) makes it easy to recognize complicated charge distributions, atomic positions, and Fermi surfaces.20 Recently, the Novel Materials Discovery (NOMAD) Laboratory, funded by the European Union Horizon 2020 Framework Programme for Research and Innovation, reported the importance of VR in CMS.21,22 Accordingly, it is clear that immersive VR can be used by both academic researchers and industry developers to evaluate computational results.

As a technology, development of VR for CMS has received much attention that has been strongly focused on interactive molecular dynamics (MD) technology (e.g., molecular manipulation). Very recently, several inspiring and innovative reports have been published.18–21 Notably, for the interested reader, O’Connor et al.18 and Goddard et al.19 published excellent reviews of VR for MD simulations. Interactive MD technology is an intriguing technology for utilization in drug discovery as it has been examined by VMD23 with stereoscopic vision on a display using NVIDIA 3D Vision. Currently, interactive MD technologies are at the developmental stage of technical verification through proposing new frameworks and/or program codes. Unfortunately, this means that the average chemical researcher does not use interactive MD as a part of daily research activities.

Tasks required for VR in CMS are not limited to manipulation of molecules. For example, there is a strong interest in the observation of 3D morphology related to the structure–property relationship of polymer nanocomposites (PNCs). By utilizing VR as a tool, the time required to recognize a complicated 3D structure should be shortened considerably, benefiting both academia and the industry (e.g., R&D of materials). For aggregated materials such as PNCs, it is desirable...
to simply change the viewpoint and cross-section position using VR rather than through molecular or instrumental manipulations. Use of VR is therefore expected to surpass current techniques in use for simulation.

Although current research on VR technology focuses on functional implementation (e.g., molecular manipulation and demonstrations using small molecular systems), there are no recent reports on commodity VR for research on PNCs. In PNC systems, a large number of molecules form multiscale structures owing to self-organization and filler aggregation, with large-scale visualization targets. For VR of PNCs, difficulties related to practical system size range are currently being tackled in order to extract useful information. Furthermore, although chemical researchers would like to use VR to visualize objects to produce sleek images in publications using biomolecular software programs such as VMD, Pymol, or Chimera, there are no corresponding reports available to guide the novice. The focus of VR publications is on the implementation of VR visualization using new codes and algorithms; thus, information on general-purpose tools widely available to the general chemical researcher, who is not a VR specialist, is largely inaccessible. Although VR has become more accessible through commodity VR, a VR divide nevertheless has occurred; although chemists can easily use VR with modern desktop PCs, they tend not to do so because appropriate information regarding VR is lacking. Solving the VR divide is therefore of great importance for material chemistry research, including CMS.

In this study, we focus on simplified use of commodity VR using a VR model generated by VMD. Historically, VR is a research subject requiring advanced computer programming skills. Although users no longer need programming knowledge to use VR, they still need to have programming skills to display very large systems with moving objects. At a minimum, system size limitations are important to consider for use by VR nonexperts or novices. In the present study, we investigate the dependence of system size on VR performance of CMS simulation data. For researchers of chemistry and material sciences in particular, a simple evaluation method with less effort is desirable. Herein, we propose a simple performance evaluation method and present its effectiveness.

In the present paper, we employed commodity VR for CMS data using HTC VIVE, Unity, and VMD, followed by a simplified performance evaluation method to clarify the limits of system size for each type of visualization of CMS data. We also present results of performance benchmarks using recent middle-class graphics processing unit (GPU) cards. GPUs are frequently released with new products, and GPU performance continually improves. To know the relative performance of recent GPUs, the established 3DMark is a very useful benchmark. Since individual performance depends on visualization targets, we decided to focus on visualization of CMS data and evaluation of the performance of VR.

This article focuses on a simplified performance evaluation method that the average chemical researcher can perform at a reasonable cost. We show that the proposed evaluation method functions as a benchmark indicator by evaluating the performance based on system size dependence for several VR presentations. We present herein a simplified performance evaluation method accessible to the average chemical researcher who is not a VR expert and will not optimize the code of a VR program.

2. BACKGROUND ON PERFORMANCE PROBLEMS IN VR

As VR systems for large capacity processing are being developed, much research on rendering algorithms, data transfer, and real-time processing have been conducted. Parameters related to performance can be classified as "system size", "rendering method" (software), and "VR hardware". In the past, even with small systems, some problems occurred due to insufficient performance of VR hardware. For example, VR hardware problems consisted of "motion tracking" and "communication to distributed parallel-rendering computers," even for high-end devices such as CAVE, though these problems have now been solved. Today, fortunately, algorithms are adopted and modified when using HTC-VIVE, Oculus Rift, and other devices with software such as Unity and Unreal Engine. The development of fast and efficient rendering techniques is still quite important, requiring system size to be addressed when considering VR for CMS used by nonexperts or novices. The problem of "data transfer" from the central processing unit (CPU) to the GPU is regarded here as a problem within the system size parameter.

