Journal of Molecular Graphics and Modelling 65 (2016) 94–99

Contents lists available at ScienceDirect

Journal of Molecular Graphics and Modelling

journal homepage: www.elsevier.com/locate/JMGM

iBET: Immersive visualization of biological electron-transfer dynamics

C. Masato Nakano[a], Erick Moen[b], Hye Suk Byun[c], Heng Ma[d], Bradley Newman[e], Alexander McDowell[e], Tao Wei[d,*], Mohamed Y. El-Naggar[c,f,g,**]

[a] Flintridge Preparatory School, La Canada, CA 91011, USA
[b] Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-0241, USA
[c] Department of Physics & Astronomy, University of Southern California, Los Angeles, CA 90089-0484, USA
[d] Department of Chemical Engineering, Lamar University, Beaumont, TX 77710, USA
[e] School of Cinematic Arts, University of Southern California, Los Angeles, CA 90089-2211, USA
[f] Molecular and Computational Biology Section, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089-0371, USA
[g] Department of Chemistry, University of Southern California, Los Angeles, CA 90089-1062, USA

ARTICLE INFO

Article history:
Received 9 November 2015
Received in revised form 13 January 2016
Accepted 21 February 2016
Available online 24 February 2016

PACS:
02.70.-c
05.10.Ln
07.05.Rm

Keywords:
Visualization
Virtual reality
Biological electron transfer

ABSTRACT

Recently, we presented a computational framework named VizBET to simulate and visualize biological electron-transfer (ET) dynamics. The visualization process was encapsulated as a plugin to the Visual Molecular Dynamics (VMD) software. However, the user's ability to understand complex, multidimensional ET pathways was severely limited when visualized in 2D on traditional computer monitors. To provide a more accurate representation with enhanced depth perception, we here present an extension of VizBET named iBET to render the VMD model of ET dynamics in a commodity virtual reality (VR) platform. The paper describes detailed procedures to export VMD models into the Unity game engine and render it in an Oculus Rift head mounted display. With the increasing availability of low-cost VR systems like the Rift and rich programmability of game engines, the iBET framework provides a powerful means to explore and understand not only biological ET processes but also a unique experiential tool for broad scientific communities.

1. Introduction

Reduction and oxidation (redox) reactions govern a variety of biological energy-conversion processes, including respiration. Electron transfer (ET) within and across biological molecules is the key process that essentially dictates these redox reactions [1]. Visualizing ET [2] could provide key insight into understanding these fundamental processes and possibly controlling them for a wide range of applications including renewable energy and wastewater treatment [3]. To provide better insight into the dynamics of biological ET processes, we have recently developed a computational framework named VizBET [4]. For simulating and visualizing ET dynamics in a multi-cytochrome complex, for instance, VizBET consists of an entire workflow of homology modeling, protein docking, molecular dynamics (MD) simulation, binding free energy computations, kinetic Monte Carlo (KMC) simulation of ET dynamics [5], and visualization of KMC simulation as a plugin to the widely used Visual Molecular Dynamics (VMD) software [6].

While VizBET was used successfully to reveal nonequilibrium phase transitions in biological ET [4], for more complex ET pathways, the user's ability to understand these processes was rather limited due to its 2-dimensional (2D) rendering on traditional computer monitors. Virtual reality (VR) technologies immerse the user within three-dimensional (3D) models and allow them to navigate through the environment [7]. This type of immersive experience provides the user with the necessary perceptive depth to enhance their understanding of the ET processes being rendered. The head mounted display (HMD) is one of the earliest VR technologies, starting with an early effort by Ivan Sutherland in 1960s [8]. Currently, there is a resurgence of interest in HMD technologies with the advent of low-cost commodity HMDs such as the Oculus Rift [9]. Thus far, however, applications for devices like the Rift have largely been limited to computer games. Reports on biomolecular visualization using the Rift are scarce [10], and we have not found any description of general procedures for scientists to export their

* Corresponding author.
** Corresponding author.
E-mail addresses: twei@lamar.edu (T. Wei), mnaggar@usc.edu (M.Y. El-Naggar).

http://dx.doi.org/10.1016/j.jmgm.2016.02.009
1093-3263/© 2016 Elsevier Inc. All rights reserved.

visual models from standard biomolecular visualization software such as VMD to the Oculus platform.

In this paper, we present an extension of VizBET named iBET (immersion in Biological Electron Transfer) to render VMD models of ET dynamics for display on the Oculus Rift HMD. The paper describes detailed procedures to export the VMD model into the Unity game engine [11] and render it for the Rift. Unity is a cross-platform game engine used to develop video games for personal computers (PCs), consoles, mobile devices, and websites. With the increasing availability of low-cost VR platforms, including smartphone-based VR [12], iBET provides a powerful and uniquely accessible means for understanding, not only biological ET processes, but also a wide variety of complex dynamical systems. The main contribution of this paper is to introduce commodity VR devices and game engines as a common desktop tool for a broad community of scientists, by providing step-by-step explanations of the procedure and an archive of resources that can be easily extended to other applications.

