Using Unity for Scientific Visualization as a Course-based Undergraduate Research Experience

Idunnuoluwa Adeniji
Kean University
Argonne National Laboratory
iadeniji@kean.edu

Lark Bancairen
Kean University
bancaila@kean.edu

Cymantha Blackmon
Kean University
blackmoc@kean.edu

Michael Casarona
Kean University
casaromi@kean.edu

Melissa Menzel
Kean University
menzelm@kean.edu

Matthew G. Niepielko
Kean University
mniepiel@kean.edu

Leonard Bielory
Kean University
Rutgers University – Center of Environmental Prediction
Hackensack Meridian School of Medicine
lbielory@kean.edu

Nan Perigo
Kean University
nperigo@kean.edu

Joseph Insley
Argonne National Laboratory
Northern Illinois University
Insley@anl.gov

David Joiner
Kean University
djoiner@kean.edu

ABSTRACT

We have developed a series of course-based undergraduate research experiences for students integrated into course curriculum centered around the use of 3D visualization and virtual reality for science visualization. One project involves the creation and use of a volumetric renderer for hyperstack images, paired with a biology project in confocal microscopy. Students have worked to develop and test VR enabled tools for confocal microscopy visualization across headset based and CAVE based VR platforms. Two applications of the tool are presented: a rendering of *Drosophila* primordial germ cells coupled with automated detection and counting, and a database in development of 3D renderings of pollen grains. Another project involves the development and testing of point cloud renderers. Student work has focused on performance testing and enhancement across a range of 2D and 3D hardware, including native Quest apps. Through the process of developing these tools, students are introduced to scientific visualization concepts, while gaining practical experience with programming, software engineering, graphics, shader programming, and cross-platform design.

KEYWORDS

Unity, Visualization, course-based undergraduate research experiences

1 INTRODUCTION

Significant study has been made in the impact on the student experience of course-based undergraduate research experiences (CUREs), in which activities focused on following the research process and performing inquiry is used in addition to or in place of traditional laboratory activity [4]. Positive benefits of CUREs have been seen in professional identity, research skills, project ownership, and higher retention [2, 3, 8, 11].

Our context for implementation is a computational science and engineering program at Kean University, a regional university in northern New Jersey. The program is part of a 5-year combined BS/MS honors program in the School of Integrative Science and Technology.

The projects presented in this paper involve development or enhancement of software in Unity Game Engine [10]. The four CUREs that the computational science students have been working on presented in this paper center around volumetric rendering of hyperstack data, and performance testing and implementation of point cloud renderers in Unity using VR hardware.

2 METHODS

2.1 Hyperstack Image Rendering

VR is increasingly being used for visualization of hyperstack data in scientific and medical imaging [9]. It has shown potential for use specifically with confocal microscopy along with other data [16].

We have worked with students implementing a technique for rendering hyperstacks by using transparency and shading within Unity's rendering pipeline, as opposed to traditional raytracing, for a process that is performant, easy to implement, and is cross platform compatible across laptop and VR hardware.

Our focus with student research projects has been on two sets of data, the first involving the formation of primordial germ cells in fruit fly embryos. Students were given the task of developing tools to count the number of cells in the embryo, with cells overlapping in 3D and requiring a 3D object detection approach. The second data set is a developing database of 3D scans of pollen grains. While multiple databases exist of 2D images of pollen grains for the purpose of both human and AI training for pollen detection, there are far fewer databases of 3D pollen scans, a notable exception being the recently introduced 3D Pollen Project [19]. As part of a project to provide 3D images of common US pollen species, our students use Unity to render

https://doi.org/10.22369/issn.2153-4136/15/1/7

hyperstack images of pollen grains to increase availability of 3D pollen data.

2.2 Point Cloud Rendering

An increasing amount of scientific data is rendered as point clouds, both due to the use of higher quality 3d scanners as well as traditional glyph renderings of large point-based computations. Creating point clouds in Unity has traditionally used geometry shaders [15]. In cases where geometry shaders are not supported, some software has instead used point shaders with a pointsize parameter [17]. Other work in the area has focused on techniques for efficient point decimation for faster loading and display, with typical renderings done at a level of one million points [5].

We set out to have students compare different techniques for point cloud rendering, as well as perform cross platform testing. Students were assigned the task initially of working with the Keijero point cloud model, and testing it with data sets of varying size, as well as across platforms, testing on Windows, Android, and OS X using where possible Direct X, OpenGL, Vulkan, and Metal graphics APIs, using both PC, Oculus Rift, Oculus Quest, and Oculus Quest 2 hardware. Over the course of this project, additional geometry-based shaders were developed and tested as well.

