An open source GUI for collaborative cloud-based X-ray optics modeling

D.L. Bruhwiler[&] M. Rakitin[#] P. Moeller[&,%] R. Nagler[&] O. Chubar[#]

[&] radiasoft [#] BROOKHAVEN NATIONAL LABORATORY [%] bivio software that works

Software for Optical Simulations (SOS) Workshop

4 October 2016 – Trieste

This work is supported in part by the US DOE Office of Science, Office of Basic Energy Sciences under SBIR awards DE-SC0006284 and DE-SC0011237.

Vision

- *develop open source software, useful for:*
  - *X-ray beamline designers & computational physicists*
    - *full access to capabilities & complexity, with minimal restrictions*
    - *flexible, publication quality graphics*
    - *a subset of them require a development environment*
  - *light source facility users*
    - *require a graphical user interface (GUI)*
    - *ease of use, intuitive*
    - *ability to understand X-ray optics of specific beamline*
- *software must be sustainable over the long-term*
  - *all components must be maintainable*
  - *easy for the X-ray optics community to contribute*

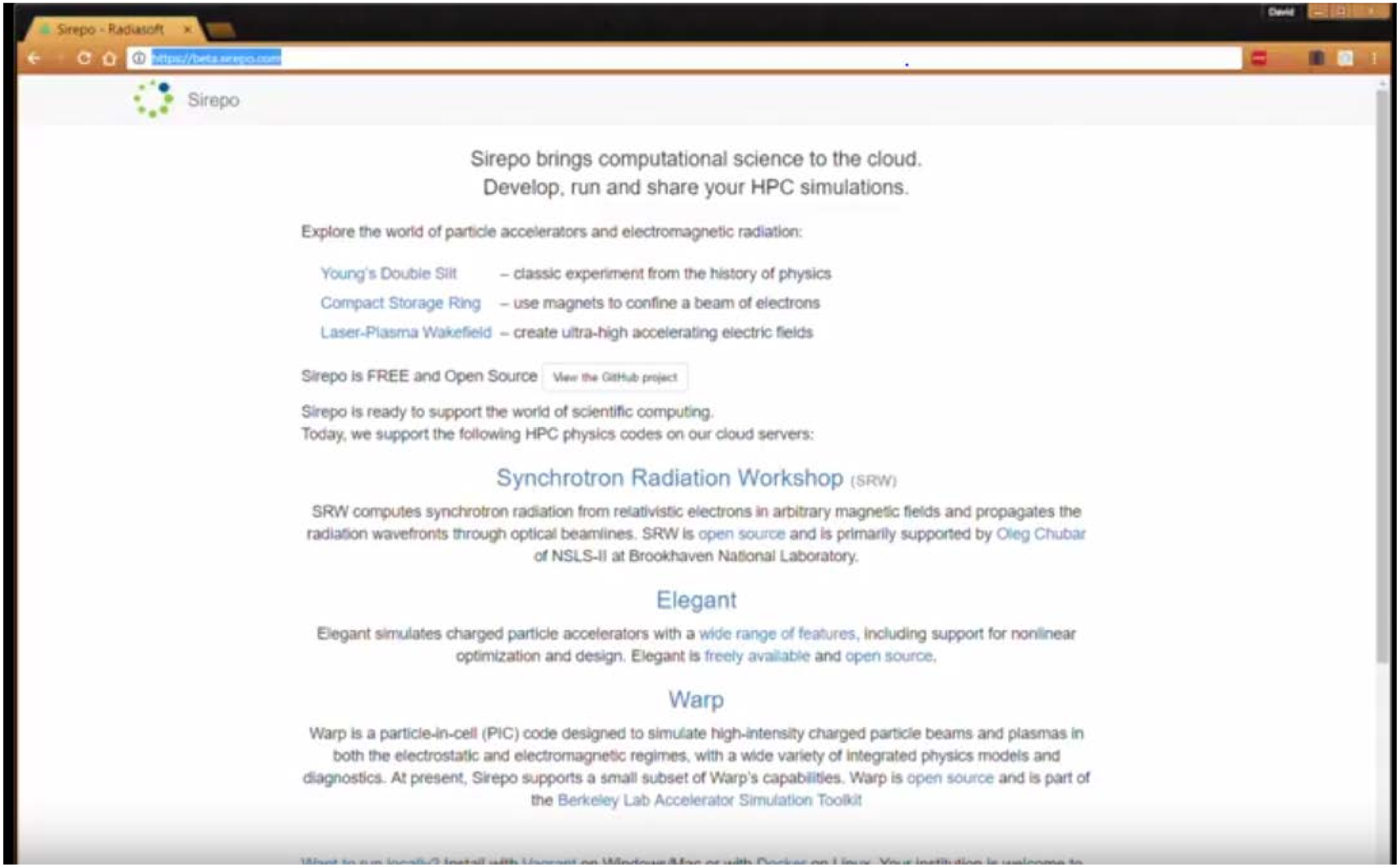
Capabilities Matrix

Capabilities	Software	Comments
electron beam dynamics	SRW, …	more 3D magnetic field sources and elements
synchrotron emission	SRW, …	improved solvers
SASE FEL	GENESIS, database…	import laser pulse data (2D and 3D)
geometric optics	SHADOW, …	interoperability with SRW
physical optics	SRW, …	improve reliability, speed
