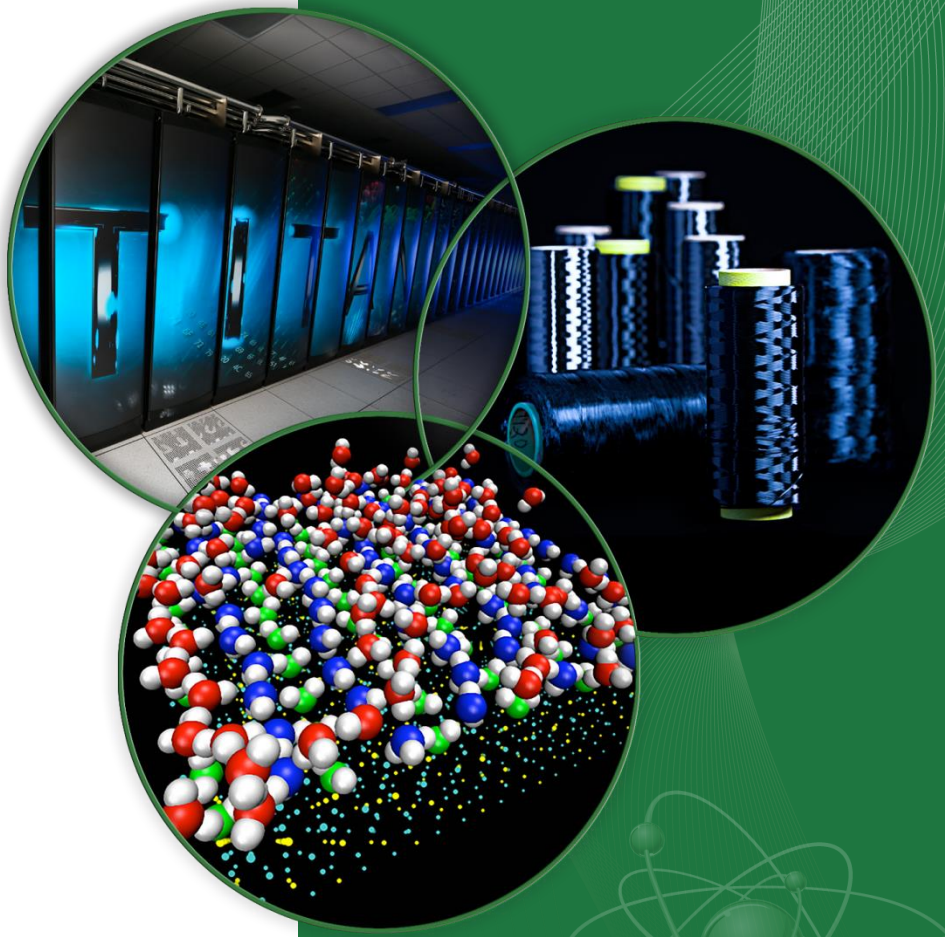
Control System Studio: BOY Details



Kay Kasemir

ORNL/SNS

kasemirk@ornl.gov

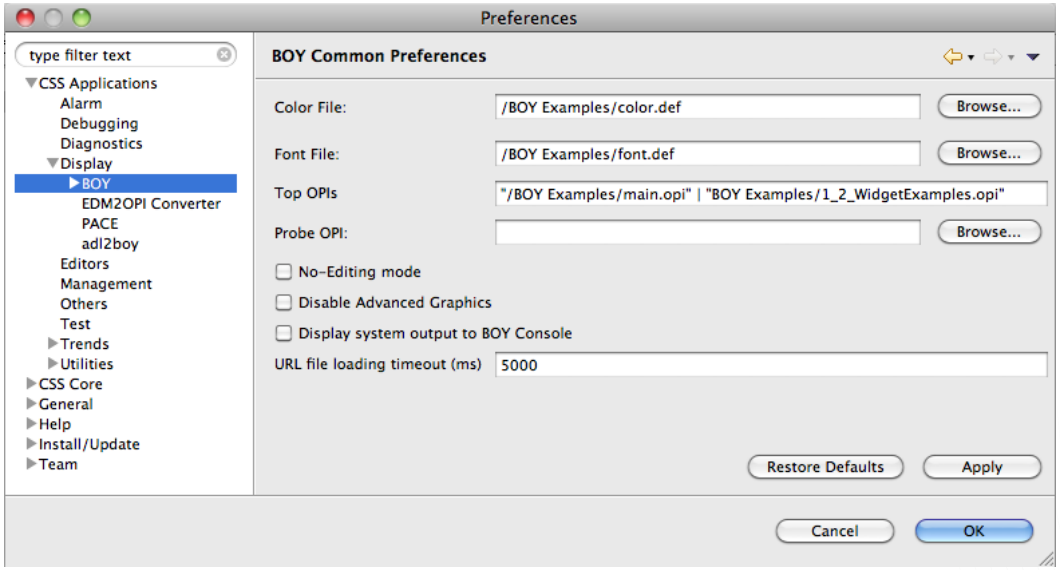
A lot of material from Nadine Utzel, ITER and BOY online help by Xihui Chen, SNS

