

GEARS User Manual

v 1.0.0

9. September 2017

1 GEARS in Unity 5

1.1 System Requirements

- **Windows 10 OS** (64-bit Version Recommended)
- **Unity Engine** (Version 5.6.1f1 Recommended)
- *Note: Be sure to also have Unity account created and logged in to the engine*
- Virtual Reality Head Mounted Display (only need one)
 1. **HTC Vive**
 2. **Oculus Rift**
- *Note: Respective runtimes and supporting software for the HMDs are also required*
- For Interactive Demos: **Leap Motion** attached to front of HMD

1.2 Installation

The default installation only includes two of the four demos available. The two non-default demos are the Interactive Viewer and the Virtual Confocal Microscopy. Although these demos provide considerable insight on the strengths of Unity GEARS in VR, they utilize large object files to represent our data meshes. Therefore their installation steps may take extra time depending on the speed of your computer. If one chooses not to include them in their installation, then the large object files will simply be left in their respective .zip files, and the corresponding scenes will be empty.

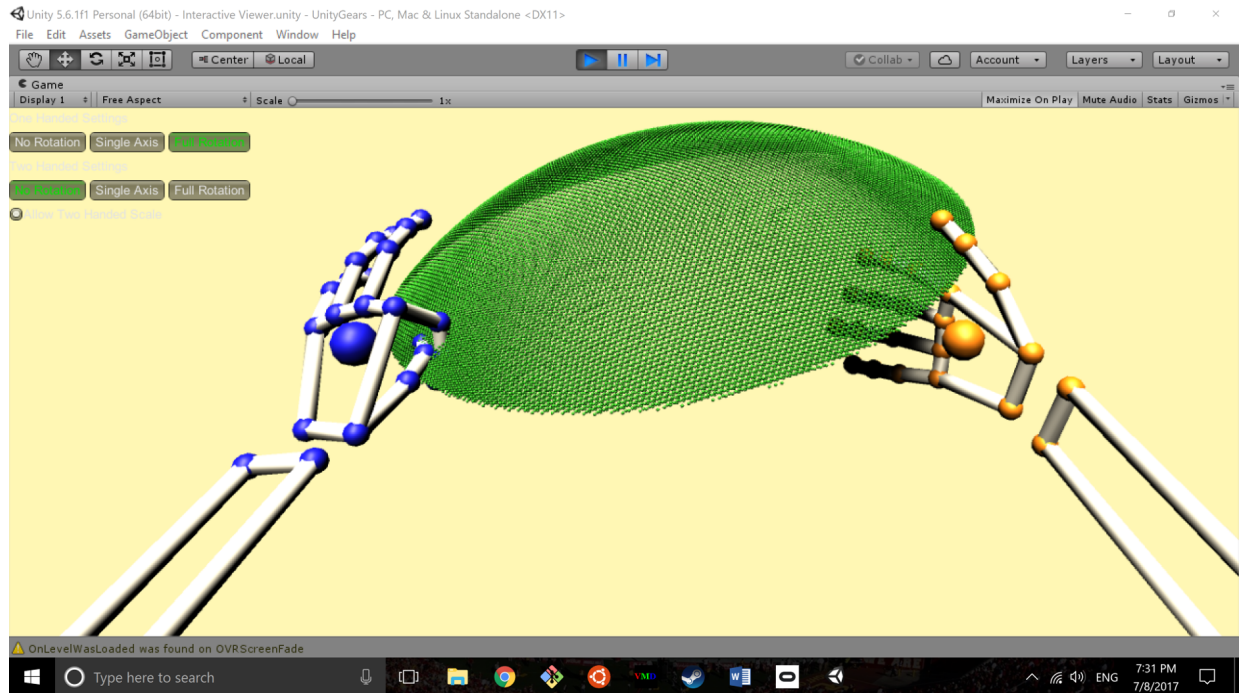


Abbildung 1: UnityGEARS

1.2.1 Setup via Python

Make sure Python 2 is installed on your Windows 10 machine. From the Windows command prompt run:

```
python.exe setup.py
```

If you wish to install the interactive and virtual confocal microscopy demos, include the appropriate flags with the above command like so:

```
python.exe setup.py --interactive --vcm
```