partial coherence	SRW, SHADOW…	improve reliability, speed
X-ray optical elements	SRW, SHADOW…	increasing diversity & complexity
element imperfections	SRW, SHADOW…	benchmark different methods
material properties	database	automated access to community resources
Python scripting	SRW, SHADOW…	how much commonality?
file formats	SRW, SHADOW…	configuration, binary data – commonality?
GUI	SRW, SHADOW…	cloud computing
Jupyter (IPython) notebooks	SRW, …	WPG effort at EXFEL
simulate experiments	SRW, …	material interactions, diagnostics
optimization & design	SRW, …	inverse algorithms, nonlinear optimizers

Open Source Cloud Computing for Science

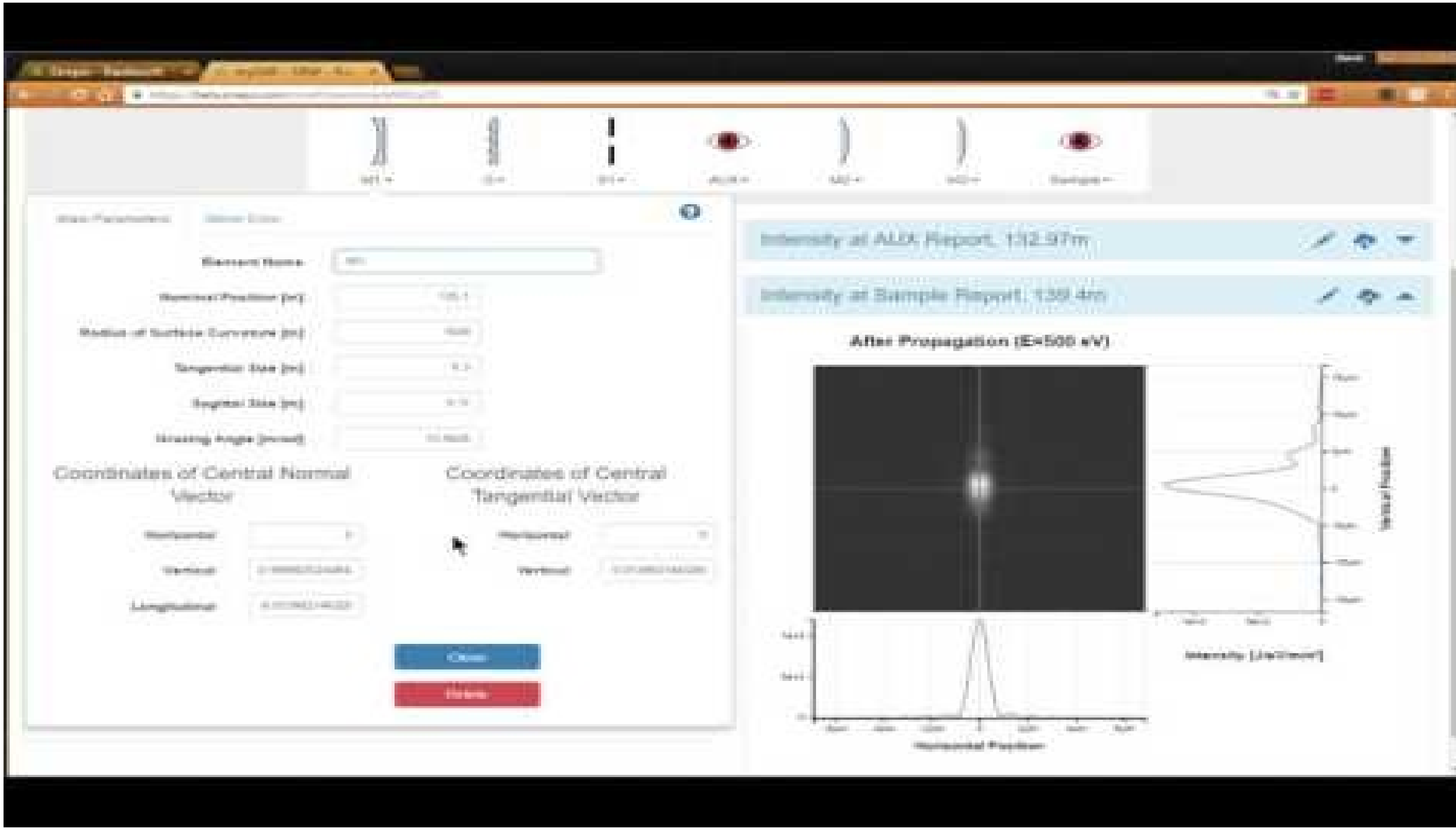
- *The browser is the GUI*
  - *HTML5 technologies, especially JavaScript*
  - *simulation code runs on a server, supercomputer, etc.*
- *Seamless legacy*
  - *export SRW python script, or IPython Notebook from browser*
  - *GUI will always help users, never restrict what they can do*
- *Containerized computing*
  - *open source technology: Docker*
    - *Docker enables rapid cloud deployment*
      - *no overhead on Linux*
    - *headless VMs on Mac OS and Windows with low overhead*
    - *eliminates pain of code installation, cross-platform development*
  - *archival, reproducibility, instantaneous collaboration*
    - *user input files, output files, etc. are saved in the container*
    - *share the container with a collaborator, students, etc.*