The historical technical development of VR has been reviewed in detail in the literature. Here, to contextualize this article, we provide a brief overview as follows. Processing resolution, image effect quality, CPU performance, and GPU performance are important considerations in VR development that have improved over time. For instance, in early versions of CAVE systems (e.g., those installed at Osaka University, the National Institute for Fusion Science, and the Earth Simulator in Japan), motion tracking and communication problems to dispersed rendering computers were often encountered prior to high-quality rendering. Within the past decade, such system problems have largely been resolved due to improvements in computing performance.

VR expressions originally used very light wire frames for rendering, which were then replaced with more realistic polygons. With each passing year, the limit of the number of objects that can be rendered has increased. This is due to improvements in both software and computer performance, though limitations were considered to originate from the rendering procedure. A general approach to render objects using a computer graphics library (e.g., OpenGL) relies on the CPU to generate object data consisting of vertices, faces, and textures and then transfers this object data with rendering instructions to the GPU. The GPU can then execute a rendering pipeline to draw objects on the display. When the transfer limit is exceeded, the transfer excess from each object slows the rendering speed. This rendering cost can be reduced by packing all operations into a single data transfer. To accelerate rendering operations on a GPU, a programmable shader (e.g., GLSL) enables handling of parallel processing through advances in general-purpose GPUs. Thus, a large number of polygons can be drawn instantaneously. Furthermore, the culling technique, in which polygons that disappear from sight are ignored, is effective at reducing rendering costs. According to the Unity document "Optimisation for VR in Unity," a feature such as occlusion culling, which omits rendering of polygons hidden behind the foreground, is an effective performance optimization method for VR. This technique is advantageous for construction of objects in 3D games (e.g., hidden polygons are almost the same even if a player moves a shoulder width). On the other hand, for condensed ball-stick systems (e.g., PNCs), the balls (spheres) and sticks...
(cylinders) cover less area behind objects, and thus polygons change drastically when moved very little. Therefore, VR walkthrough of condensed ball-stick systems suffers from problems associated with depth more than from details related to drawing polygons.

When the size of a system increases, problems result when drawing beyond the limits of data transfer. This is especially challenging for very large systems with moving objects, in which rendering processes become slower due to poor data-loading performance. As a result, frames are dropped, and dropped frames cause judder. High-level programming skills for high-performance data loading are required to address the issue. Even in the absence of object motion, judder can occur when the system size increases. It is important to prevent judder in presentations and other educational uses because it can cause motion sickness symptoms (e.g., eyestrain, headache, nausea, or vomiting).

In modern VR setups such as HTC VIVE and Unity, we expected that the factor causing judder would originate from polygon depth processing, as mentioned above. To improve rendering performance related to avoiding judder, much useful information can be found on the Unity website. However, this information may be too technical for the average chemical researcher. For nonexperts or novices of VR, system size of the VR model can be recognized as the primary index to control performance and to avoid judder. Therefore, it is beneficial to know system size limits quantitatively (i.e., number of polygons and vertices) in typical system configurations for each type of VR presentation. Moreover, for quantitative evaluation, it is desirable to have a performance evaluation method that can be easily implemented as a standard.

3. RESULTS AND DISCUSSION

We generated VR data with VMD and displayed it using VR on VIVE by using Unity. The possibility of using other VR devices (Oculus Rift and PSVR) is described in Supporting Information A. For the present benchmark measurements, we generated two types of data: ball-stick and isosurface. The present study examines the data size limits of various VR presentations. For the systematic benchmark measurements, we use the VR device as a pendulum (see details in Section 5.3) to generate periodic oscillatory motion during measurements of a time series in frames per second (fps).

3.1. VR Data Generated by VMD. 3.1.1. Visualization of Ball-Stick Data. In VMD, ball-stick systems can be drawn by using PDB, xyz format files, and LAMMPS trajectory files. In addition, the "Nanotube Builder" extension in VMD is useful for making atomic positions and bonds of multilayer graphene without positions required for the input data.

To investigate ball systems without sticks, we used the LAMMPS trajectory of a phase-separated system of ABA block copolymers with bead-spring simulations of the Kremer-Grest model. Details of the methods used for this simulation can be found in the literature. Here, we built a chain of eight A-segments, 48 B-segments, and another eight A-segments. The spherical aggregations of the A-segments were nearly identical in size because the same simulation parameters were used. We prepared four systems: (i) 256 chains of 64 atoms, (ii) 512 chains of 64 atoms, (iii) 768 chains of 64 atoms, and (iv) 1024 chains of 64 atoms. Snapshots of the system are shown in Figure 1. VDW style was used as the drawing method in the graphical representation. Here, we set the sphere scale to 0.4.

