Manipulating Proteins in PyMOL with a Wii Remote

Project Proposal for BME220
Taught by Professors Karplus and Gerloff, Spring, 2007
Submitted by Jonathan Magasin (jmagasin@soe.ucsc.edu)

Motivation

Research in protein structure depends critically on good software tools for visualizing molecular models. While programs such as PyMOL impressively render these models in 3D, their 2D mouse-based UI's make interacting with the models nonintuitive and oftentimes frustrating. A 3D input device could provide a natural way to interact with a protein model, if only an affordable one existed. Fortuitously, Ian Rickard has set out to create such a device based on Nintendo's relatively inexpensive Wii Remote. His design uses two Wii Remotes as cameras to track the motion of a third Wii Remote to which infrared LEDs have been affixed. The IR LEDs allow the camera Wiimotes to track the user's motions with the third Wiimote. Ian recently completed a prototype device which we are eager to apply to PyMOL. The purpose of this project is to enable PyMOL users (at UCSC at least) to interact naturally with protein models via the Wiimote. The outputs of the project will include:

1. A Wiimote interface utility that smoothes, translates, and buffers the Wiimote data stream and that exports a general API for applications to detect Wiimote events. This utility should have no PyMOL dependencies so that other viewers (perhaps RasMol or Protein Explorer) may support the Wiimote in the future.

2. User interface support for converting Wiimote events raised by the interface utility into PyMOL events for the viewer. It will be ideal for the user if PyMOL's Wiimote support makes no changes to the core PyMOL code (requiring the user to recompile). However, an external module for the UI event handler may be infeasible due to performance or other factors.

3. Documentation explaining how to set up the PyMOL, the Wiimote and all support software and APIs will be written.

Of course we hope this project will also provide useful feedback for Ian's continued efforts on the Wiimote.

Chris Szeto and I would like to work on this project jointly, with one of us focusing on the Wiimote interface utility, the other on the user interface.

Milestones and tasks

Bold items are milestones in rough order of completion (though worked on in parallel). The tasks within each milestone have very rough estimates for number of days of effort.

M1. Specify PyMOL UI for Wiimote and API for interface utility

- Meet with regular users of RasMol or PyMOL to discuss usability problems and ways they might would like to use the Wiimote with a viewer. Read papers on 3D UI's (one mentioned in references below). (~1wk)

- The API defined by the Wiimote interface utility will be affected by the types of operations the user wants to do. Supporting vibrational feedback in the Wiimote would require a bidirectional API. Whether button, rotation, and translation events are distinct events, and how frequently they are raised might be affected. For example, perhaps the user intends for Wii movements to result in viewer updates when a "grab" button is pressed. There may also need to be a control API for changing event rates on the fly (perhaps for controlling coarseness of motion). (~1wk)

M2. Develop skills needed for implementing

- PyMOL: Another day or two of using PyMOL to understand better its features and UI problems will be helpful. Getting familiar with the code base will take some effort. (1-2 days)

- Learn Python. (1-2wks)

- Learn TCP/socket support for getting Wiimote input stream, and possibly for PyMOL as well since Python has straightforward socket support. (1-2 days)

- Review/learn linear algebra for converting from coordinate systems used by Wiimote app (Cartesian and Quaternion) to that expected by the PyMOL UI. (1 week?)

- Set up tools and hardware for developing with Wiimote. Ian expects it will take about two to three weeks to prepare another prototype. I imagine a few days to a week before things are running smoothly on my and Chris's laptops.

M3. Write design and implementation specification

We can work on this in parallel with the above tasks. The basic design should be pretty straightforward: The Wiimote interface utility will be a separate process. It will read the input stream (TCP socket), clean up the data as necessary and convert to the coordinate system expected by PyMOL, buffer it, and raise an event for the PyMOL UI. PyMOL, ideally through an add-on module, will convert the event to viewer commands to update the displayed model. (~1-2wks, with likely refinements)

M4. Implement, tune and test

Getting a working version up soon, even with a reduced feature set, will likely be important for the success of the project. First, it will allow us to work more productively in parallel on the PyMOL UI and interface utility. Second, it will allow us to seek help earlier if needed from Ian for any changes to the Wiimote apps. Third, it will give us greater time for tuning so that the UI doesn't just work but, hopefully, is intuitive and fun to use.

User documentation and poster
A tutorial for setting up the Wiimote and the supporting apps will be written. I also hope to participate in the poster session this spring.

Open issues

1. Ian's Wiimote apps run on his MacBook, so it should be straightforward for Chris (MacBook Pro) and I (MacBook) to get set up for development on our laptops. One unknown is whether the MacBook, with limited OpenGL hardware support, can handle both the Wiimote motion processing and PyMOL rendering. PyMOL by itself performs very well on my MacBook (even with large proteins rendered as balls or surfaces). Also, the frame rate for the Wiimote (currently 100fps) is probably higher than we would need for PyMOL. So I think performance problems are unlikely.

2. What platforms will be tested? The interface utility and PyMOL UI changes will be platform-independent, but testing them on non-Macs entail significant additional work.

3. Currently the Wiimote motion data is accurate to within about 1 cm, though once triangulation is implemented accuracy is expected to improve to about 1-2mm.

Resources

- Ian Rickard has already been very gracious with his time in meeting with us to give an overview and demo of the Wiimote. We will need his assistance throughout the project for Wiimote support.

- Suggestions from Professor Karplus on the UI design should be especially valuable.

References

Documentation and tutorials for learning Python or about TCP/socket APIs pervade the web. One useful Python tutorial I will use is:

van Rossum, G. *Python Tutoria*l (2006) on *World Wide Web* http://docs.python.org/tut/

The PyMOL home page is a useful starting point for finding all things PyMOL. It has binaries, documentation, source code (links) and project status information.

Delano, W.L. *The PyMOL Molecular Graphics System* (2002) on *World Wide Web* http://www.pymol.org

RasMol and its successor Protein Explorer have UI's regarded as superior to PyMOLs and may inspire some features for our UI. They should also be considered when designing the interface utility API so that this project might enable future Wii remote support for RasMol.

Sayle, R., Milner-White, E.J. “*RasMol: Biomolecular graphics for all*”, *Trends in Biochemical Sciences (TIBS)*, September 1995, Vol. 20, No. 9, p. 374.

For 3D UI design one brief paper of interest with copious additional references is:

Leach, G., Al-Qaimari, G., Grieve, M., Jinks, N., McKay, C. *Elements of a Three-dimensional Graphical User Interface* (1996) on *World Wide Web*
http://goanna.cs.rmit.edu.au/~gl/research/HCC/interact97.html