3 CURES PROJECTS

3.1 Hyperstack Image Rendering Projects

3.1.1 Primordial Germ Cells rendering network viewer

This project has focused on creating multi-user spaces in VR to view and share 3D models of primordial germ cell data from a confocal microscope. Primordial Germ Cells (PGCs) are embryonic precursors that pass on both genetic and epigenetic information to succeeding generations [5]. While broadly applicable to any hyperstack image data, the particular application is driven by a project developing automated analysis tools for detecting cells, and the need to validate results in a 3D view. We have pursued a multi-user environment to allow remote viewing of data by multiple simultaneous viewers, based on Photon 2 Unity Networking. A web-based front end lets users upload these images and other pertinent data, and configuration files for a pre-built unity scene are automatically created. Additional tools are also in development to allow users more fine-grained control over the generated VR models, including the ability to save changes. By advancing these elements, the project aims to offer a comprehensive, user-friendly platform for 3D hyperstack image model visualization and collaboration.

3.1.2 3D Pollen Database

Seasonal allergic rhinitis (SAR) is a common inflammatory condition caused by pollen grains released by trees, grasses, weeds, or molds [6]. Many people are affected by the cold-like symptoms caused by these various pollen species [12]. Therefore, streamlining the identification and distribution of real-time pollen conditions is important because it can provide allergy sufferers with useful information to help reduce pollen exposure. We apply confocal microscopy to capture diversities in pollen grain structures, which can be further used in 3D analysis of pollen structures. Whereas other types of microscopes can only allow the external characteristics of samples to be seen, confocal microscopy allows a sample to be imaged in slices along its z-axis which are then used to create 3D and cross-sectional images. This allows the images to not only display the external morphologies and characteristics that are unique to each pollen species, but also the internal morphologies and characteristics.

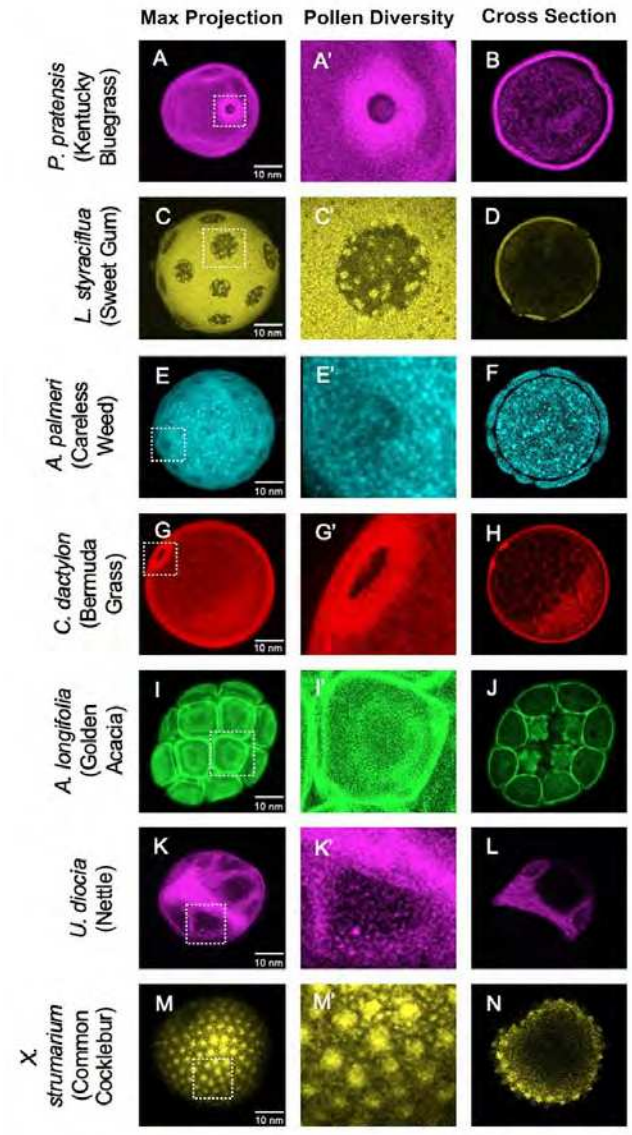

Figure 1. Confocal microscopy captures diversity in pollen structures. A-M) Max project confocal images (first column), captures differences in species' structures. A′-M′) Enlarged sections (second column) of the broken white boxes in the first column highlight key features for each pollen species. A-M) Image cross sections (third column), reveals additional differences between pollen species' structures.

