ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WESH4002

A PyDM USER INTERFACE FOR AN LCLS SIMULATOR

M. Gibbs[*], W. Colocho, J. Shtalenkova, A. Osman, H. Slepicka
SLAC National Accelerator Laboratory, Menlo Park, USA

Abstract

PyDM (Python Display Manager) is a framework for building control system user interfaces. A user interface for the LCLS (Linac Coherent Light Source) simulator has been built in PyDM. The simulator interface gives a realistic experience of operating many parts of the LCLS accelerator, and can be used for training new accelerator operators on routine tasks. This interface also provides a good demonstration of the experience of using PyDM in a real-world environment.

SIMULACRUM: AN LCLS SIMULATOR

Recently at the SLAC National Accelerator Laboratory, an LCLS accelerator simulation system called Simulacrum has been developed. This simulator runs an accelerator model, and uses the model to populate EPICS PVs that mimic the real accelerator. The goal of Simulacrum is to let users take unmodified accelerator software applications used to operate LCLS, and run them against a simulator that could be running on a development server, or even on a user's laptop. The reverse is also true: users can develop new applications or displays against Simulacrum, using the same code that they would to interact with the real accelerator's control system.

While Simulacrum was originally developed as an aid in developing accelerator software, but it has also been identified as a training tool for operators. It can help new operators gain some intuition of how the electron beam behaves, and let them explore in an environment free of the time pressure and up-time demands that come with a facility delivering beam to user experiments. Rather than just reading training documents and procedures, the tools used to operate the machine can be used in the simulation.

PyDM

PyDM [1] is a framework for building control system graphical user interfaces (GUIs). PyDM is built with Qt [2] and the PyQt [3] Python-Qt bindings. It is packaged with a set of widgets appropriate for interacting with scientific and industrial equipment. PyDM is also highly extensible by users, who can add new widgets, support for new data sources, and add custom client-side logic to control system displays. At SLAC, some portions of the GUI used to run the LCLS accelerator are being replaced with new PyDM-based displays. Simulacrum is being use to test the new displays while most of the control system is offline during facility upgrades. The PyDM simulator interface presented here shares many of the same files that will be used for the real LCLS accelerator.

[*] mgibbs@slac.stanford.edu

AUTO-GENERATION OF DISPLAYS FROM DEVICE LISTS

PyDM displays can use Python code to build themselves from lists of devices. In this interface, each display queries an EPICS PVAccess table PV for a list of relevant device types, and generates user interface elements for every item in the list. By using this method, the same interface files can be used against any accelerator lattice definition used by the simulator.

PyDM's Template Repeater Widget is used frequently in these displays. This widget takes a user interface file representing the controls for a single device, along with a list of devices supplied as a list of Python dictionaries, and renders the user interface file once for each item in the list.

DISPLAYS

The top-level interface is organized by area and subsystem, laid out in a grid. For example, a user might select the column corresponding to ”LI25” (linac sector 25), and the row corresponding to ”Magnets”. This will take the user to a listing of all magnets in the simulated sector 25. PyDM's browser-like navigation tools let the user page forward and backward through displays they have visited, or go directly back to the top-level ”Home” display.

An alarm tree in the simulator collects low-level alarms into summary alarms. These summary alarms are propagated through the displays, up to the top level, allowing the user to quickly identify and traverse the displays to find the relevant screen to address the alarm.

Magnets

The magnet displays are organized in tabs for each type of magnet. In each tab is a straightforward list of controls, one row for each magnet. The controls provide the basic functions one would expect: magnet strength can be changed, via a line edit widget or slider. Other functions, saving the magnet field setpoint, or loading a saved setpoint, are available from a drop-down menu.

Collimators

The collimator displays contain controls to move collimator jaws directly, or set ’center’ and ’gap’ values that move two jaws together. The collimator displays utilize PyDM's ”Widget Rules” feature to animate a schematic of the collimators: the on-screen position of two widgets representing the collimators are bound to the PVs that report the collimator jaw positions.

RF

The RF displays show information and control for the klystrons in the simulation. Each klystron has controls for a phase shifter, controls for the klystron's high voltage power supply, and many status readbacks for klystron interlocks.

Tuning

The tuning display shows a plot of simulated x-ray pulse intensity versus time. Along with the intensity plot are controls for commonly-used magnets for tuning the pulse intensity. The display uses Python code to save "checkpoints" from which one can re-load magnet settings.

Beam Position Monitors

The beam position monitor displays plot the electron beam's position using a custom widget, described in detail in the next section. Python code is used to fetch the z-positions of each beam position monitor along the beam line in the selected area, and set the viewing range of the beam position plot accordingly.

CUSTOM WIDGETS

The "Steering" display, and the custom beam position it utilizes is an example of highly specialized, task-specific code that can be embedded in a PyDM display. The Steering widget draws a pan-and-zoomable plot of 176 beam position monitor values that update in real time. Superimposed on the plot are buttons to increment and decrement dipole corrector magnet strengths to adjust the electron beam's trajectory. The code for this display was extracted from a stand-alone PyQt electron beam orbit display application, and can now be embedded and re-used in any display where a real-time view of the orbit (or a portion of it) is desired. While in the simulator, the beam's position is updated at less than 10 Hz, this widget has been used to display real orbit data at 60 Hz.

CONCLUSION

A set of PyDM displays were created as a graphical user interface to Simulacrum, an LCLS accelerator simulator. The accelerator simulator provides an ideal test-bed for control system screens, and includes a large variety of the widgets and data types that would be used in a real control system. A custom widget for linac beam steering have been

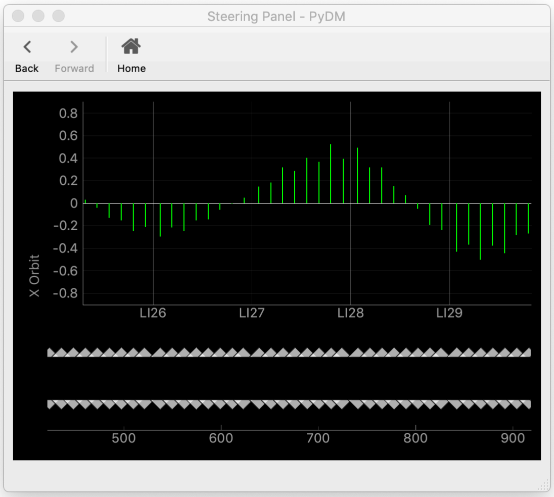

Figure 1: Linac Steering Widget in PyDM. The beam's transverse position (in the X or Y axis, depending on the panel being used) is plotted as a green bar. Below the plot of beam position measurements, increment/decrement controls for each corrector magnet are displayed. By clicking and dragging the plot, the user can pan the view to see different areas. Zooming in and out is possible by scrolling with the mouse wheel or trackpad.

added (see Fig. 1), demonstrating some of PyDM's potential as a platform for high-level application development. The interface closely resembles the real LCLS accelerator user interface, and is a good tool for training purposes. The simulator interface will also be used for community outreach activities, to give the public a glimpse at operating a particle accelerator.

ACKNOWLEDGMENTS

The authors would like to thank all of the contributors to PyDM.

REFERENCES

[1] M. Gibbs and H. Slepicka. (2019). PyDM - Python Display Manager, `https://slaclab.github.io/pydm/`
[2] The Qt Company. (2019). Qt, `https://www.qt.io`
[3] Riverbank Computing Limited. (2019). PyQt, `https://riverbankcomputing.com/software/pyqt/intro`