June 2014

BOY Font, Color Preferences

Menu *CSS*, *Preferences*:

- Locate the BOY settings
- Check *Color File*, *Font File*, *Top OPIs*:
  Are they set to use files from the BOY Examples?
- Open associated files in text editor



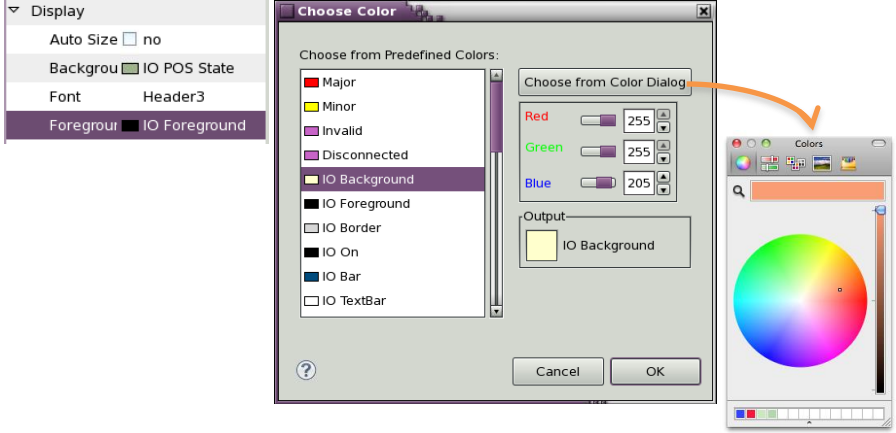
Font, Color Names

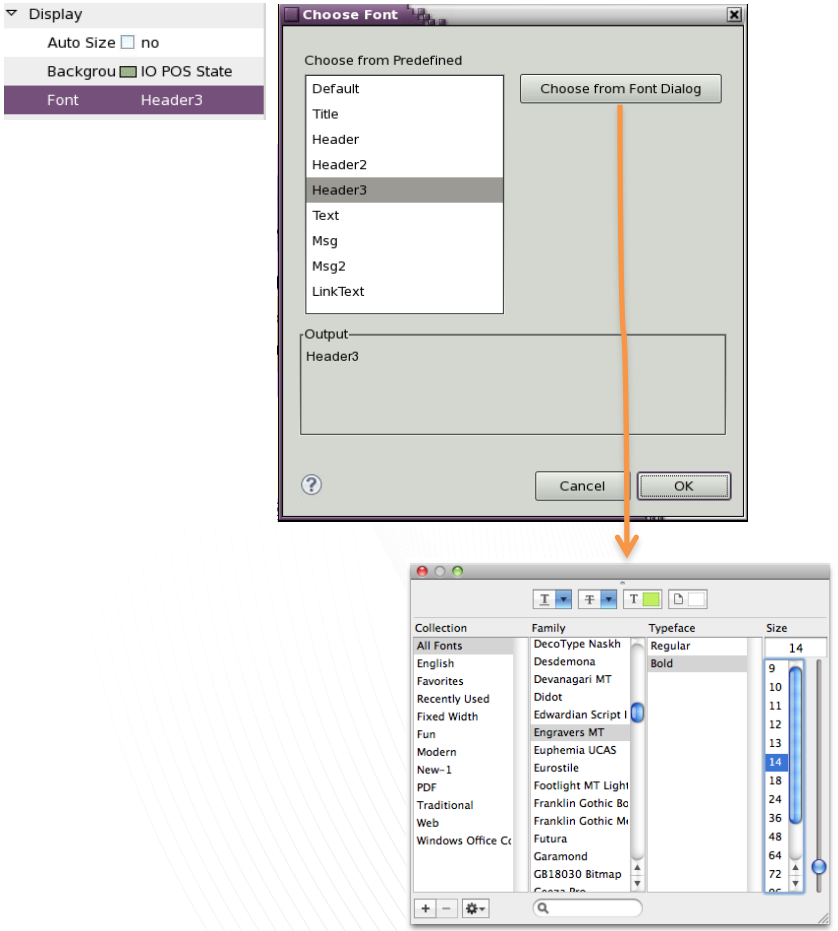
When configuring a color (foreground, background, border, …) or font (Text Update font, …), you have two options:

a) Pick any color or font
   - RGB resp. Name, Typeface, Size

b) Pick a Predefined Color resp. Font

What is better? Why?