The initial data used is shown in Figure 1. Each pollen species shown can be identified by their distinctive characteristics. Specifically, Kentucky Bluegrass (A) has one aperture, Sweet Gum (C) has many holes with spikes, Careless weed (E) has many circular indentations, Bermuda Grass (G) has one aperture and a disc shape, Golden Acacia (I) is split into many cube-shaped sections, Nettle (K) has an irregular shape, and Common Cocklebur (M) has small spikes on the exterior.

The goal of this project is to collect, database, and disseminate multiple 3D renderings of pollen grains for further analysis.

3.1.3 Hyperstack rendering clipping shader implementation.

A portion of Unity game and object rendering relies heavily on the computer graphics shader attached to objects to apply per-pixel calculations on GPU hardware and to allow for modification of the look and feel of objects based on shading and lighting effects. Shaders are used to create a wide variety of visual effects for both static and dynamic user interfaces. Built-in Unity shaders are typically focused on the needs of gaming, and visualization can often be simplified or made more computationally efficient by directly writing shader code that allows per-pixel (fragment) calculation of viewed effects based on data at known points (vertices).

In the case of our hyperstack images, a custom vertex-fragment shader is used to add transparency to pixel values below a threshold and modify the transparency of viewed pixels based on their intensity value.

Clipping is a standard technique used in 3D visualization software. It can be implemented in Unity at the shader level provided the origin of the clip plane in world space, the normal of the clip plane relative to the clip plane origin, and the position of the pixel rendered in world space.

This student project was to implement and test the efficiency of clipping in the shader with a combination of a dot product and the CG clip function, or in a more advanced approach a step operation in place of clip to create a variable that can be used to create a more blended view (e.g., rendering the positive side of the clip plane with a different transparency value). This has been added to our custom shader in the hyperstack renderings in the two projects mentioned above.

4 Point Cloud Rendering Projects

4.1.1 HACC simulation visualization

Traditional large scientific data exploration predominantly relies on 2D and 3D visualization tools. However, a transformative shift is occurring as lower cost virtual reality (VR) hardware emerges, offering immersive experiences. This study showcases the development of a data pipeline from conventional visualization tools like Paraview to Unity Game Engine and the Oculus Quest 2 headset.

The primary goal is to explore this transition, with a focus on enabling a more comprehensive understanding of complex datasets.

5 STUDENT PROJECT RESULTS

5.1 Hyperstack Image Rendering

Student work on hyperstack image rendering focused on techniques for image preparation for rendering. Our pipeline was to open the hyperstacks in Image J and save each z-slice as an individual jpeg. Images were imported into the Unity editor as RGBA 32 bit sprites with alpha as transparency and read/write selected. The images were then projected onto quads, using an unlit 2 sided transparent shader. The shader was designed to clip any pixels below a set threshold and apply an alpha level proportional to the total brightness of the pixel. The quads were spaced in the scene evenly according to their height in *z*.

5.1.1 Primordial Germ Cells rendering network viewer

A network-based visualization tool has been created, with a web-based management interface so that users can easily download the visualization tool, as well as create and manage cell models. This will eventually allow for additional custom analytics tools to be run simultaneously with the model creation so that prevalent cell data can be highlighted through automated analysis and then confirmed visually. The scene created for the networked viewer is displayed in Figure 2.

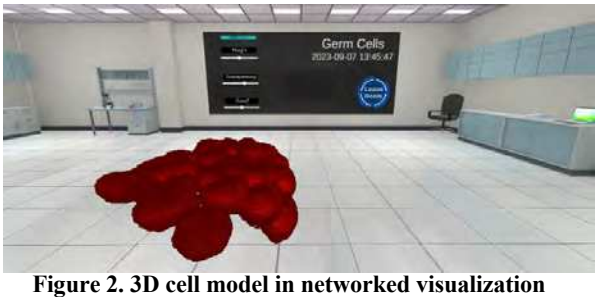

Figure 2. 3D cell model in networked visualization application. Cell images are loaded into the lab environment and compiled into a 3D model. The user interface (UI) located on the board displays all relevant cell information as well as the slider interface that is able to transform the 3D cell model. The lab scene provides a comfortable environment for working with the model as well as provides a professional feeling to the application.