- *Note: This script assumes that your version of Unity is stored at "C:\Program Files\Unity\Editor\Unity.exe"*

1.2.2 Manual Setup

- If you wish to install the interactive and virtual confocal microscopy demos, unzip the following folders:

GEARS\UnityGEARS\Editor\Assets\Demo1-DataViewer\GeometryData.zip
GEARS\UnityGEARS\Editor\Assets\Demo2-VirtualConfocalMicroscopy\GeometryData.zip

- Unzip the following file regardless of whether you choose to install the interactive and confocal microscopy demos:
GEARS\UnityGEARS\Editor\Assets\Leap_Motion_CoreAssets_4.2.1.zip
- Open up Unity.exe and fill out any necessary login credentials.
- Select “Open” to open a new project, and navigate to the GEARs\UnrealGEARS\Editor\ folder and select it. Unity will generate all the necessary project files.

1.3 How to use

Press the play button on the top of the editor window. Switch between demos using the number pad (0-4). Number pad and demo scene correspondences are as follows:

Empty Environment

Demo1 - Interactive Viewer with Leap Motion control

Demo2 - Virtual Confocal Microscopy with Leap Motion Control

Demo3a - Lennard Jones Molecular Dynamics Simulation in real time

Demo3b - Kinetic Monte Carlo simulation of electron transfer in cytochrome proteins (also real time)

1.4 Demo Descriptions

1.4.1 Demo 1 - Interactive Data Viewer for Post-Processing

Here we display a quick method for viewing precomputed simulation data. This method highlights a minimal number steps to viewing your data in VR via game engines. The data in this scene was computed via simulation on a high performance computing cluster (HPCC), then immediately converted into a 3D mesh that we could import into our scene.

Leap Motion controls are also implemented for interaction. Using the Leap, once can grab, reorient, and scale their simulation data with their own hands (no extra controller necessary). This allows for the user to intuitively modify how they view their data, facilitating accelerated data acquisition. The directions for Leap Motion use are as follows: * Make sure that the Leap Motion device is attached to the front of your HMD * Keep your palms wide open about 1 foot or more away from your face * To grab an

object, pinch using your index finger and thumb. You can use one or both hands to then move, rotate, or scale the data object. Note, this grab will always register when the hands in the scene are pinching, so be sure to keep your palms wide open when you don't intend to grab. (The Leap Motion controls are powered by the LeapRTS.cs script, provided by Leap Motion) * One-handed and Two-handed half/full rotations can be toggled using the GUI interface on the computer monitor (not in the HMD). * To scale the size of the data up, pinch with both hands then pull your arms apart as if you were stretching a rubber band. To scale further, unpinch your fingers, bring your arms close together, then repeat the process again.

1.4.2 Demo 2 - Virtual Confocal Microscopy Tool

The previous demo is intended to embolden researchers to adapt existing scientific programs for use in VR, but, for our purposes, it is not enough to simply provide interoperability. To fully realize the promise of immersive scientific computing, we need to develop new tools to enhance these simulations and visualizations. Current data visualization tools, like VMD and ParaView, limit researchers to interaction via the mouse and keyboard; however, with rich programming options, visual scripting capabilities, and an abundance of pre-integrated control schemes, GEARS yields opportunities for immersive, interactive data analysis. An example of one of these tools is our ongoing development of a rendering method called *virtual confocal microscopy*. This method mimics the noninvasive optical sectioning possible in confocal microscopy. Utilizing Unity's surface shader capabilities, we control how each vertex on the structure is rendered to highlight certain areas or planes of the material in the simulation. Our solution seeks to generate a viewing plane that sits in front of the user's head, follows their head movement, and always maintains a set distance from the user. This distance, as well as the thickness of the highlighted viewing plane can be specified and changed by the user, however, for simplicity, this demo excludes the viewing thickness to two modes: on or off.

Controls: * Toggle the confocal plane on and off by pinching your index fingers and thumb together then releasing. * When the confocal plane is turned on, you should only see a slice out of the data object. Every other part should be transparent. Move the confocal plane through the structure by using your head movements. Don't be afraid to take a few steps backward or forward to scan through the entire data set.