Exercise: Use Predefined Fonts

- Add a Label to your display
  - Set font to the predefined Title font
  - Set text to something like “This is the Title”
- Add another Label
  - Assert that it uses the “Default” font

Portable Usage of Fonts

Fonts differ between operating systems: “Times New” vs. “adobe-times-..” etc.

How can an OPI file “Look the same” on Windows, OS X, Linux?

1. If possible, install the same fonts on all your computers
   - Microsoft “Office” fonts available on most Windows and Mac OS computers because they also run MS Office
   - MS Office fonts are also available for Linux! Google “free office fonts Linux”
2. BOY fonts.def file allows system-specific tweaks

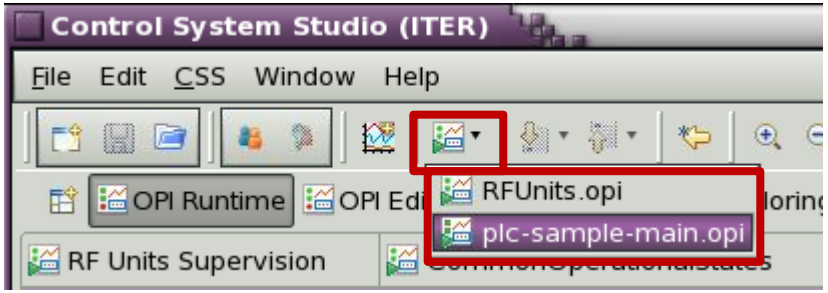
```
# Though using the same MS Office font
# on all operating systems, the sizes seem
# somewhat different.
# Fix that by using different sizes for
# each OS:
Default=Verdana-regular-10
Default(macosx_cocoa)=Verdana-regular-14
Default(linux_gtk)=Verdana-regular-10

# Same with “Header1”: OS X needs bigger font
# for same on-screen pixel size
Header1=Verdana-bold-24
Header1(macosx_cocoa)=Verdana-bold-36
```

Exercise: Schema File

- Create a new display file “schema.opi”
  - Add a Text Update
    - Background Color: Yellow
    - Foreground Color: Red
  - Save, close the schema.opi
- Menu CSS, Preferences, CSS Applications, Display, BOY, OPI Editor
  - Set the “Schema OPI” to the schema.opi that you just created
- Create a new OPI file
  - Add a Text Update widget
  - Notice its initial Background & Foreground color?

Preferences: Top OPIs, Site wide settings

- Top OPIs: Appear in Toolbar



- Path names for color & font files, “Top” OPIs, Schema can be web links
  - Instead of /BOY Examples/font.def use http://some.server.org/path/font.def

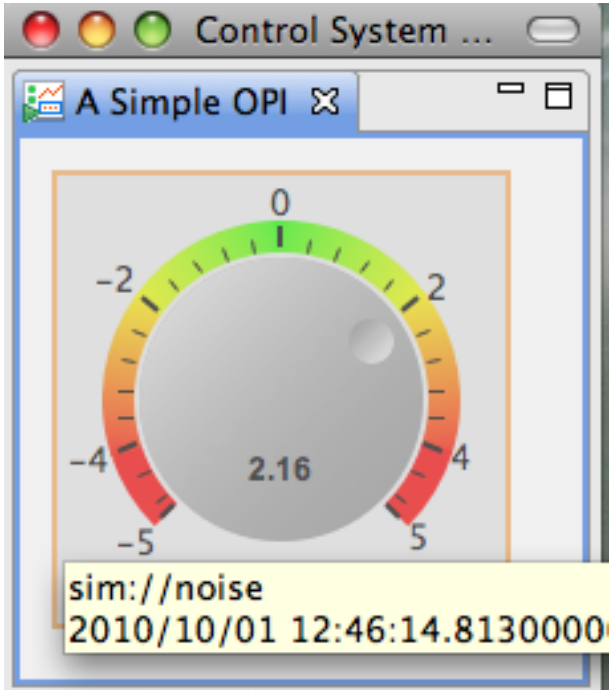
Good for site-wide files like your top-level control system screen!

Suggestions for your site

- After gaining some experience with BOY, somebody with design talents defines which colors, fonts, … to use for displays at your site
- Pick fonts that look similar on all operating systems
- Create color.def, font.def, schema.opi
  - Place these on a web server
  - Configure CSS for your site to use the http://… paths to the *.def and schema.opi
- You can put your *.opi files into CVS
  - or subversion, Mercurial, GIT, …CSS can include support for these
- Each night, you can publish the current *.opi files from CVS on your web server
  - Point the “Topi OPIs” to http://web.server/opis/main.opi
  - End users can now easily run the “current” version from the Toolbar

Main Idea: Simple Things are Easy

1. Drag a widget, e.g. Knob, from palette to editor
2. Enter the PV name in Properties view
3. Click the "Run" button to execute!