5.1.2 3D Pollen Database

An early rendering is shown in Figure 3, with data from a rendering of a Ragweed pollen grain. Input data for rendering was 1576x1576x60, with 100+ FPS performance on a TensorBook w/ GeForce 3080. Other performance measures were 30+ FPS for an Oculus Quest Pro connected with link cable, and 20+ FPS for a native Quest Pro app with data downgraded to 1024x1024x60. Of note was that as we moved from a high-end workstation to a native Quest app, images larger than 1024x1024 noticeably degraded performance.

Further images were limited to 1024x1024, and communication with the biology team resulted in additional student projects focused on production of pollen data. Figure 4 shows a scan of a Golden Acacia pollen grain. Input data for rendering was 1024x1024x90, with 300+ FPS performance on a TensorBook w/ GeForce 3080. Other performance measures were 60+ FPS for an Oculus Quest Pro connected with link cable, and 40+ FPS for a native Quest Pro app.

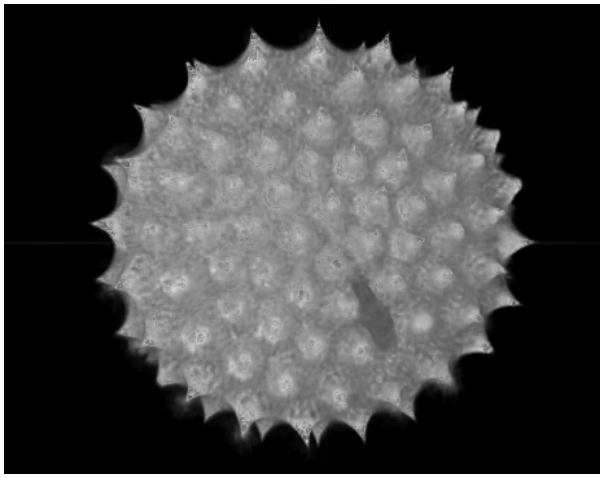

Figure 3. Hyperstack rendering of confocal scan of a Ragweed pollen grain.

In summary, the study found that confocal microscopy can be used to produce detailed images of pollen grain species and provide 3D images that can be reconstructed in Unity.

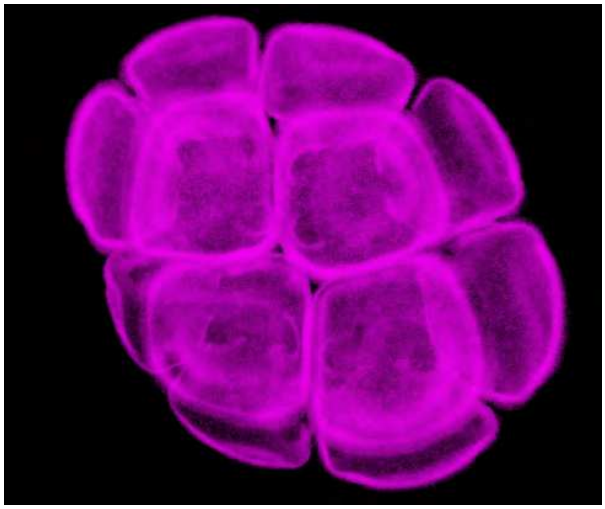

Figure 4. Hyperstack rendering of confocal scan of a Golden Acacia pollen grain.

5.1.3 Hyperstack rendering clipping shader implementation

A clip plane effect was added to the shaders used in the hyperstack viewer, and results can be seen in Figure 5. Clipping was implemented by calculating the world space position of each vertex $\vec{p}_{ws}$ in the shader, and setting shader properties for the base (in world space coordinates) $\vec{b}_{ws}$ and normal $\hat{n}$ (relative to the base) of the clipping plane, at which point the clip function could be applied to $(\vec{p}_{ws} - \vec{b}_{ws}) \cdot \hat{n}$. Addition of clipping plane had no impact on performance with the GeForce 3080 test, Link Cable, or native Quest app.