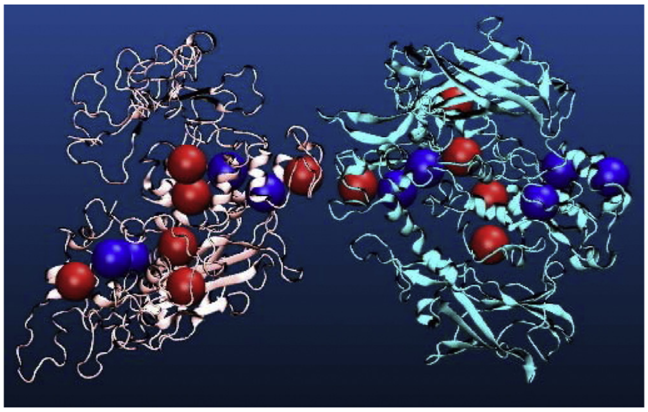


Fig. 2. A snapshot of VizBET visualization of ET dynamics in a cytochrome complex. Fe atoms are represented as spheres, and are overlaid with the NewCartoon representation of the entire protein complex, where two cytochromes named MtrF and OmcA are represented in magenta and cyan shades, respectively. Fe^{2+} and Fe^{3+} are represented by red and blue spheres, respectively. The figure shows a single snapshot from a KMC simulation.

2. Methods

2.1. System overview

To simulate ET dynamics in an assembly of proteins, the workflow of the VizBET framework [4] starts with the structures of individual protein molecules that are downloaded from the protein data bank (PDB) [13]. VizBET first pre-processes the PDB files to account for residues that were not resolved in the crystal structure with the aid of the homology-modeling web server, I-TASSER [14]. The structural outputs from this pre-processing step are then used as inputs to MD simulations that calculate the trajectories of all atoms by numerically integrating Newton's equations of motion. Individual protein configurations taken from the MD simulations are used as inputs to a protein-docking program, ZDOCK [15], to predict the structure of the protein complex. ZDOCK typically returns 2000 top configurations according to simple electrostatic and shape criteria. Our C-RANK program [4] screens these configurations into a small subset of biologically plausible configurations, according to structural and ET criteria. Subsequently, MD simulations are performed to re-solvate and relax the rigidly docked complex structure, while estimating the binding free energy with the MM/PBSA method [16]. Next, VizBET performs KMC simulations of ET dynamics [5] in the selected complex configurations. Finally, the visualization module named ETViz animates the ET dynamics in the KMC simulations [4].

At the final stage of the VizBET workflow lies the ETViz plugin [4] to the VMD software [6], which animates ET dynamics in KMC simulations using the Tcl scripting language [17] (Fig. 1). VMD is a molecular visualization program for large biomolecular systems using 3D graphics and built-in scripting. For this work, we have used VMD version 1.9.1. In the original VizBET, outputs from VMD were only rendered on a traditional computer monitor. Fig. 2 illustrates a visual output of VizBET, using ET in a complex of two cytochromes, MtrF and OmcA, as an example [4]. Such a cytochrome complex potentially plays an essential role for long-distance ET in bacterial nanowires produced by *Shewanella oneidensis* MR-1 [18,19]. Here, each decaheme cytochrome contains 10 heme groups. The iron (Fe) atom in each heme can exist in either of the two valence states, Fe^{2+} or Fe^{3+}. Conversion of the irons between Fe^{2+} and Fe^{3+} allows for the hopping of electrons between the hemes. KMC simulation simulates the hopping of electrons between Fe atoms, as well as the injection (*i.e.*, reduction) and ejection (*i.e.*, oxidation) of electrons.

The iBET framework enhances the perceptual depth over that of VizBET by exporting the visual output into VR instead of a 2D computer monitor (Fig. 1). We selected the Oculus Rift as our commodity device [9]. The system consists of a HMD and a head-movement sensor (Fig. 3). The HMD provides a stereoscopic 3D perspective to the user by showing images from different viewpoints to the left and right eyes. The sensor tracks the movement of the user's head to recalculate the images accordingly. In addition, the user navigates in the 3D model using a mouse and a keyboard. Optionally, a gamepad can be used as an input device for navigation. Porting the molecular model to VR is enabled by the Unity game engine [11]. In Unity, objects in the 3D model are manipulated and animated using the C# programming language [20] and can subsequently be rendered in VR (Alternatively, the JavaScript language [21] can be used for scripting.).