To probe ball-stick systems, we used the Nanotube Builder extension to make four configurations of multilayer graphene: (i) 12 096 atoms (12 layers of 5 nm × 5 nm), (ii) 24 192 atoms (24 layers of 5 nm × 5 nm), (iii) 32 256 atoms (32 layers of 5 nm × 5 nm), and (iv) 40 320 atoms (40 layers of 5 nm × 5 nm). Space-filling CPK style was used as the drawing method. Here, we set the sphere scale to 1.0. Snapshots are presented in Figure 2.

3.1.2. Visualization of General Isosurfaces. VMD can plot isosurfaces from field data in the AVS Field format as well as the Gaussian cube format. For the AVS Field format, however, VMD supports only uniform field types. In the AVS Field module in VMD, the order of description of the FLD file is restricted (refer to the sample file given in the Supporting Information). Here, we used 3D volume data of filler morphologies in rubber collected by using focused ion beam scanning electron microscopy (FIB-SEM). Details concerning the measurements are provided in the literature. We prepared three systems: (1) 192 × 192 × 192, (2) 256 × 256 × 256, and (3) 320 × 320 × 320 grids. Snapshots of these systems are shown in
Figure 3. Snapshots of examined 3D volume data of filler morphologies in rubber: (a) $192^3$, (b) $256^3$, and (c) $320^3$ grids. The volume fraction of the fillers was $\sim 20\%$. The data were gathered by using FIB-SEM.\textsuperscript{58}

Table 1. Specifications of the Produced VR Data

|                         | size (bytes) | polygons    | vertices     |
|-------------------------|--------------|-------------|--------------|
| ball (LAMMPS)           |              |             |              |
| (i) 16 384 atoms        | 413 210 617  | 2 097 152   | 6 291 456    |
| (ii) 32 768 atoms       | 826 443 537  | 4 194 304   | 12 582 912   |
| (iii) 49 152 atoms      | 1 239 653 997| 6 291 456   | 18 874 368   |
| (iv) 65 536 atoms       | 1 652 887 641| 8 388 608   | 25 165 824   |
| ball-stick (multilayer graphene) |        |             |              |
| (i) 12 layers           | 583 315 016  | 2 956 608   | 8 869 824    |
| (ii) 24 layers          | 972 260 361  | 4 928 024   | 14 784 072   |
| (iii) 32 layers         | 1 555 475 601| 7 884 632   | 23 653 896   |
| (iv) 40 layers          | 1 944 327 201| 9 855 704   | 29 567 112   |
| general isosurface (AVS field) |      |             |              |
| (i) $192^3$ grids       | 170 973 576  | 1 380 791   | 1 618 465    |
| (ii) $256^3$ grids      | 367 382 947  | 2 868 093   | 3 464 770    |
| (iii) $320^3$ grids     | 661 293 531  | 5 081 872   | 6 236 966    |

Figure 4. Frame rates for various systems with NVIDIA GeForce GTX 1060 and GTX 1070. (a) LAMMPS (balls), (b) multilayer graphene, (c) AVS fields with NVIDIA GeForce GTX 1060, (d) LAMMPS (balls), (e) multilayer graphene, and (f) AVS Fields with NVIDIA GeForce GTX 1070.

Figure 3. Isosurface style was used as the drawing method, with the isovalue set to 92.0.

3.1.3. Specifications of the Produced VR Data. We produced the VR data using VMD\textsuperscript{23} as listed in Table 1. The table also presents the specifications of the VR data. We directly counted the number of polygons and vertices in the “.obj” file because this information is absent. Since the polygons and vertices depend on the degree of sphere and bond division, we
used the default parameters: sphere resolution and bond resolution are 12 for the CPK or VDW style, and the step value is 1 for the isosurface style.

3.2. Evaluation of Data Size Limits of Various VR Presentations. We compared the system performances of two GPU cards: NVIDIA GeForce GTX 1060 and GTX 1070. Both GPUs have 6 GB memory. The GTX 1060 has 1280 CUDA cores and the GTX 1070 has 1920 CUDA cores. In the performance tests, one of each of these GPUs was installed in a personal computer with an Intel Core i5 (6600K processor, 3.50 GHz) and 32 GB memory, running Windows 10 Professional as the OS. We placed the center of mass of the displayed objects at a height of 150 cm from the origin. The area with $z < 0$ was cut by using the “Cross Section Shader” for comparison with dynamic cutting of displayed objects in the next subsection.