SRW in the cloud: https://beta.sirepo.com



https://www.youtube.com/watch?v=1hhivULQwOM

Sharing your SRW simulation, Part 1:

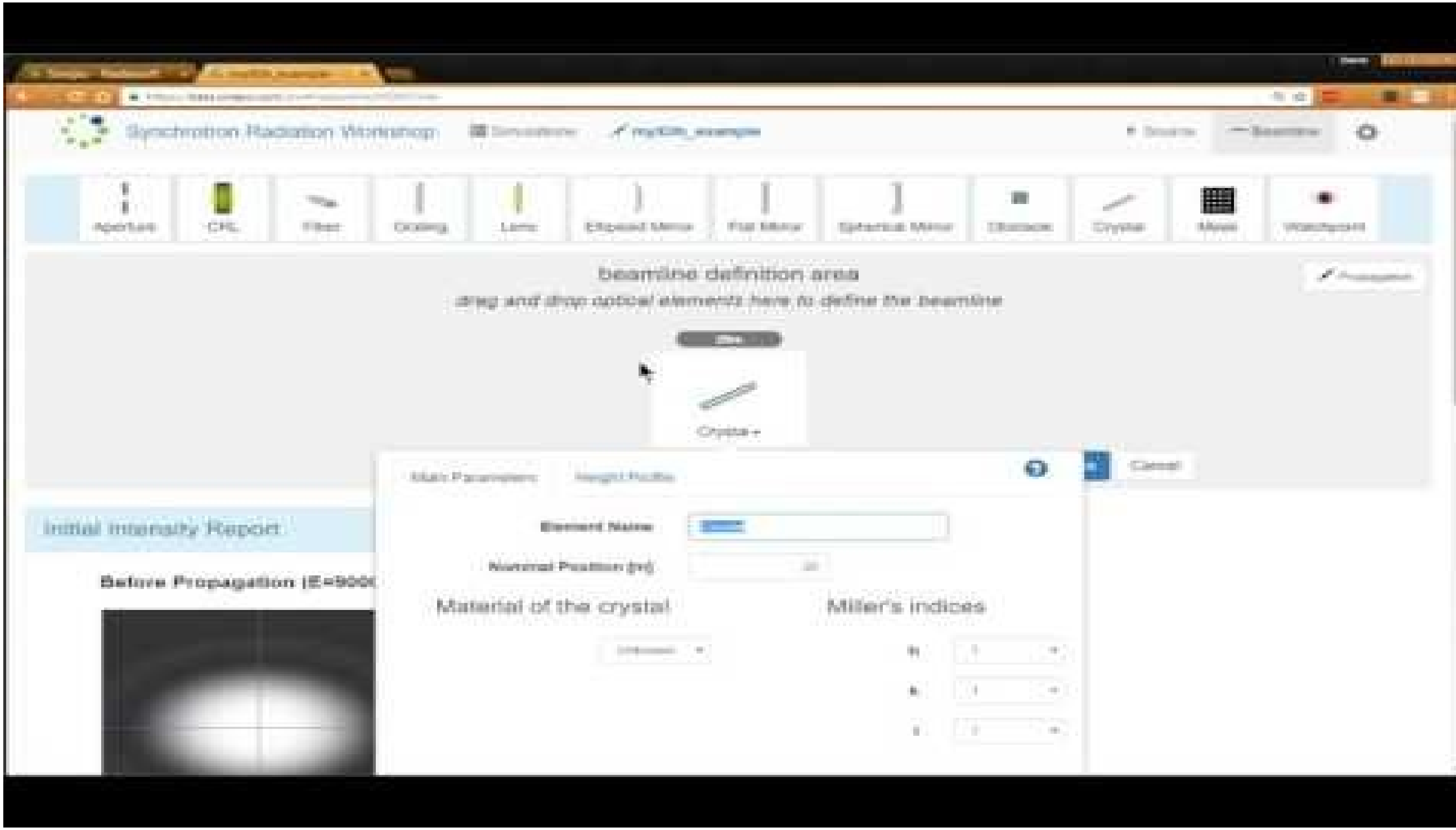