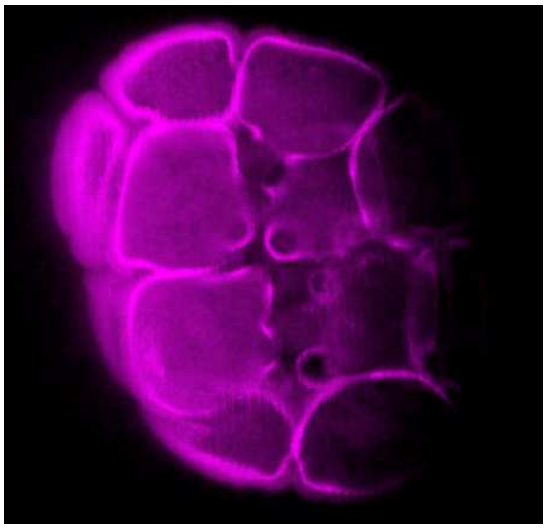

Figure 5. Hyperstack rendering of confocal scan of a Golden Acacia pollen grain with clipping plane applied.

5.2 Point Cloud Rendering

One challenge with developing point cloud rendering approaches that work across a wide range of platforms, including Windows, Mac, and VR headsets, is the difference in graphics API availability for those platforms, with DirectX being windows specific, Vulkan working on Windows or Android, OpenGL supported by Windows, Android, and Intel-based Macs, and Metal supported by Macs. Metal is the only graphics API supported for the M series of Macs [1].

Figure 6 shows a test of the point cloud rendering process, applied to a 10,000,000-point Sierpinski Trapezoid calculated using an iterated function system. The cloud rendered on a TensorBook with a GeForce 3080 at 100+ FPS. Similar performance on other hardware include Quest Pro connected with link cable and 10 million points at 30+ FPS. A natively compiled Quest Pro app with 1,000,000 points ran at 10+ FPS.

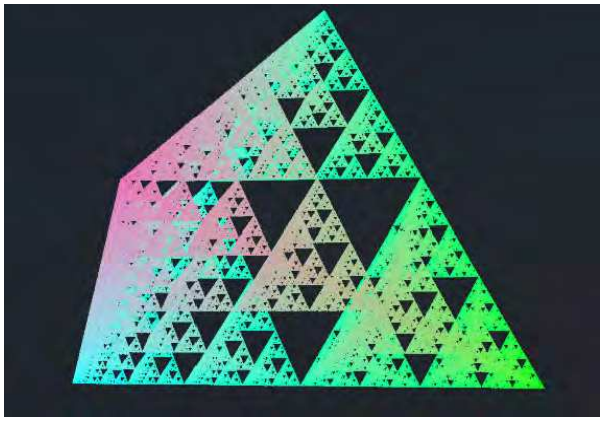

Figure 6. Point cloud of a 10-million-point Sierpinski Tetrahedron

Geometry shaders have proven to have wide cross platform compatibility and good performance on all hardware tested except for M series chip-based Macs. Future student projects will investigate workarounds for rendering on newer Macs restricted to the Metal graphics API.

5.2.1 HACC simulation visualization

This project applied virtual reality to the identification of patterns and clusters within point cloud data, in particular a set of data generated by a Hardware/Hybrid Accelerated Cosmology Code (HACC) simulation (Figure 7, Figure 8).

Figure 7. Scene view of HACC data in the VR environment

In addition to enhancing scientists' comprehension of their datasets, this study emphasizes the integration of VR into the broader field of large data exploration, which includes features like data interaction, manipulation, and in-depth analysis. This study employs a custom import script designed for transferring particle data generated by a HACC simulation into Unity. The files we work with contain a sample of approximately 4 million particles (out of up to 2 trillion simulated), from which we load a selected subset. The point cloud data is presently rendered as a point topology mesh using a custom geometry shader.

Through the immersion of users in virtual environments, this study significantly amplifies the identification of patterns and clusters when compared to conventional methods. In addition, we achieved an average PC frame rate of approximately 1012.3 frames per second (FPS) when visualizing 10,000 particles on an Alienware Aurora R15 with a GeForce 3090. When considering the entire dataset of approximately 4 million particles, we attained an average PC frame rate of approximately 235.9 FPS. Beyond static visuals, a custom time lapse technique animates data, providing insights into pattern evolution.