Figure 4 shows the measured time series of frame rates for various systems when using the GeForce GTX 1060 and GTX 1070. From Figure 4a, it is clear that the frame rate decreases as the number of atoms increases. For the case with 16,384 atoms, since the rate was near 90 fps, the influence of judder was considered negligible. In general, rendering lower than the refresh rate of the hardware used results in judder.\textsuperscript{31,32} The refresh rate of HTC-VIVE is 90 Hz. For the case with 32,768 atoms, since the rate was around 75 fps, the influence of judder was not negligible. For cases with 49,152 and 65,536 atoms, since the frame rate was around 45 fps, the influence of judder was clearly observed. To avoid judder, we enabled the “Interleaved reproject” function of SteamVR, which drops the frame rate to 45 fps by reprojecting every other frame. We found that the frame rates oscillated at a cycle of $\sim 2.1$ s. A smaller number of objects located in front of the HMD showed higher frame rates. Consistent with the ratio of 1.33 associated with the benchmark 3DMark,\textsuperscript{26} we confirmed that the frame rates of GeForce GTX 1070 were better than those of GeForce GTX 1060 (Figure 4d). Here, the average frame rates were compared between GeForce GTX 1060 and GeForce GTX 1070, and the ratio of fps from GTX 1070/GTX 1060 was calculated. For the case of 32,768 particles, the GeForce GTX 1060 rate was 70.86 fps and GeForce GTX 1070 rate was 89.58 fps, with a ratio of 1.26. When increasing the number of particles to 49,152, the GTX 1060 rate was 46.66 fps, whereas the GTX 1070 rate was 70.58 fps, with a ratio of 1.51. The latest GPUs, such as GeForce RTX 2080 and 2080 Ti, can achieve ratios relative to 3DMark\textsuperscript{26} for GeForce GTX 1060 of 2.1 and 2.33, respectively. We expect the threshold for the number of particles to increase when using the latest GPUs. Verification using more recent GPUs is outside the scope of this study, but it is a future task that we intend to pursue.

From Figure 4b,e, it is clear that the performance of multilayer grapheme is better than that of dense balls, although the numbers of polygons and vertices of the multilayer graphene were slightly larger than those of the dense balls. We expected the load from drawing dense objects to be larger than that from drawing sparsely distributed objects. To verify this, we tested cases with 2048 and 4096 chains of 64 atoms and generated VR data for A-segments only. Here, the A-segments numbered 32,768 and 65,536 atoms for 2048 and 4096 chains, respectively. Figure 5 shows the frame rates of densely and sparsely distributed balls of 32,768 and 65,536 atoms when using the NVIDIA GeForce GTX 1060. We concluded that the frame rate was smaller for denser objects.

From Figure 4c,f, it is clear that the frame rates from the isosurface of the filler morphologies observed by FIB-SEM\textsuperscript{56} were better than those from multilayer graphene for similar numbers of polygons because the filler morphologies contained fewer vertices.

3.3. Performance for Dynamic Cutting of Displayed Objects. We examined the performance from dynamic cutting of displayed objects by using the Unity asset Cross Section Shader. Dynamic and intuitive control of cross sections enables researchers to better understand computational results. Figure 6 shows the results of dynamic cross-section cutting of the examined VR data. Snapshots are presented in Figure 6a–c. The frame rates of these objects when using NVIDIA GeForce GTX 1060 and GTX 1070 are given in Figure 6d–i. We found that the ball-stick systems with dynamic cutting performed worse than those with static cutting (examined in the previous subsection). On the other hand, the isosurface systems with dynamic cutting performed better than those with static cutting. We believe that this difference is due to the substantially increased number of displayed polygons visible in the cross section. In general, performance depends on the rendering algorithm and shaders used. Although our results cannot directly extend to different shaders due to the complexity of shader implementation (including deferred rendering\textsuperscript{33}), analyses and considerations for the shader “OnePlaneBSP” used in the present study may provide a useful guideline. Studies to estimate the performance of different shaders and improve performance are currently in progress.

After cutting the displayed objects, part of the polygon mesh disappears from the display area. Although we expected that this decrease in polygons would result in an increase in the frame rate, the performance of cut objects was worse than that of the complete objects. To use Cross Section Shader, the shader program OnePlaneBSP should be applied to the objects (see Supporting Information D3).

A general rendering process mainly contains four operations: (1) transformation of vertex positions in the polygon mesh, (2) tessellation from those vertices, (3) rasterization of the generated triangles, and (4) computation of color and other attributes of each pixel. The surface shader in OnePlaneBSP automatically generates programmable shader codes for those operations. The shader checks whether each pixel lies in front of the cutting plane, $(p - a) \cdot n > 0$, where $p$ is the coordinate of the pixel, $a$ is the center of the plane, and $n$ is the normal vector of the cutting plane.\textsuperscript{34} When the pixel lies in front of the cutting plane, the pixel makes up the undisplayed objects as the attribute of the pixel in the operation (4). The calculation cost for pixel attributes increases with the number of polygons since each
To render cut objects, the surface shader in the shader program code OnePlaneBSP in Cross Section Shader (see Supporting Information D3) was used.

Here, the surface shader executed rendering processes (i.e., operations 1–4, above), with the calculation cost for pixel attributes depending on the number of polygons, as color and other attributes of each polygon are assigned to pixels. Although the surface shader program halted operations for pixels in parts of the cut object polygons, cost reduction was ultimately negligible. The reason for this is that the shader program performed shader operations on the pixels of all the polygons even though some of the polygons were hidden from view. Therefore, the rendering load does not decrease when cutting objects. Additionally, the interior of the objects observed in the cross-section view became filled-in white in the shader, as shown in Figure 6a–c. The reverse side of each polygon is generally ignored in the rendering process, and consequently, objects when viewed from the cross section seem to contain holes. This extra rendering cost causes a decrease in the frame rate when cutting objects.