While there are many proprietary physics engines for game development, including CryEngine and Unreal Engine, Unity is one of the most widely used development platforms and one of the first (and only) to provide native support for the Oculus interface. Unity's personal license is free, widely available, and designed for easily accessible game development, making it an attractive option

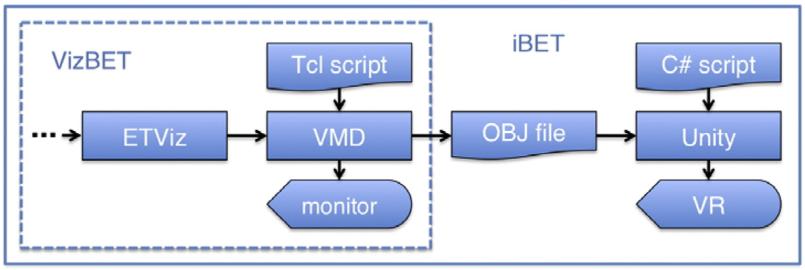


Fig. 1. Workflow of the iBET framework for immersive visualization of biological ET. The original VizBET framework is enclosed by the dashed lines, in which only the final stage (*i.e.*, the ETViz visualization module) is shown.

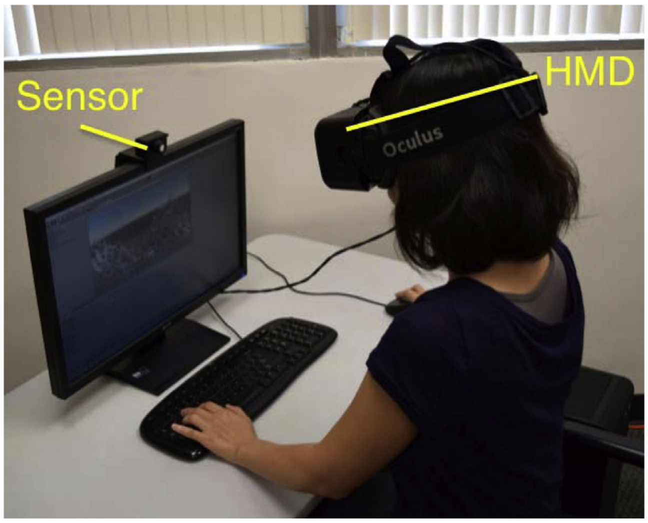


Fig. 3. iBET rendering of ET dynamics in a cytochrome complex in Oculus Rift.

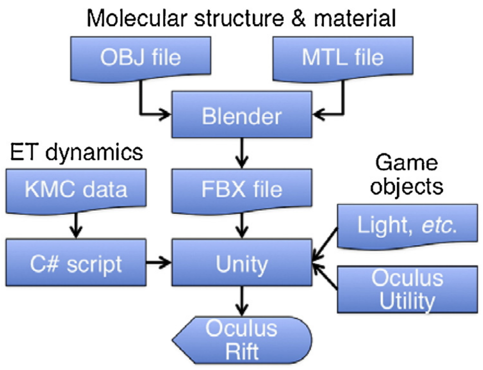


Fig. 4. Detailed iBET workflow. Biomolecular structures and material definitions, which are exported from VMD as Wavefront .obj and .mtl files, are combined into a Unity-readable .fbx file by the Blender graphics software. KMC simulation data on ET dynamics is processed by a user-defined C# script to create dynamic effects, which animate user-added game objects (*e.g.*, lights and a floor) in the Unity game engine. Prefabricated Oculus utilities for Unity allow the user to interact with the scene using the Oculus Rift VR device.

for scientists looking to incorporate it into their research endeavors. Also, unlike some other engines, Unity is not built with a specific type of game in mind – by contrast Unreal is designed for first-person shooters (FPS) – providing a "clean slate" to start from when building a simulation.