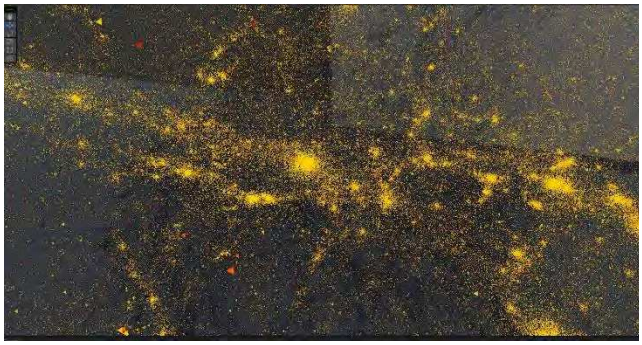

Figure 8. Headset view of HACC data in the VR environment

6 DISCUSSION

VR driven scientific visualization can provide a source of rich, authentic research-based activities for students learning skills in computational science. As a real-world example for students, scientific visualization using Unity in VR has many features we have found beneficial to students. Unity's C# language is similar to Java, which is the primary language used for our students' computer science coursework.

In training students to use Unity for visualization projects, we have developed an offering of our freshman research course sequence focused on Unity programming. Materials used in this are a combination of public materials available at https://learn.unity.com [18], in particular the ubiquitous "Roll-A-Ball" lesson, as well as hosted materials written for the course, available at https://joinerda.github.io/ [14], in particular the "Hello Unity," "Using GetComponent," and "Lorenz Butterfly" tutorials, which allow for covering of Unity in a computational science context. Typically, students then brainstorm ideas on projects. Past course offerings have included projects on using VR to hand count primordial germ cells, as well as non-visualization projects such as creating a lab safety simulation. When adding in discussions of VR, class projects have focused on the concepts of grabbing, rotating, scaling, and activating (pressing) objects – activities that are ubiquitous across modern VR toolkits.

Interested students continue in sophomore, junior, and senior years with independent research, working specifically on visualization projects.

7 REPRODUCIBILITY

For adding virtual reality hardware support to our Unity models, we are using the XR Interaction Toolkit (XRI) , which provides a player object that can be used with a variety of VR systems. Students are instructed to use XRI version 2.3.2 or later, and to include the samples when installing, as the samples includes a working player controller and locomotion system that they can quickly copy and paste from the sample scene into their scene rather than configuring from scratch. Students need the ability to plug their headset into their laptop in order to install the apps they create, so at least one USB port needs to be available, and an appropriate cable. Students use the personal edition of Unity Game Engine, though depending on the use case there are also options to request a Unity license for education through Unity's grant program. Students use the community edition of Microsoft Visual Studio C# compiler.

For installing to Quest, Unity needs to be configured for Android build. Recent versions of Unity do not require extensive additional installation of Android toolkits, and installation can be managed solely through the Unity Hub. ASTC Texture compression is selected in the build settings.

Dependencies for XR Interaction Toolkit include the AR Foundation, XR Plugin Management, Oculus XR Plugin, and OpenXR Plugin (if using Oculus Link through a cable) packages. Version information and package requirements are as follows: Unity Version for tests: 2021.3.8f1, AR Foundation 4.2.8, Oculus XR Plugin 3.0.2, OpenXR Plugin 1.4.2, XR Interaction Toolkit 2.3.2, Android Graphics API OpenGLES3.2, Linear color space, OpenXR Plugin used for PC, Oculus Plugin used for Android, Multi-Pass Rendering for both plugins.

Drosophila primordial germ cells were labeled and detected using immunofluorescence and was carried using polyclonal anti-Vasa (Boster Bio, cat #DZ41154) and secondary Alexa Fluor 568 (anti-rabbit, ThermoFisher, cat #A10042) as previously described in detail [7]. Confocal microscopy was carried out using the Leica STELLARIS 5 white light laser system and Leica LIGHTNING following previously established protocols [7]. For pollen images, dry pollen samples were mounted in ProLong Glass (Life Technologies) and imaged using pollen's autofluorescent properties that are produced when exposed to UV.

ACKNOWLEDGMENTS

We thank Kean University, The Center for Aerobiological Research, and Kean's Center for Biological Imaging for providing pollen slides and imaging facilities. Work on this project was supported by NSF IUSE-HSI 2247157 and the primordial germ cell project was supported by NIH R15HD102960. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

REFERENCES

[1] Apple Developer Forum. 2022. Are OpenGL and OpenCL supported on Apple silicon? Retrieved from https://developer.apple.com/forums/thread/694866.

[2] Aysha Alneyadi, Iltaf Shah, and Syed Salman Ashraf. 2019. An innovative bioanalytical research project course to train undergraduate students on liquid chromatography–

mass spectrometry, *Biochem. Mol. Biol. Educ.* 47, 3, 228–238.

