Virtual Reality in Unity as a Tool for Scientific Computing and Visualization
Brandon Horton, Erick Moen, Michael Qian, Aiichiro Nakano

**Introduction**
- The Importance of Visual Data: Anscombe’s Quartet
- 2D → 3D Natural venue for VR
- The CAVE: → Bulky and Expensive (Obsolete?)
- VR Head-Mounted Device → Inexpensive and Portable

**Tools**
- Modern Game Engine: Unity or Unreal
- VR HMD: Oculus, HTC Vive
- Controller: Leap Motion, Xbox

**Workflow for Scientific VR in Unity**

**Immersive Data Visualization**
- VMD-generated object files quickly loaded into Unity
- Leap Motion replaces keyboard and mouse for more direct controller experience
- Unity vertex and fragment shaders help dynamically render pieces of the desalination membrane
- Rendering plane follows head movement via Oculus for intuitive viewing control

**Beyond Visualization: Active Simulation**
- Not limited to visualization → Unity capable of executing simulations
- C# Kinetic Monte Carlo Code spawned on new thread
- Simulation populates an animation queue, while Unity pops according to user-specified rate → Interactive Simulation Steering
- Result: Simultaneous simulation and rendering in real-time

**Future Directions**
- Direct pipelining from VMD to Unity from the command line → Direct parsing of simulation data in Unity
- Generic Kinetic Monte Carlo Visualization → Beyond biological electron transfer
- Visualization of High Performance Simulations via Server-Client Communication with Unity

Special Thanks to Masato Nakano, Moh El-Naggar, Hye Suk Byun, and Tao Wei

\[
PPR_{W,\alpha} = \alpha \sum_{i=0}^{n}(1 - \alpha)^i \left( \frac{B_{W,\alpha}}{W} \right)^i
\]