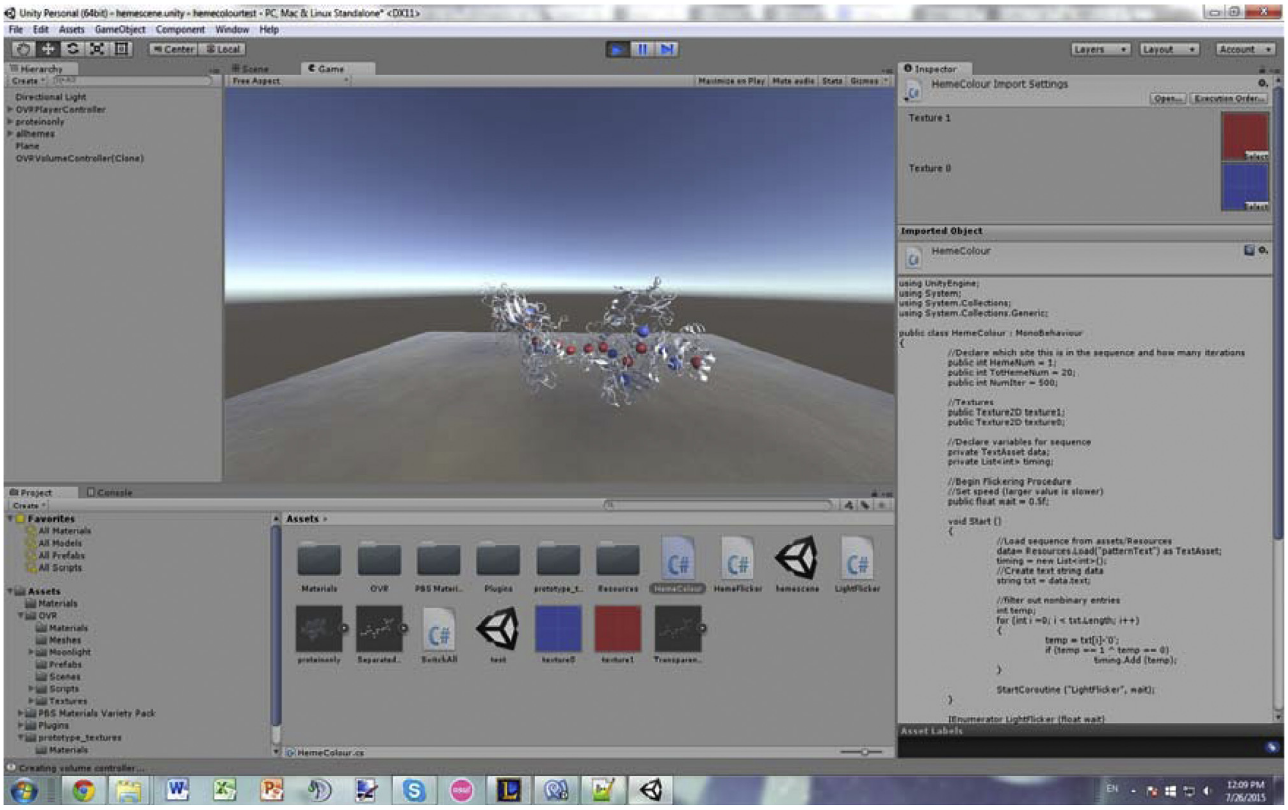
2.2. Detailed procedure

The first step of the iBET workflow (Fig. 4) is file conversion. Instead of showing the visual output on a traditional monitor, we export the 3D model from VMD into a geometry definition file in the Wavefront OBJ format (Fig. 1) [22]. This is achieved by selecting the Wavefront rendering option in VMD. VMD outputs an .obj file and an .mtl file, which respectively describe the 3D object structure and material. To convert the file into a format more appropriate for the Unity engine, we employ Blender [23] to change the format to .fbx, a file format used for interoperability in different design and rendering engines and software. Blender is an open-source 3D computer graphics software. For this project, we used version 2.75a.

First, we import the .obj file by selecting the 'Import' menu from the 'File' drop-down menu and choosing the 'Wavefront (.obj)' option. This option imports both the .obj file and the associated .mtl file, which provides the data for the material and texture of the object. Then, in order to UV map (*i.e.*, to determine a 2D representation of the surface of the 3D object) the structure, we enter Blender's 'Edit Mode' and then select the 'Mesh' submenu. From there, we access the 'UV Unwrap' submenu and then select the 'Smart UV Project' option. This will create the UV map, which corresponds to the input model. Subsequently, we export the structure and UV map as an .fbx file by going to the 'File' menu, then selecting the 'Export' subcategory and the 'FBX (.fbx)' option.

Next, we import the .fbx file to the Unity system. We have used Unity 5.1.2f1 on a PC running the Windows 7 operating system. We first create a new project, selecting the 3D option. To import the .fbx file, we access the 'Assets' menu, then select the 'Import New Asset' option. Here, asset refers to any digital material usable by Unity. For our protein complex, to simulate the twenty hemes on which the ET simulation is depicted, we create twenty spheres in Unity. Each sphere is created by selecting '3D Object' in the 'GameObject' menu and selecting the 'Sphere' option. To better reflect the size of the hemes from the VMD visualization, we adjust the scale of the spheres. Unity allows for the categorization of game objects (which represent physical structures in the scene or game environment) into other overarching game objects. In our case, we group the hemes under one object, named 'allhemes,' to simplify the animation process and to organize the project by separating the hemes from the two proteins. We have named the spheres $A1, A2, \ldots, A10$ and $B1, B2, \ldots, B10$ to fit the format of the VMD structure, where hemes designated with prefix A are assigned to the protein MtrF, and hemes designated with prefix B are assigned to the protein OmcA. Subsequently, we drag and drop the newly imported asset from the dock at the bottom of the screen into the scene window near the center.

