ORBIT DISPLAY

Matt Gibbs
Accelerator Operations Specialist for LCLS
**Orbit** (noun) - The trajectory of the beam in a particle accelerator. Typically measured at several discrete locations by beam position monitors (BPMs) installed throughout the beam line.
PREVIOUS TOOLS FOR VIEWING THE ORBIT AT SLAC
PREVIOUS TOOLS FOR VIEWING THE ORBIT AT SLAC
ORBIT DISPLAY

PREVIOUS TOOLS FOR VIEWING THE ORBIT AT SLAC
### PREVIOUS TOOLS FOR VIEWING THE ORBIT AT SLAC

<table>
<thead>
<tr>
<th>Feature</th>
<th>EDM “Fast Orbit”</th>
<th>Java “Orbit Display”</th>
<th>MATLAB “BPMs vs. Z”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Update Rate</strong></td>
<td>60 Hz</td>
<td>10 Hz</td>
<td>5 Hz</td>
</tr>
<tr>
<td><strong>Reference Orbits</strong></td>
<td>No</td>
<td>Yes, from live data and config saves</td>
<td>Yes, from live data</td>
</tr>
<tr>
<td><strong>Saves to Logbook</strong></td>
<td>No</td>
<td>Yes, raster images</td>
<td>Yes, Postscript files</td>
</tr>
<tr>
<td><strong>Configurable Viewing Window</strong></td>
<td>Minimal choices</td>
<td>Yes</td>
<td>Multiple pre-defined options</td>
</tr>
<tr>
<td><strong>Fit Data to Model</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
GOALS FOR A NEW DISPLAY

- 60 Hz update rate
- List of BPMs and their positions automatically retrieved from model
- Reference orbits from live data, saved orbit files, and saved machine configs
- Saves logbook data in publication-usable format
- Fully configurable viewing window
- Can fit orbit data to machine model in real-time
**EDM?**

+ Fast
+ Easy to use

- Minimal interactivity
- Hard to Maintain
ORBIT DISPLAY

PLATFORM

JAVA?

- Slow to launch and slow to run
- No expertise in Operations Group
MATLAB?

+ Easy plotting
+ Operators know it
+ Support libraries for LCLS already exist
- Slow launch times
- Subpar GUI tools
- Difficult to achieve 60 Hz update rate
PYTHON?

+ Easy plotting
+ Operators know it
+ Good GUI tools
+ Fast launch times
+ Fast at run time

- No support libraries for LCLS
ORBIT DISPLAY

PLATFORM

PYTHON

+ PyQt
+ PyQtGraph
+ NumPy
+ Matplotlib
ARCHITECTURE

- Modular
- Put as little functionality in GUI classes as possible
- Abstract base classes allow for flexibility in implementation
郎巻きとパンニングはマウスを用いて容易に変更可能 - フォーカスを失わずにプロットパラメータを調整する必要はありません。

- 自由にサイズを調整してワークスペースにフィットさせることが可能。
- ズームインしたときにより多くの情報を提供する適応型のタックマーク。
- クラッターフリーなメインウィンドウはオーバーヘッドディスプレイとしても使用できる。
ORBIT DISPLAY

Reference Orbit  Measurement  Fitting

View Mode

- Single Orbit: Live Orbit
- Difference between: Live Orbit and 2: (Sep 28 2018 13:32:04), with relative TMIT

Take New Reference Orbit

Reference Orbits

- Live Orbit
  - 2: (Sep 28 2018 13:32:04)
Developing while using live data is not always possible

No beam = No testing

Simulator mode uses a ‘SimOrbit’ class to generate realistic fake data
**SIMULATED ORBIT**

- SimOrbit is just another BaseOrbit implementation
- No changes to the application or OrbitView necessary, besides adding the command line flag
- “ProxyBPM” lets any NumPy array be the data source
MODIFICATIONS FOR XFEL

- During commissioning, a quick way to steer the beam by hand was needed

- New concepts needed for XFEL version:
  - Multiple beam paths
  - Sub-trains
  - Corrector magnet control overlay
  - DOOCS, not EPICS
  - Two data sources: BPM Front-Ends, and Orbit Middle Layer Service
- XFELOrbitDisplay adds widgets for selecting beam destination, sub-train to view, and corrector magnet overlay.

- XFELOrbit implements data collection from DOOCS, where it is efficient to get all orbit data as a NumPy array in one call, so ProxyBPM is used again.

Steering controls overlaid at z-position of each magnet.
CONCLUSION

- Python, PyQt, and PyQtGraph make a good platform for writing very fast interactive accelerator software

- Thinking about the architecture ahead of time pays off - modifications down the road are much easier
THANKS!

QUESTIONS?