https://www.youtube.com/watch?v=Ioa7TZ3PkGc

Sharing your SRW simulation, Part 2:

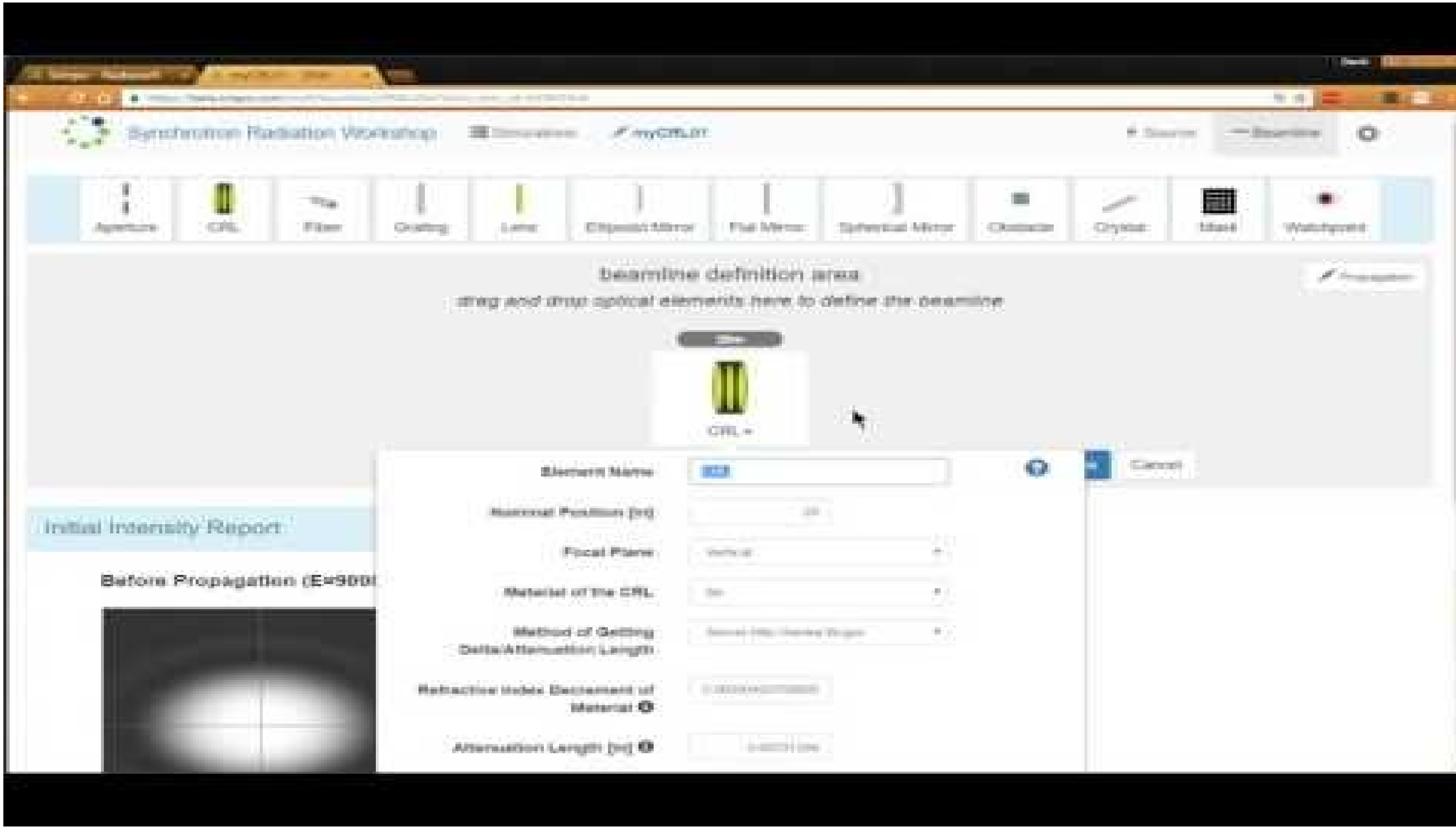
https://www.youtube.com/watch?v=DJBCrmB7Kc0