As a game engine, it is necessary to consider how the user or player will interact with the objects and environments created in Unity. The concepts of a camera to represent the player's point of view and the locomotion necessary to orient oneself and move around the scene are both important. One simple method of facilitating player movement is the implementation of a floor on which a player can walk around. To do this, we access the 'GameObject' menu, select '3D Object,' then select 'Plane.' As a 3D game object, this plane acts as a solid object that can interact with the player, it can then be placed under the player as a floor. To orient the plane below the protein and hemes, we left-click on the object in the scene window and use the green z-axis arrow to move the plane down, or access the coordinates in the inspector panel and input a desired height. The floor allows us to take advantage of prefabricated subroutines provided by Oculus that can be downloaded from their Oculus Developers website [24]. The integration kit is known as 'Oculus Utilities for Unity' and it contains a player asset that can be placed in our scene. This asset is represented by a directed camera and it contains guidelines for control inputs. The version we use here is 0.1.0-beta. The integration kit contains a Unity package file called OculusUtilities.unitypackage. To import the package and thus the Oculus-compatible player and camera assets to Unity, we access the 'Assets' menu, select 'Import Package,' then click 'Custom Package' and select the file OculusUtilities.unitypackage. Importing this Unity package will create a subfolder called OVR in the project's Assets folder. In this OVR folder is another called Prefabs, which contains the camera and player controllers that can be dragged and dropped into the scene. The Oculus player controller already contains its own camera and controls for moving the player through the scene. For our protein simulation, we simply drag in the player controller and leave nearly all default parameters unchanged, with the exception of the position. Because we added the Oculus-compatible camera, we must delete the 'Main Camera' that is automatically in the scene. Additionally, to make the project support VR, we must select the option 'Virtual Reality Supported,' which is located in 'Edit,' then 'Project Settings,' then 'Player.' At this point, we can



```
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;

public class HemeColour : MonoBehaviour
{
        //Declare which site this is in the sequence and how many iterations
        public int HemeNum = 1;
        public int TotHemeNum = 20;
        public int NumIter = 500;

        //Textures
        public Texture2D texture1;
        public Texture2D texture0;

        //Declare variables for sequence
        private TextAsset data;
        private List<int> timing;

        //Begin Flickering Procedure
        //Set speed (larger value is slower)
        public float wait = 0.5f;

        void Start ()
        {
                //Load sequence from assets/Resources
                data= Resources.Load("patternText") as TextAsset;
                timing = new List<int>();
                //Create text string data
                string txt = data.text;

                //filter out nonbinary entries
                int temp;
                for (int i =0; i < txt.Length; i++)
                {
                        temp = txt[i]-'0';
                        if (temp == 1 ^ temp == 0)
                                timing.Add (temp);
                }

                StartCoroutine ("LightFlicker", wait);
        }

        IEnumerator LightFlicker (float wait)
```

Fig. 5. Screenshot of the Unity game engine showing the cytochrome-complex data and Oculus control modules.

build an executable by choosing 'File' then 'Build Settings.' For the executable to contain the current scene, we must select 'Add Current.' The user can run the executable and interact with the static scene.

To animate the color change in Unity, we have created a C# script. Unity allows scripts to be attached to any level of game object, including the individual spheres that we created for hemes as well as the *allhemes* object contains those spheres. To attach the script, we select the game object and drag and drop the script to the inspector panel. The script contains a *Start* function, which is called by Unity before gameplay begins to initialize the object. Our *Start* function reads a file containing the result of a KMC simulation. The data consists of a number of time steps, N_{step}, where the data at each time step is a collection of N_{heme} (= 20) numbers that represent the occupation (1) or un-occupation (0) of the N_{heme} hemes by electrons, *i.e.*, Fe^{2+} or Fe^{3+}, respectively. The N_{step} data frames, each with N_{heme} occupation numbers, are stored in a list data structure. The main function of the script is *ColorFlicker*, which is called as a coroutine. In Unity, coroutines are special functions that maintain their execution outside of the normal frame update regime. Their execution can be paused, yielding control back to Unity, and started again after *Update* has run. We take advantage of this functionality as an optimization that allows time sensitive tasks to be performed periodically without burdening the *Update* function, which could be called several times a second depending on the frame rate [25]. *ColorFlicker* progresses through a loop of the binary values provided by the KMC simulation to animate the ET dynamics. At each step, each occupied heme is represented by a sphere with red color, by assigning a predefined material texture for the color to the corresponding sphere object. Each unoccupied heme, on the other hand, is represented by a blue sphere. The time between KMC steps, and the subsequent change in heme color, is customizable and dependent on the global variable *wait*. The red and blue textures were obtained from the Unity Asset Store. The C# script is shown in the Appendix A.

It should be noted that VMD does provide stereographic capabilities, which can be shown in conventional VR devices such as CAVE. However, the stereographic rendering itself does not provide the rich programmability of graphic objects and animation supported by modern game engines such as Unity's utilities for Oculus.

3. Results

Fig. 5 is a screenshot of the ET dynamics in the heme network of a MtrF-OmcA cytochrome complex being animated from within Unity. The central image is a 2D representation of the 3D scene transmitted to the Oculus Rift HMD. The user uses a mouse to rotate the scene, and the W, S, A and D keys in a keyboard to move in the forward, backward, left, and right directions, respectively. In addition, the user can physically turn their heads to look around different directions. The head-movement tracking system senses the movement and renders appropriate images. In this application, the occupation and un-occupation of each heme are represented by red and blue colors, respectively, of a sphere that represents an Fe ion in the heme.