4. SUMMARY

We demonstrated VR for CMS simulations using commodity VR devices. VR is an effective method not only for CMS simulations but also for chemical information and modeling in order to understand complicated 3D structures. In the present study, VR data were generated using VMD\(^3\) and displayed on the VIVE\(^4\) VR device via Unity. For CMS, we assessed the drawing performance of ball systems, ball-stick systems, and isosurface systems. The ball system data were prepared from bead-spring simulations of ABA block copolymer phase separations.\(^3\) The ball-stick systems were generated as multi-layered graphene by using the Nanotube Builder function of VMD. The isosurface systems were graphical representations of 3D volume data of filler morphologies in rubber observed by FIB-SEM.\(^3\) We systematically measured the frame rate for these systems while moving either the VR device or a handheld device as a pendulum.

We measured time series of frame rates for various systems using NVIDIA GeForce GTX 1060 and GTX 1070. For each condition, the frame rates of GeForce GTX 1070 were better than those of GeForce GTX 1060, reflecting the difference in the number of processing cores. The frame rate increased with the
number of polygons in each system. We found that the spatial density of objects also influenced the frame rates by comparing ball systems and ball-stick systems. In the ball systems, the performances of sparsely populated systems were better than those of densely populated ones with the same numbers of polygons. On the other hand, the isosurface system showed much better performance when containing the same number of polygons due to fewer isosurface vertices than other systems. For practical use, we measured the performance of dynamic cross-section cutting. The frame rates decreased because, in the rendering program, hidden polygons from cutting undisplayed objects were not discarded until the last rendering process. The frame rates also decreased as the polygons visible in the cross section increased. We concluded that for drawing objects without judder using GeForce GTX 1070, fewer than 10 million polygons and 30 million vertices should be used, except in dense ball systems. For systems filled with dense ball objects, fewer than 4 million polygons and 13 million vertices (corresponding to 32 768 particles) are sufficient for drawing objects without judder. Regarding the latest GPUs such as GeForce RTX 2080 Ti, the 3DMark26 performance was 2.33 times that of GeForce GTX 1060, which is the minimum requirement for operating HTC-VIVE. Because the performance ratio between GeForce GTX 1070 and GTX 1060 in this study agrees with that of the 3DMark benchmark, the threshold is expected to improve with newer GPUs. Incidentally, along with the performance improvement of the GPU, the number of pixels of HMD increased. Recently, HTC VIVE Pro was released with 2880 × 1600 pixels, which is 1.78 times that of HTC VIVE (2160 × 1200). We expect that fps performance of HTC VIVE Pro with GeForce RTX 2080 Ti is comparable to that of HTC VIVE with GeForce GTX 1060. Investigation of the performance of these systems in detail is a future task.

Finally, we concluded that the commodity VR equipment was useful for VR observations of the isosurface of filler morphologies with 3203 grid and dense particle systems with 32 768 particles. This shows that VR can be utilized for various research subjects of CMS, including PNCs.

5. METHOD

5.1. Preparation of VR Data by VMD. For practical use of VR for CMS, the method to make a VR model is similar to that used for 3D printing, which can also be regarded as a type of VR that produces a physical object in the process. Figure 7 illustrates the composition of software and hardware for practical application of VR to CMS. As shown, the VR-authoring tool directly controls the VR device, and VR-authoring tools generally require 3D VR data. Here, Wavefront OBJ, VRML, STL, and PLY are commonly used formats of VR data. The VR data are modeled using a VR modeler such as VMD or AVS/Express, which can handle CMS simulators such as VASP, TOMOBO, LAMMPS, and Gaussian. The VR formats and VR-authoring tools can also be used for 3D printers.

VMD is a molecular visualization program developed and maintained with NIH support by the Theoretical and Computational Biophysics group at the Beckman Institute, University of Illinois at Urbana-Champaign. VMD is mainly designed for modeling, visualization, and analysis of biological macromolecules such as proteins. It is also useful for modeling CMS simulation data, as shown in Figure 8. Note that VMD can read POSCAR, CHGCAR, XML (VASP), and Gaussian cube formats. In VASP, atomic positions and charge densities are recorded by POSCAR and CHGCAR, respectively. In VASP and higher, they are recorded in one XML format file, whereas in Gaussian, they are recorded in one file in the Gaussian cube format. An explanation of visualizing VASP data and Gaussian data is given in the Supporting Information. Besides atomic positions and charge densities, CMS studies are interested in the properties of Fermi surfaces. Here, to visualize Fermi surfaces by using VMD, BXSF format data must be converted into field data that is supported by VMD, such as the Gaussian cube format. Here, the BXSF format for XCrystDen is widely used to present Fermi surfaces in density functional theory simulations. Similar approaches are described in the NOMAD documents.