1.4.3 Demo 3 - Real Time Simulator

To enhance the dynamic aspects of GEARS, in these demos we go beyond recreations of previously computed results to generate immersive and interactive VR simulations. These simulations run in real time on the engine, allowing the user to put on their

head set and jump right in. Although simulations follow many different paradigms, we provide examples in molecular dynamics and kinetic Monte Carlo, both of which can utilize our methodologies for real time simulation rendering in VR: Run-and-Render and Render-when-Ready (discussed further in our publication tbd).

A. Molecular Dynamics (MD) This demo consists of a system of particles under a Lennard Jones potential. The code was translated to C# directly from Aiichiro Nakano's sequential **MD code** written in C. This particular simulation utilizes the Run-and-Render method where we calculate a time step on the Update() function (called once per frame in Unity). The game waits for this calculation to finish then immediately update our particle positions B. Kinetic Monte Carlo This demo consists of a system of 20 heme sites on a cytochrome protein undergoing electron transfer. The code was translated to C# from Hye Suk Byun's simulation from [insert citation]. This particular simulation utilizes our Render-when-Ready method where we offload the time step calculation onto a separate thread. The game thread then continues, only updating the heme occupation states when user-dictated time has passed. The speed at which the position updates occur is controlled by the slider floating in front of the protein. The user can slow down and speed up the simulation using the left and right arrow keys. When the slider is all the way on the left, the simulation is paused.

1.5 Customization

What good are our demos gonna do for your research? If you really want to have fun with GEARS, you gotta adapt it to your own schtuff. Here's how you can drag and customize the simulation data. I'll be referring to several "windows" within the Unity editor in this section. If you are unfamiliar with the nomenclature, then we suggest reading through **this section** in the Unity manual first.

1.5.1 Demo 1

Getting your data into a Unity scene

1. Mesh-ify your data Turning your data into a 3D mesh is how we're going to get Unity to recognize it. Unity supports several **3D formats**, but for our demo we only use obj files. We ran an MD simulation on an HPCC, then output the atom coordinates in a single frame to an .xyz file. This .xyz file was loaded into **Visual Molecular Dynamics (VMD)**, then rendered into the Wavefront format (.obj and .mtl). Once rendered, we imported the wavefront file by simply dragging and dropping it into the Unity editor **Project window** from the File Explorer.
2. Create a new Unity **scene** or work off of ours

(located in \Editor\Assets\Demo1-DataViewer\InteractiveViewer.unity)

3. Drag & Drop your data .obj file into the scene from the Project window to the **Hierarchy window**.
4. Adjust your data's transform to a comfortable position, rotation, and scale relative to the camera in the scene. Do this via the **Inspector** window after selecting your data in the Hierarchy window.

Leap Motion Controls Doing the above steps will get your data into the scene and ready to view in VR. However, if you would like to have some control over your data while you're viewing it, a control scheme will be necessary. Unity supports many types of controllers and **input methods**, however, for our demos we chose to use the Leap Motion for its gesture control library and well supported **prefabs** which can be brought into Unity scenes without any extra programming. Assuming the Leap Motion library folder was properly unzipped during the installation step described above, one can implement Leap Motion controls into our project template as follows:

1. Replace the camera
2. Create Pinch Detectors
3. Attach LeapRTS.cs to data object

1.5.2 Demo 3a and 3b

Here's how you start simulating and visualizing things in the engine. For more detail on that, refer to our publication. 1. Run and Render 2. Render When Ready

2 GEARS in Unreal Engine 4.16

2.1 System Requirements

- Windows 10
 - Unreal Engine 4.16
 - VR Head Mounted Display (HMD)
1. If using HTC Vive, Steam and SteamVR is required
 2. If using Oculus Rift, Oculus Runtime is required
- Game Controller with Directional Pad
 - Tested Controllers: Xbox 360 Controller, Xbox One Controller, and Oculus Remote

- Visual Studio Community 2017

2.2 Installation

1. Install the Epic Games Launcher and Unreal Engine 4 from the [Epic Games](#) website. The current project was developed in version 4.16.
2. For VR mode, install one of the HMDs runtimes.
 - [Oculus Rift](#)
 - [HTC Vive](#)
3. Generate Visual Studio files by right clicking the Unreal project file (Lammps-VR.uproject) in the LammpsEditor directory. If your computer does not associate .uproject files with the Unreal Editor, then you may have to open up the .uproject file via the Epic Games Launcher first.