Supplementary archive, S1.rar, contains executable file S1.exe and directory S1_data that in turn contains associated resources. After extracting the contents of the archive, the executable and data directory must be placed in the same directory before running the .exe file. The system on which the executable was tested consists of:

- Oculus Rift VR Development Kit 2 (DK2).
- PC with Intel Core i5 quad-core central processing unit (CPU) with NVIDIA GeForce GTX 750 graphic processing unit (GPU). Oculus DK2 minimum requirements include a dedicated graphics card

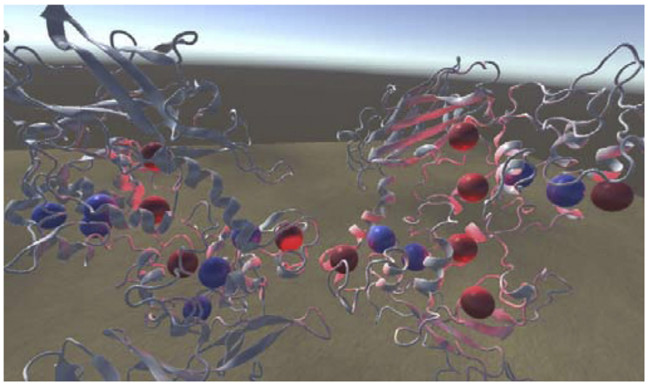


Fig. 6. Alternative visualization mode, where lighting effects are used to represent the occupation of a heme by an electron.

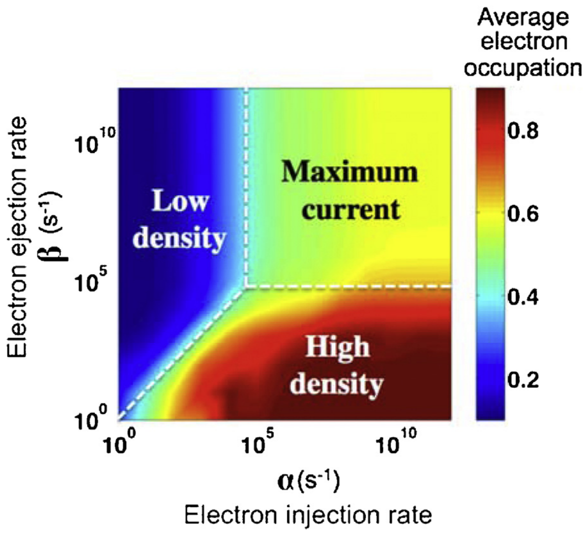


Fig. 7. Nonequilibrium phase diagram of the time-averaged electron occupation density as a function of the electron injection (α) and ejection (β) rates.

NVIDIA GTX 600 series or AMD Radeon HD 7000 series with DVI-D or HDMI graphics output.

- Windows 7 operating system (DK2 supports both Windows 7 and 8).
- Unity software version 5.1.2f1.
- Blender software version 2.75a.
- Oculus software development kit (SDK) version 0.6.01.
- Oculus Utilities for Unity version 0.1.0-beta.

To demonstrate the versatility of Unity, we have also developed a complimentary application, in which the occupation of a heme by an electron is represented by turning on a point light source at the Fe-ion position in each heme. These point lights are placed by accessing the 'GameObject' menu, the 'Light' submenu, then the 'Point Light' option. The flickering of these lights is simulated by changing the intensity of the lights between 0 and the maximum value, 8. Fig. 6 shows the lighting effect as shown in the Unity screen.

While this paper focuses on immersive scientific visualization in a commodity VR system using biological ET dynamics as an example, such visualization has already led to nontrivial findings in biophysics. For example, we found novel nonequilibrium phase transitions (Fig. 7): as the ratio of the electron injection rate to the electron ejection rate increases, we observed a transition from a low-density (LD) phase, where the hemes are mostly oxidized, to a high-density (HD) phase, where the hemes are mostly reduced. In addition, a maximum-current (MC) phase was found, when both injection and ejection rates exceed a threshold value. Our dynamic visualization played a critical role in revealing the mechanisms that underlie these transitions. By playing back and forth the ET animation like Fig. 6, we recognized essential patterns. Namely, the electron flow is limited by the small number of electrons in the LD phase, while it is limited by the congestion of electrons in the HD phase. On the contrary, the large electric current in the MC phase is facilitated by balanced electron injections and ejections [4].