Using VR, atomic positions can be presented as ball-stick systems, whereas charge densities and Fermi surfaces can be represented as isosurface systems. VR data of CMS simulations such as VASP and Gaussian can be expressed as combinations of balls and sticks and isosurfaces. Other CMS simulations, such as classical MD simulations and mesoscale simulations, can also use balls, sticks, and isosurfaces. Examples of ball-stick data are presented in Figure 8e, whereas examples of isosurface data are presented in Figure 8d. Since we want to determine the limitations of the drawing process of large systems, we decided that it is important to know the drawing characteristics of ball-stick and isosurface systems. Therefore, in the present benchmark measurements, we separately evaluated the performances of these systems.

5.2. Control with VR-Authoring Tool by Unity. Unity is a VR-authoring tool developed by Unity Technologies. Unity is widely used as a cross-platform game engine for 3D video games and simulations. In the present work, we used Unity version 2017.3. Note that Unity Technologies offers free educational licenses for academic institutions. Supporting Information B describes the setup for Unity version 2017. Supporting Information C is a step-by-step description of the use of Unity version 2017 for VR of CMS simulation data. Supporting Information D describes the setup of a function for cutting VR objects using Unity, whereas Supporting Information E describes the setup for measuring frame rates (in fps) in Unity. Note that a brief explanation of how the Unity asset Cross Section Shader works is given in Section 3.3.

5.3. Measurement Conditions. For practical use, we want to understand the behavior of the frame rates for various object sizes during walk-through VR observations using cross sections.
from a handheld device. Since the motions during actual walk-through VR observations are not reproducible, it is necessary to evaluate a reproducible motion. Furthermore, it is desirable to separate the movements of the VR device and the handheld device. To estimate the frame rate systematically, we required certain devices on a standard scale.

As a device for moving the head-mounted display (HMD) periodically, we first considered mounting the HMD on a rail frame with a motor. However, the production cost of such a device is much more than the cost of HTC VIVE, so it is not realistic in terms of commodity VR. As a device that periodically moves the HMD at low cost, we relied on the principles of nature and created a pendulum of the VR device and of the handheld device as shown in Figure 9. For the handheld device, we placed the VR device on the table at $z = -40$ cm. We swung the pendulum of the handheld device along the $z$-direction at its natural frequency. For a length of 125 cm, the natural period is $\sim 2.24$ s. Here, the swing width of the pendulum was approximately from $-50$ to $+50$ cm. We manually maintained a constant swing width and direction. We recorded the number of frames drawn per second, with the first several seconds discarded and the following 50 s used. Here, we measured the frame rate in the Play mode of Unity to monitor several values by replacing the given models and measurements for simplicity (with the novice user in mind), although built binaries generally show slightly better performance than the Play mode.

**ASSOCIATED CONTENT**

* Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acsomega.8b03483.

Availability of other VR devices, setup of Unity 2017 for VR, step-by-step use of Unity 2017, setup of VR object cutting function, measurement of frame rate, supporting note (PDF)

Sample data; ball-stick systems drawn using PDB, xyz format files, and LAMMPS trajectory files, and isosurface system (ZIP)
ACKNOWLEDGMENTS

The authors thank Prof. Y. Kawazoe, Prof. G. P. Das, Dr R. Sahara, I. Takahashi, Dr T. Murashima, Dr N. Iwaoaka, and Prof. H. Takano for useful discussions. They also acknowledge support from A. Ueda, Dr S. Ueda, and Dr M. Hoshino in preparing the data. One of the authors (K.H.) was supported by the Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures. This work was partially supported by JSPS KAKENHI, Japan, Grant No. JP18H04494.

REFERENCES

(1) Cruz-Neira, C.; Sandin, D. J.; DeFanti, T. A. In Surround-Screen Projection-based Virtual Reality: The Design and Implementation of the CAVE, Proceedings of the 24th International Conference and the 9th Users Workshop, 1993; pp 135–142.

(2) Feibenti, A.; Nishimoto, A.; Thiggen, T.; Talandis, J.; Long, L.; Pirtle, J. D.; Peterka, T.; Verlo, A.; Brown, M.; Pleppys, D.; Sandin, D.; Renambot, L.; Johnson, A.; Leigh, J. In CAVE2: A Hybrid Reality Environment for Immersive Simulation and Information Analysis, Proceedings Volume 8649, The Engineering Reality of Virtual Reality 2013, 2013, 864903.

(3) Stein, J. Why Virtual Reality Is About to Change the World Time, 2015. http://time.com/3987022/ (accessed Dec 1, 2018).

(4) Bal0, A. R.; Wang, M.; Ernst, O. P. Accessible Virtual Reality of Biomolecular Structural Models using the Autodesk Molecule Viewer. Nat. Methods 2017, 14, 1122–1123.