Automated access of X0h data server at ANL



https://www.youtube.com/watch?v=r6JEAUXfJoI

Automated access of materials data at LBL



https://www.youtube.com/watch?v=d0Vxj8yYZQA

Experts can download Python script

https://beta.sirepo.com/python-source/srw/znElhQ9a

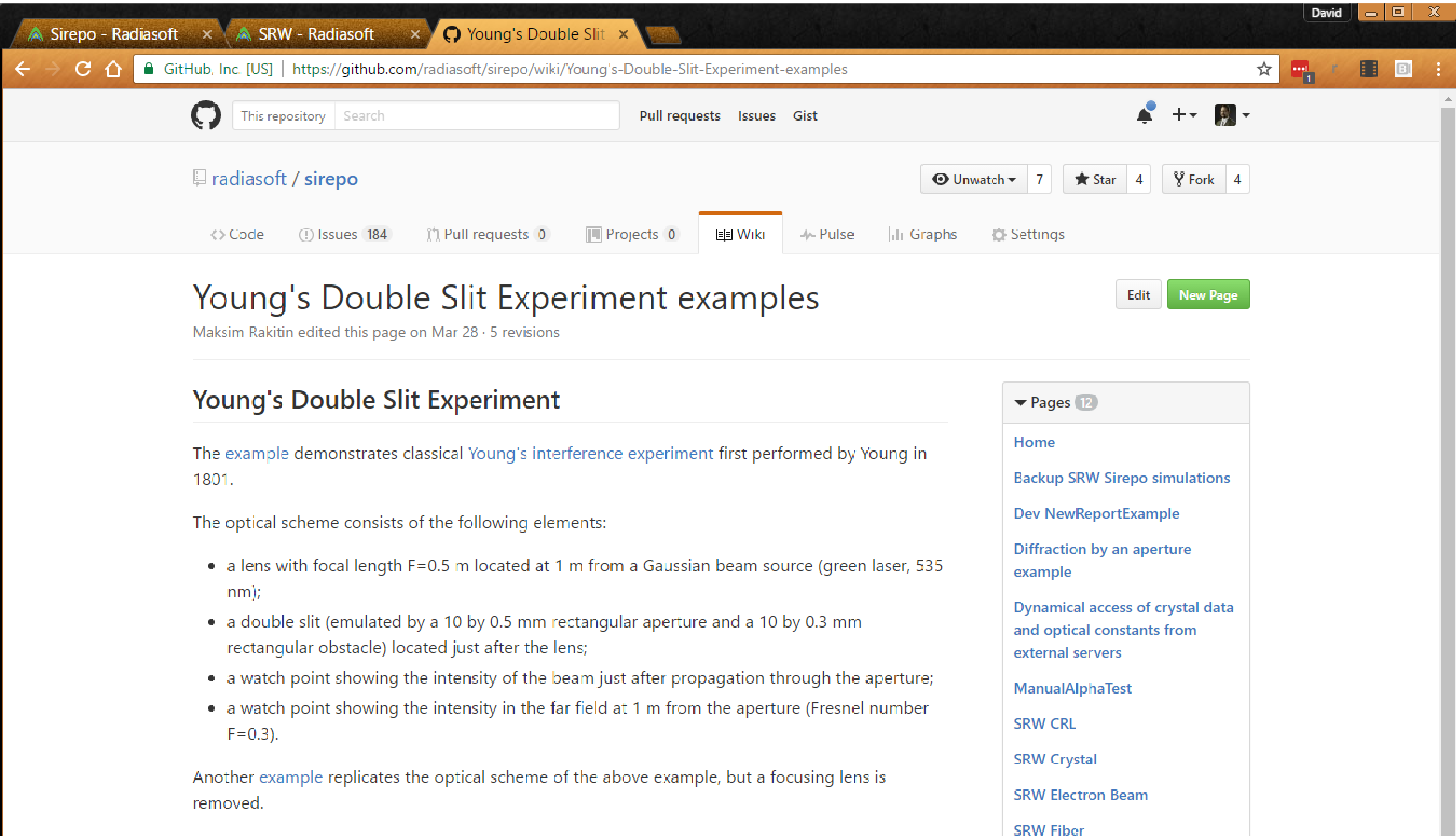
- Run SRW on your own cluster via this script
- Make edits
- Share it with collaborators
- If the structure is preserved, it can be imported back to the GUI

```
#!/usr/bin/env python
try:
    __IPYTHON__
    import sys
    del sys.argv[1:]
except:
    pass


import srwl_bl
import srwlib
import srwlpy


def set_optics(v=None):
    el = []
    el.append(srwlib.SRWLOptA("r", "a", 0.0015, 0.0015, 0.0, 0.0))
    el.append(srwlib.SRWLOptD(22.0))
    el.append(srwlib.SRWLOptMirSph(_r=1410.0, _size_tang=0.5, _size_sag=0.11, _nvx=0.999902524484, _nvy=0.0, _nvz=-0.0139621463265,
_tvx=0.0139621463265, _tvy=0.0))
    el.append(srwlib.SRWLOptD(13.0))
    el.append(srwlib.SRWLOptA("r", "a", 3e-05, 0.01, 0.0, 0.0))
    el.append(srwlib.SRWLOptD(11.0))
    el.append(srwlib.SRWLOptA("r", "a", 0.00125, 0.00125, 0.0, 0.0))
    el.append(srwlib.SRWLOptMirEl(_p=64.75, _q=1.0, _ang_graz=0.0025, _size_tang=0.5, _size_sag=1e+40, _nvx=0.0, _nvy=0.999996875002,
_nvz=-0.00249999739583, _tvx=0.0, _tvy=-0.00249999739583))