2.3 How to use LammpsVR Editor

1. Real-time Simulation Mode
2. Make a new level
3. From Blueprints/Simulation, drag the BP_LammpsController blueprint class into the level editor preview window (setting to the origin works fine).
4. Setup the BP_LammpsController (via the Details window in the Unreal Editor):
 - LAMMPS Dll and Input Script
 1. Specify the dll you wish to use under the “Dll Name” parameter in the “Lammps” Category.
The dll must be located in the LammpsEditor/Content/LammpsResource/LammpsDll/ directory.
 2. Specify the LAMMPS input script for the simulation in the “Input Script” parameter under the “Lammps” category. This script must be located in the LammpsEditor/Content/LammpsResource/Scripts/ directory.
 3. Choose either Simulation Mode or Animation Mode
 1. Simulation Mode
Make sure that the “Animation Mode” box is unchecked.

2. Animation Mode

Check the box label “Animation Mode”. You will then need to specify which time steps to visualize based on LAMMPS dump files. LAMMPS dump files must be located in the

Content/LammpsResource/LammpsDump/ directory. Dump filename must start with a prefix that you specify, followed by a “.” and the time step number. Each dump file should only contain the state of a single time step. Example: “fracture.1234567.dump”. Prefix, starting time step, final time step, and animation time step intervals can be specified under the “Animation” category in the BP_LammpsController Details window.

- Particle Colors and Radii By default, a particle visualization manager will assign a fixed particle radius and random color to all particle types in the LAMMPS instance. However, if you would like to customize the radii and color of your particles, you can edit them under the “Particle Management” category in the Details windows of the Unreal Editor. To customize:
 1. Select the “+” sign next to the “Particles” parameter under the “Particle Management” category to add a new particle type to provide a custom color and radius for a new particle
 2. Specify the particle’s LAMMPS type number, radius (in atomic units), and color.

5. Default User Controller Setup:

6. Drag a player spawning point into the editor preview window. The blueprint for this can be found in the Modes window of the Unreal Editor.

7. In the World Settings windows of the Unreal Editor, set the “GameMode Override” parameter to BP_VRGameMode. Then under “Selected GameMode”, set the “Default Pawn Class” to “BP_VRPawn”

8. If you would like to create your own player controller, then this step is not necessary.

9. LAMMPS Input Script Rules:

- The current working directory gets set to the LammpsVR/Content/LammpsResource/. Therefore, if a LAMMPS input script references another file, the path to that file must start from the LammpsResource directory.
- LAMMPS commands that are not compatible with the compiled DLL will cause the LAMMPS instance to freeze or crash. For example, MPI is not included in our provided DLL. Therefore, if a LAMMPS input script makes use of the “processor” command, the engine will crash. The engine also freezes in instances

where the LAMMPS input script references a file that does not exist. Therefore, always double check your path names.

- For animation mode, the LAMMPS input script must setup the LAMMPS system before running. Because the animation mode does not actually use any of the force calculations, dummy values for the system initial parameters can be used. For example, one can setup the animation system in their LAMMPS input script via the following LAMMPS commands:

```
boundary      p p p
atom_style atomic
read_data path/to/sim/data/file.data
pair_style zero 5.0
pair_coeff     * *
mass 1 1.0
mass 2 1.0
mass 3 1.0
mass 4 1.0
thermo_style custom step temp etotal
thermo 1
```

2.3.1 Demo 1: Real-time Simulations

For this demo, we display the real time simulation capabilities of GEARS. In the Content/Levels/Demo1/ directory, you'll find several sample levels each of which is running a different LAMMPS simulation. Through these data sets you will see the various LAMMPS commands that the system is able to handle as well.

- The first example is a basic Lennard Jones molecular dynamics simulation, where all particles are of a single type.
- The second example is a simulation of Nano-porous Silica. In this example, the initial configuration is set via a separate .data file and the Vashishta pairstyle is utilized in the simulation.
- The third example simulates chemical vapor deposition, using both an initial configuration, pair style, and reaxff force field package compiled into our LAMMPS DLL.

2.3.2 Demo 2: Animation

Fracture = rerunning precomputed dump files