(5) Norby, M.; Grebner, C.; Eriksson, J.; Roström, J. Molecular Virtual Reality for Drug Designers. J. Chem. Inf. Model. 2015, 55, 2475–2484.

(6) Stone, J. E.; Sherman, W. R.; Schulten, K. In Immersive Molecular Visualization with Omnidirectional Stereoscopic Ray Tracing and Remote Rendering, 2016 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2016; pp 1048–1047.

(7) Stone, J. E.; Kohlmeyer, A.; Vandvort, K. L.; Schulten, K. Immersive Molecular Visualization and Interactive Modeling with Commodity Hardware. Lect. Notes Comput. Sci. 2010, 6454, 382–393.

(8) O’Connor, M.; Deeks, H. M.; Dawn, E.; Metalta, O.; Roudaut, A.; Sutton, M.; Thomas, L. M.; Glowacki, B. R.; Sage, R.; Tew, P.; Wonnacott, M.; Bates, P.; Mulholland, A. J.; Glowacki, D. R. Sampling Molecular Conformations and Dynamics in a Multisizer Virtual Reality Framework. Sci. Adv. 2018, 4, No. eaat2731.

(9) Goddard, T. D.; Brilliant, A. A.; Skillman, T. I.; Vergenz, S.; Tyrvahnt-Drake, J.; Meng, E. C.; Ferrin, T. E. Molecular Visualization on the Holodeck. J. Mol. Biol. 2018, 430, 3982–3996.

(10) Goddard, T. D.; Huang, C. C.; Meng, E. C.; Petersen, E. F.; Cough, G. S.; Morris, J. H.; Ferrin, T. E. UCSF Chimera X: Meeting Modern Challenges in Visualization and Analysis. Protein Sci. 2018, 27, 14–25.

(11) Johnston, A. P. R.; Rae, J.; Ariotti, N.; Bailey, B.; Lilja, A.; Webb, R.; Ferguson, C.; Maher, S.; Davis, T. P.; Webb, R. I.; McGhee, J.; Parton, R. G. Journey to the Centre of the Cell: Virtual Reality Immersion into Scientific Data. Traffic 2018, 19, 105–110.

(12) Borrel, A.; Fourches, D. RealityConvert: a Tool for Preparing 3D Models of Biochemical Structures for Augmented and Virtual Reality. Bioinformatics 2017, 33, 3816–3818.

(13) Zheng, M.; Waller, M. P. ChemPreview: An Augmented Reality-Based Molecular Interface. J. Mol. Graphics Modell. 2017, 73, 18–23.

(14) Lv, Z.; Tek, A.; Da Silva, F.; Emperour-mot, C.; Chavent, M.; Baaden, M. Game On, Science - How Video Game Technology May Help Biologists Tackle Visualization Challenges. PLoS One 2013, 8, No. e57990.

(15) Salvadori, A.; Fusè, M.; Mancini, G.; Rampino, S.; Barone, V. Diving into Chemical Bonding: An Immersive Analysis of the Electron Charge Rearrangement Through Virtual Reality. J. Comput. Chem. 2018, 39, 2607–2617.

(16) VIVE. https://www.vive.com/ (accessed Dec 01, 2018).

(17) Oculus Rift. https://www.oculus.com/rift/ (accessed Dec 01, 2018).

(18) Unity. https://unity3d.com/ (accessed Dec 01, 2018).

(19) Unreal Engine. https://www.unrealengine.com/ (accessed Dec 01, 2018).

(20) Choy, T.-S.; Naset, J.; Chen, J.; Hershfield, S.; Stanton, C. A Database of Fermi Surfaces in Virtual Reality Modeling Language (vml). Bull. Am. Phys. Soc. 2000, 45, L36–L42. http://www.phys.ull.earm/surface/ (accessed December 1, 2018).

(21) Garcia-Hernández, R. J.; Kranzlmüller, D. In Virtual Reality Toolkit for Material Science: NOMAD VR Tools, Lecture Notes on Computer Science, 2017; pp 309–319.

(22) The NOMAD (Novel Materials Discovery) Laboratory. Final Assessment of Visualization Tools. https://www.nomad-coe.eu/uploads/outreach/Public%20Deliverables/NOMAD_D3.3_public_07Nov2017.pdf (accessed Dec 01, 2018).

(23) Humphrey, W.; Dalke, A.; Schulten, K. VMD - Visual Molecular Dynamics. J. Mol. Graphics 1996, 14, 33–38.

(24) Schrödinger, Inc.. The PyMOL Molecular Graphics System, Version 2.2. https://pymol.org/ (accessed Jan 15, 2019).

(25) SMART Laboratory Caffeine. http://smart.sns.it/caffeine/ (accessed Jan 15, 2019).