The needs of immersive visualization are better illustrated by ET dynamics in a more complex 3D geometry, *i.e.*, a lattice of outer-membrane cytochrome (MtrF-OmcA) complexes in a bacterial nanowire produced by *Shewanella oneidensis* MR-1. The nanowire was found to be a cylindrical pipe made of membrane, on which thousands of outer-membrane cytochrome complexes are distributed [26]. Our preliminary study found a percolation transition, in which the nanowire becomes conductive as the time scale to observe ET dynamics exceeds a threshold value. While the transition time was found sensitive to the spatial arrangement of MtrF-OmcA complexes on the membrane, overlapping proximal and distal membranes in 2D rendering makes it nearly impossible to recognize patterns responsible for the difference. Here, it is necessary for the user to walk through both outside and inside of the cylindrical nanowire in the immersive iBET to provide some insight into the critical path that plays an essential role for the percolation transition.

In its current form, iBET does not support multiple users, but we are researching this possibility. We envision this next, multi-user version of iBET as a long-range collaboration tool. The Oculus Rift and Unity both support multiple users and there are research projects on additional hardware for multi-user Oculus Rift experiences [27], but this hardware is not currently widely available.

4. Summary

We have described an extension of our VizBET framework to render the VMD model of ET dynamics in a commodity VR platform. The paper details procedures for extracting information from the VMD model, importing them into the Unity game engine, and rendering the final scene in the Oculus Rift. This iBET framework provides a powerful means of understanding not only biological ET processes but also a wide variety of complex dynamical systems. Such immersive visualizations and analyses [28] are expected to provide valuable insight into the microscopic mechanisms of ET processes in not only biological but also other novel nanostructures such as dislocation-mediated metallic nanowires in ceramics [29,30].

VR provides a unique opportunity for researchers in all fields to "walk through" data and identify trends that might have otherwise been obscured. With the increasing availability of low-cost systems such as the Rift, HTC Vive [31], Microsoft HoloLens [32] and Google Cardboard [33], VR is an attractive alternative to traditional data visualization. While iBET has been implemented on the Oculus Rift platform with Unity, the VMD-to-VR workflow is generic and could migrate to other VR platforms and game engines straightforwardly.

Acknowledgments

The authors are grateful for insightful discussions on multiheme cytochromes with Dr. Liang Shi (Pacific Northwest National Lab) and Dr. Tom Clarke (University of East Anglia). KMC simulations by HSB at the University of Southern California were supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, Division of Chemical Sciences, Geosciences, and Biosciences Grant DE-FG02-13ER16415. Additional work on visualization is supported by Air Force Office of Scientific Research Grant FA9550-14-1-0294 to ME-N. The work at Lamar University was supported by the Research Enhancement Grants (REG) fund from Lamar University.

Appendix A.

Appendix

The following C# script animates the color change of hemes in Unity.

```
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;

public class HemeColor : MonoBehaviour
{
  //Declare which site this is in the sequence and how many iterations
  public int HemeNum = 1;
  public int TotHemeNum = 20;
  public int NumIter = 500;

  //Textures
  public Texture2D texture1;
  public Texture2D texture0;
  private Texture mainTexture;

  //Declare variables for sequence
  private TextAsset data;
  private List<int> timing;

  //Begin Flickering Procedure
  //Set speed (larger value is slower)
  public float wait = 0.1f;

  void Start ()
  {
    //Cache renderer component main texture
    mainTexture = gameObject.GetComponent<Renderer>().material.mainTexture;

    //Load sequence from assets/Resources
    data= Resources.Load("patternText") as TextAsset;
    timing = new List<int>();
    //Create text string data
    string txt = data.text;

    //filter out nonbinary entries
    int temp;
    for (int i =0; i < txt.Length; i++)
    {
      temp = txt[i]-'0';
      if (temp == 1 ^ temp == 0)
        timing.Add (temp);
    }

    StartCoroutine ("ColorFlicker", wait);
  }

  IEnumerator ColorFlicker (float wait)
  {
    //infinite loop for animation
    for (int j = 1; j > 0; j++) {
      //select appropriate sequence from timing list
      for (int i = HemeNum-1; i < NumIter*TotHemeNum; i=i+TotHemeNum) {
        if (timing [i] == 1) {
          mainTexture = texture1;
          yield return new WaitForSeconds (wait);
        } else {
          mainTexture = texture0;
          yield return new WaitForSeconds (wait);
        }
      }
    }
  }
}
```

Appendix B. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.jmgm.2016.02.009.