    el.append(srwlib.SRWLOptD(0.5))
    el.append(srwlib.SRWLOptMirEl(_p=11.5, _q=0.5, _ang_graz=0.0025, _size_tang=0.5, _size_sag=1e+40, _nvx=0.999996875002, _nvy=0.0,
_nvz=-0.00249999739583, _tvx=-0.00249999739583, _tvy=0.0))
    el.append(srwlib.SRWLOptD(0.5))

    pp = []
    pp.append([0, 0, 1.0, 0, 0, 4.0, 4.0, 3.0, 3.0])

    pp.append([0, 0, 1.0, 1, 0, 1.0, 1.0, 1.0, 1.0])
    pp.append([0, 0, 1.0, 0, 0, 1.0, 1.0, 1.0, 1.0])

    pp.append([0, 0, 1.0, 1, 0, 1.0, 1.0, 1.0, 1.0])
    pp.append([0, 0, 1.0, 0, 0, 1.0, 6.0, 1.0, 1.0])
```

You can export and import JSON files

– enables sharing

Open source: https://github.com/radiasoft



Acknowledgments

Many helpful discussions with and feedback from the following scientists:

- Gerard Andonian
- Alexey Buzmakov
- Carsten Fortmann-Grote
- Mark Glass
- Jacek Krzywinski
- Luca Rebuffi
- Ruben Reininger
- Liubov Samoylova
- Manuel Sanchez del Rio
- Timur Shaftan
- Xianbo Shi
- Lin Zhang

Please try our beta implementation of SRW in the cloud: https://beta.sirepo.com
Please send all feedback to David Bruhwiler bruhwiler@radiasoft.net
YouTube videos can be found here: http://radiasoft.net/youtube

This work is supported by the US DOE Office of Science, Office of Basic Energy Sciences under Award # DE-SC0011237.