(26) The Nomad (Novel Materials Discovery) Laboratory. Final Assessment of Visualization Tools. https://www.nomad-coe.eu/uploads/outreach/Public%20Deliverables/NOMAD_D3.3_public_07Nov2017.pdf (accessed Dec 01, 2018).

(27) Marschner, S.; Shirley, F. Fundamentals of Computer Graphics, 4th ed.; CRC Press: New York, 2016.

(28) LaValle, S. M. Virtual Reality; Cambridge University Press: New York, 2016.

(29) Ohno, N.; Kageyama, A. Introduction to Virtual Reality Visualization by the CAVE System. In Advanced Methods for Space Simulations; Usui, H., Omura, Y., Eds.; TERRAPUB: Tokyo, 2007; pp 167–207.

(30) Optimisation for VR in Unity. https://unity3d.com/learn/tutorials/topics/unity3d/optimisation-vr-unity (accessed Dec 01, 2018).

(31) Bex, P. J.; Edgar, G. K.; Smith, A. T. Multiple Images Appear When Motion Detection Fails. J. Exp. Psychol. Hum. Percept. Perform. 1995, 21, 231–238.

(32) Kolasinski, E. M. Simulator Sickness in Virtual Environments; Army Research Institute for the Behavioral and Social Sciences: Alexandria, VA, 1995.

(33) Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. J. Comput. Phys. 1995, 117, 1–19.

(34) Hagita, K.; Kato, K.; Tominga, T.; Jinnai, H. Direct Confirmation of Structure-Scattering Pattern Relationship of Phase Separated ABA Block Copolymers. Soft Matter 2019, 926–936.

(35) Krenner, K.; Grest, G. S. Dynamics of Entangled Linear Polymer Melts: A Molecular-Dynamics Simulation. J. Chem. Phys. 1990, 92, 5057–5087.

(36) Hagita, K.; Higuchi, T.; Jinnai, H. Super-Resolution for Asymmetric Resolution of FIB-SEM 3D Imaging using AI with Deep Learning. Sci. Rep. 2018, 8, No. 8577.

Author Contributions

Katsumi Hagita: 0000-0002-6708-7468

ORCID

K. Hagita

Author Information

Corresponding Author

*E-mail: hagita@nda.ac.jp.

Acknowledgements

The authors declare no competing financial interest.

Author Contributions

K.H. conceived this work. K.H. and S.M. developed the framework of the VR. All authors performed visualizations, analyses, and discussed the obtained results. All authors contributed to the manuscript. All authors have given approval to the final version of the manuscript.
(37) Lauritzen, A. In Deferred Rendering for Current and Future Rendering Pipelines, SIGGRAPH 2010 Course: Beyond Programmable Shading, 2010; pp 4–11.
(38) Kresse, G.; Hafner, J. Ab Initio Molecular Dynamics for Liquid Metals. Phys. Rev. B 1993, 47, 558–561.
(39) Ono, S.; Noguchi, Y.; Sahara, R.; Kawazoe, Y.; Ohno, K. TOMBO: All-Electron Mixed-Basis Approach to Condensed Matter Physics. Comput. Phys. Commun. 2015, 189, 20–30.
(40) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Petersson, G. A.; Nakatsuji, H.; Li, X.; Caricato, M.; Marenich, A. V.; Bloino, J.; Janesko, B. G.; Gomperts, R.; Mennucci, B.; Hratchian, H. P.; Ortiz, J. V.; Izmaylov, A. F.; Sonnenberg, J. L.; Williams-Young, D.; Ding, F.; Lipparini, F.; Egidi, F.; Goings, J.; Peng, B.; Petrone, A.; Henderson, T.; Ranasinghe, D.; Zakrzewski, V. G.; Gao, J.; Rega, N.; Zheng, G.; Liang, W.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Throssell, K.; Montgomery, J. A., Jr.; Peralta, J. E.; Ogliaro, F.; Bearpark, M. J.; Heyd, J. J.; Brothers, E. N.; Kudin, K. N.; Staroverov, V. N.; Keith, T. A.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Rendell, A. P.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Millam, J. M.; Klene, M.; Adamo, C.; Cammi, R.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Farkas, O.; Foresman, J. B.; Fox, D. J. Gaussian 16, revision A.03; Gaussian, Inc.: Wallingford, CT, 2016.
(41) Kokalj, A. XCrySDen—A New Program for Displaying Crystalline Structures and Electron Densities. J. Mol. Graphics Modell. 1999, 3–4, 176–179.
(42) Wang, J.-T.; Chen, C.; Wang, E.; Kawazoe, Y. Magic Monatomic Linear Chains for Mn Nanowire Self-Assembly on Si(001). Phys. Rev. Lett. 2010, 105, No. 116102.
(43) Zhang, S.; Zhou, J.; Wang, Q.; Chen, X.; Kawazoe, Y.; Jena, P. Penta-Graphene: A New Carbon Allotrope. Proc. Natl. Acad. Sci. U.S.A. 2015, 112, 2372–2377.