New Features in OpenSim 4.0

A modern visualizer for creating beautiful images and movies. Researchers, teachers, and clinicians need to make clear and compelling visuals to share their work with others. Researchers also need fast and flexible visualizations to help them build and troubleshoot models and simulations. The new visualizer in OpenSim 4.0 allows users to:

- Easily add and adjust lighting and shadows
- Automatically visualize analytical geometry in your model, like spheres and planes, without adding meshes to your model file.
- Create movies automatically on 64-bit machines on both Mac and Windows

We provide default lighting that will work well for most users and have updated the look of OpenSim muscles to help you create high impact visuals out of the box.

We're in the process of updating the User's Guide with instructions on how to use these features.

Compatibility with both the Mac and Windows operating systems. OpenSim 4.0 runs natively on both Mac and Windows (i.e., no virtual machine needed). See Supported Platforms for more information.

Improved support for motion capture data in the C3D file format. We know that importing your experimental data to use in OpenSim can be challenging and time-consuming. In OpenSim version 4.0, by incorporating the BTK package (created by Arnaud Barre), we now support reading marker and force data from C3D files. In general, you will still need to create Matlab or other scripts to read in your data, but new functionality makes this process much easier. We also provide Matlab utilities to (i) read Data into OpenSim-compatible data tables, (ii) output data as Matlab Structures, (iii) rotate marker and force plate data, and (iv) write marker and force plate data to OpenSim file format (.trc and .mot, respectively). See our documentation on C3D (.c3d) Files for more information.

Custom modeling and simulation studies via Matlab, Python, and C++. We have overhauled OpenSim's API or Application Programming Interface to make it easier for researchers to create their own custom workflows in either Matlab, Python, or C++.

- Any Component in OpenSim can generate Outputs (e.g., a Muscle can output its force-generating capacity), which users can report to an internal table, a file, or the console using a Reporter.
- The OpenSim API now also provides tools to manage the exchange of data within a simulation. Any Output generated by a Component (e.g., a Muscle's fiber length) can be received by another Component as an Input (e.g., a stretch-based reflex Controller that generates an excitation signal based on a Muscle's fiber length as an Input).
- DataTables are in-memory containers that can be used to store experimental and simulation data (e.g., marker locations, muscle-fiber lengths, and excitation signals).

Read more in our new API Guide.

Frames & Points to define, track, and report reference frames and points of interest. A Frame is an OpenSim representation of a reference frame and a Point is an OpenSim representation of any location in space. Frames and Points provide a convenient way to locate physical structures, such as joints and muscle attachments, and perform spatial calculations. For example, if your system includes contact, you might define a Frame that is aligned with the normal direction of a contact surface and whose origin is at the center-of-pressure. You can then easily report the contact forces normal to the contact surface and the location of the center-of-pressure, without repeatedly performing the transformations in your main or analysis code. Read More in our API Guide.

The **new StatesTrajectory class** allows scripting users to easily perform post-hoc analyses on simulation results. Previously, users would need to load a Storage file and manually reconstruct a State for each time point of the motion before proceeding with the desired analyses (e.g., computing fiber force at each state in a motion). The StatesTrajectory class automates this process and directly provides users with a trajectory of states from a Storage file. See the example Code/Python/posthoc_StatesTrajectory_example.py in the Resources directory.

Bug Fixes

We've fixed many bugs. See a full list.

New Examples and Documentation

- Examples are now stored in a "Resources" directory of your choosing when you install OpenSim (e.g., C:/Users/<username>/Documents /OpenSim/4.0). We've provided several new utilities via Matlab and GUI scripting. These utilities will help you read in your experimental data and edit models. Find them in the Code/Matlab and Code/GUI folders in the Resources directory.
- See how to use OpenSim through Matlab to convert C3D files to .trc (marker) and forceplate data files (Code/Matlab/Examples/c3dExport.m)
- Learn about the new features of the API by building an assistive device to help a simple hopper model jump (Code/CPP/ExampleHopperDevice and Code/Matlab/Hopper_Device in the Resources directory).
- Learn how to build a complex model with a parallel mechanism via the new Luxo Lamp C++ example (Code/CPP/ExampleLuxoMuscle in the Resources directory).
- We've created a new API Guide for developers using the API in Matlab, Python, or C++.