References

[1] R.A. Marcus, N. Sutin, Electron transfers in chemistry and biology, Biochim. Biophys. Acta 811 (1985) 265–322.
[2] I.A. Balabin, X.Q. Hu, D.N. Beratan, Exploring biological electron transfer pathway dynamics with the Pathways plugin for VMD, J. Comput. Chem. 33 (2012) 906–910.
[3] B.E. Logan, Exoelectrogenic bacteria that power microbial fuel cells, Nat. Rev. Microbiol. 7 (2009) 375–381.
[4] C.M. Nakano, H.S. Byun, H. Ma, T. Wei, M.Y. El-Naggar, A framework for stochastic simulations and visualization of biological electron-transfer dynamics, Comput. Phys. Commun. 193 (2015) 1–9.
[5] H.S. Byun, S. Pirbadian, A. Nakano, L. Shi, M.Y. El-Naggar, Kinetic Monte Carlo simulations and molecular conductance measurements of the bacterial decaheme cytochrome MtrF, ChemElectroChem 1 (2014) 1932–1939.
[6] W. Humphrey, A. Dalke, K. Schulten, VMD: visual molecular dynamics, J. Mol. Graphics Modell. 14 (1996) 33–38.
[7] H. Rheingold, Virtual Reality: Exploring the Brave New Technologies, Summit Books, New York NY, 1991.
[8] I.E. Sutherland, A head-mounted three dimensional display, Proc. Am. Fed. Info. Process. Soc. Conf. (1968) 757–764, AFIPS, I (ACM, 1968).
[9] B.A. Davis, K. Bryla, P.A. Benton, Oculus Rift in Action, Manning Publications, Shelter Island, NY, 2015.
[10] H.J. Li, K.S. Leung, T. Nakane, M.H. Wong, iview. an interactive WebGL visualizer for protein-ligand complex, BMC Bioinf. 15 (2014) 56.
[11] W. Goldstone, Unity 3. x Game Development Essentials, 2nd ed., Packt Publishing, Birmingham, UK, 2011.
[12] J.L. Olson, D.M. Krum, E.A. Suma, M. Bolas, A design for a smartphone-based head mounted display, Proc. Virtual Reality Conf. (2011) 233–234 (IEEE, 2011).
[13] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne, The protein data bank, Nucl. Acid. Res. 28 (2000) 235–242.
[14] A. Roy, A. Kucukural, Y. Zhang, I-TASSER: a unified platform for automated protein structure and function prediction, Nat. Protoc. 5 (2010) 725–738.
[15] B.G. Pierce, K. Wiehe, H. Hwang, B.H. Kim, T. Vreven, Z.P. Weng, ZDOCK server: interactive docking prediction of protein-protein complexes and symmetric multimers, Bioinfomatics 30 (2014) 1771–1773.
[16] C. Paissoni, D. Spiliotopoulos, G. Musco, A. Spitaleri, GMXPBSA 2.1: a GROMACS tool to perform MM/PBSA and computational alanine scanning, Comput. Phys. Commun. 186 (2015) 105–107.
[17] J.K. Ousterhout, K. Jones, Tcl and the Tk Toolkit, Second ed., Addison-Wesley, Boston, MA, 2009.
[18] C.R. Myers, K.H. Nealson, Bacterial manganese reduction and growth with manganese oxide as the sole electron-acceptor, Science 240 (1988) 1319–1321.
[19] M.Y. El-Naggar, S.E. Finkel, Live wires, Scientist 27 (2013) 38–43.
[20] J. Albahari, B. Albahari, C# 5.0 in a Nutshell, O'Reilly, Sebastopol, CA, 2012.
[21] D. Flanagan, JavaScript: The Definitive Guide, 6th ed., O'Reilly, Sebastopol, CA, 2011.
[22] J.X. Chen, Guide to Graphics Software Tools, 2nd ed, Springer, Berlin, 2009.
[23] https://www.blender.org.
[24] https://developer.oculus.com.
[25] http://docs.unity3d.com/Manual.
[26] S. Pirbadian, S.E. Barchinger, K.M. Leung, H.S. Byun, Y. Jangir, R.A. Bouhenni, S.B. Reed, M.F. Romine, D.A. Saffarini, L. Shi, Y.A. Gorby, J.H. Golbeck, M.Y. El-Naggar, Shewanella oneidensis MR-1 nanowires are outer membrane and periplasmic extensions of the extracellular electron transport components, Proc. Natl. Acad. Sci. U. S. A. 111 (2014) 12883–12888.
[27] J. Thomas, R. Bashyal, S. Goldstein, E. Suma, MuVR: a multi-user virtual reality platform, Proc. Virtual Reality Conf. (2014) 115–116 (IEEE, 2014).
[28] D. Bhattarai, B.B. Karki, Atomistic visualization: space-time multiresolution integration of data analysis and rendering, J. Mol. Graphics Modell. 27 (2009) 951–968.
[29] A. Nakamura, K. Matsunaga, J. Tohma, T. Yamamoto, Y. Ikuhara, Conducting nanowires in insulating ceramics, Nat. Mater. 2 (2003) 453–456.
[30] K. Tsuruta, E. Tochigi, Y. Kezuka, K. Takata, N. Shibata, A. Nakamura, Y. Ikuhara, Core structure and dissociation energetics of basal edge dislocation in alpha-Al_2O_3: a combined atomistic simulation and transmission electron microscopy analysis, Acta Mater. 65 (2014) 76–84.
[31] http://www.htcvr.com.
[32] https://www.microsoft.com/microsoft-hololens/en-us.
[33] https://www.google.com/get/cardboard.