[3] Sara E. Brownell, Mary Pat Wenderoth, Roddy Theobald, Nnadozie Okoroafor, Mikhail Koval, Scott Freeman, Cristina L. Walcher-Chevillet, and Alison J. Crowe. 2014. How students think about experimental design: novel conceptions revealed by in-class activities. *BioScience* 64, 2, 125–137.

[4] Alaina J. Buchanan and Ginger R. Fisher. 2022. Current status and implementation of science practices in course-based undergraduate research experiences (CUREs): A ystematic literature review," *CBE—Life Sci. Educ.* 21, 4, ar83.

[5] Ryan M. Cinalli, Prashanth Rangan, and Ruth Lehmann. 2008. Germ cells are forever. *Cell* 132, 4, 559–562.

[6] Arianna Dondi, Salvatore Tripodi, Valentina Panetta, Riccardo Asero, Andrea Di Rienzo Businco, Annamaria Bianchi, Antonio Carlucci, Giampaolo Ricci, Federica Bellini, Nunzia Maiello, et al. 2013. Pollen-induced allergic rhinitis in 1360 Italian children: Comorbidities and determinants of severity," *Pediatr. Allergy Immunol.* 24, 8, 742–751.

[7] Dominique A Doyle, Florencia N Burian, Benjamin Aharoni, Annabelle J Klinder, Melissa M Menzel, Gerard Carlo C Nifras, Ahad L Shabazz-Henry, Bianca Ulrich Palma, Gisselle A Hidalgo, Christopher J Sottolano, et al. 2023. Germ granule evolution provides mechanistic insight into drosophila germline development. *Mol. Biol. Evol.* 40, 8, msad174.

[8] Jennifer C. Drew and Eric W. Triplett. 2008. Whole genome sequencing in the undergraduate classroom: outcomes and lessons from a pilot course. *J. Microbiol. Biol. Educ.* 9, 1, 3–11.

[9] Corentin Guérinot, Valentin Marcon, Charlotte Godard, Thomas Blanc, Hippolyte Verdier, Guillaume Planchon, Francesca Raimondi, Nathalie Boddaert, Mariana Alonso, Kurt Sailor, et al.. 2022. New approach to accelerated image annotation by leveraging virtual reality and cloud computing. *Front. Bioinforma.* 1, 777101.

[10] John K. Haas. 2014. A history of the unity game engine. Retrieved from https://digitalcommons.wpi.edu/iqp-all/3207

[11] D. I. Hanauer, J. Frederick, B. Fotinakes, and S. A. Strobel. 2012. Linguistic analysis of project ownership for undergraduate research experiences. *CBE—Life Sci. Educ.* 11, 4, 378–385.

[12] Abdullah Aburiziza, Mohammed A Almatrafi, Aishah Saud Alonazi, Mawaddah Hani Zatari, Samah Ali Alqouzi, Rasha Abdulaziz Mandili, Wedad Taher Hawsawi, and Rehab Hejji Aljohani. 2008. The prevalence of nasal symptoms attributed to allergies in the United States: Findings from the burden of rhinitis in an America survey. *Allergy Asthma Proc.* 29, 6, 600–8.

[13] Elias Neuman-Donihue, Michael Jarvis, and Yuhao Zhu. 2023. FastPoints: A state-of-the-art point cloud renderer for Unity. Retrieved from https://arxiv.org/abs/2302.05002

[14] David Joiner. n.d. David Joiner – Computational Science Educator. Retrieved from https://joinerda.github.io/

[15] Santana Núñez, José Miguel, Trujillo Pino, Agustín Rafael, Ortega Trujillo, and Sebastián Eleazar. 2019. Visualization of large point cloud in unity. *Eurographics Tech. Rep. Ser*. Retrieved from http://hdl.handle.net/10553/70567

[16] C. Stefani, A. Lacy-Hulbert, and T. Skillman. 2018. ConfocalVR: immersive visualization for confocal microscopy. *J. Mol. Biol.* 430, 21, 4028–4035.

[17] Keijiro Takahashi. 2017. Pcx. Retrieved from https://github.com/keijiro/Pcx.

[18] Unity. 2023. Learn game development w/ Unity; Courses & tutorials in game design, VR, AR, & Real-time 3D. Retrieved from https://learn.unity.com/.

[19] Oliver J. Wilson. 2023. The 3D Pollen Project: An open repository of three-dimensional data for outreach, education and research. *Rev. Palaeobot. Palynol.* 312, 104860.