There is more, but don't go overboard!

Keep logic on the IOC.

Display is only for the display.

Don't implement whole application in BOY.

Widgets and Properties Galore

- Compared to EDM, MEDM, … BOY tries to offer specialized widgets
  - Grouping Container instead of Lines
  - LED instead of Circle-with-color-rule
  - Image Button instead of Images with conditional visibility in front of invisible button
  - Tabbed Container instead of embedded window, many invisible buttons, conditionally visible graphics, local PVs to update the display inside the embedded window
- .. with many Properties
  - Alarm sensitive Border/Background/…
  - Precision, Limits, … from PV or direct entry
  - Actions

Widgets and Properties Galore because..

Display file describes Meaning:

LED to display something, not Circle that happens to change color.

Group of related widgets, not rectangle that happens to surround something.

Border color to reflect alarm state, not arbitrary change in color.

Font name “Title”, not “Arial-bold-12”.

Displays with same Representation (Lines, circles with changing color, “Arial-bold-12”) look the same as displays with Meaning (group, LED, Title).

But they are like compiled binaries without source code. Less useful in the long run.

In the future, files with Meaning will be easier to translate for other, new tools than files with only Representation.

User Interfaces..

.. are the visible, attractive part of the control system.

.. are just that. Logic belongs onto the IOC.

.. come and go. Don’t get too tricky with the current one.

Disclaimer: Rules & Scripts

… can change any property of any widget:

- Change text of label based on a PV
  - i.e. build your own Text Update
- Change color of an Ellipse based on PV
  - i.e. build your own LED

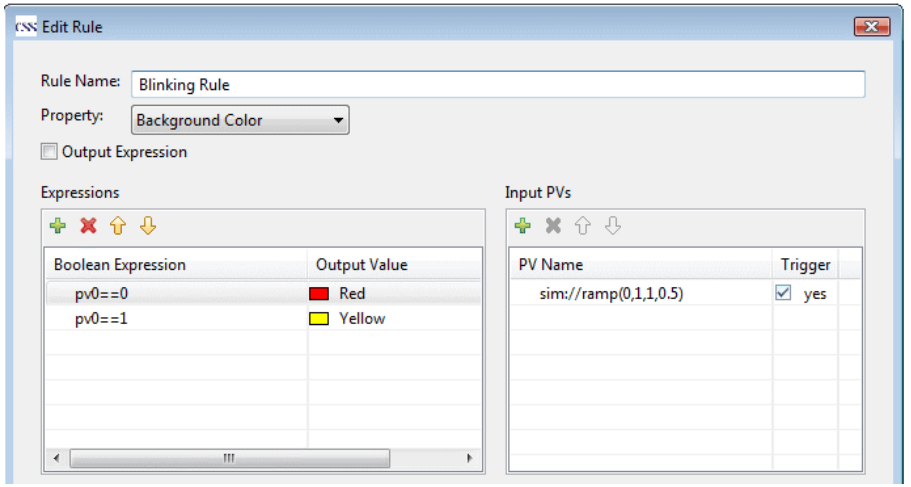
Based on last slide, that is a bad idea!

Still, there are places where rules and scripts can be very powerful.

Rules, Scripts

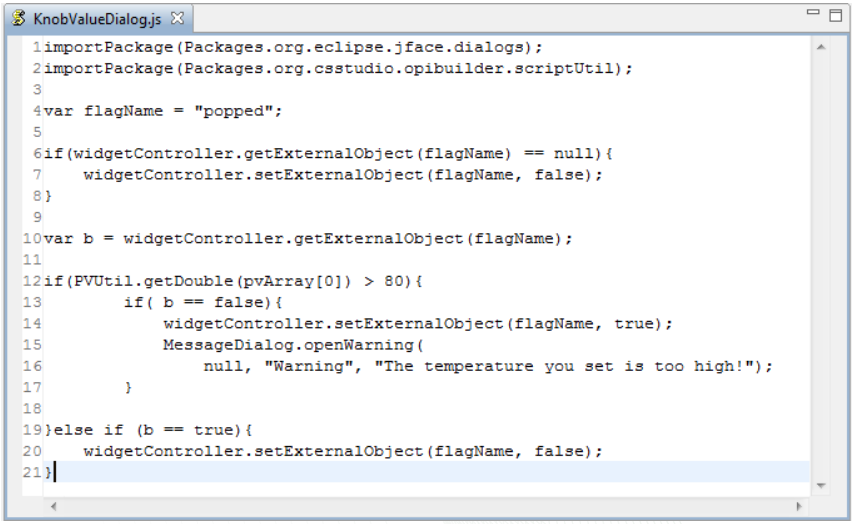
Rules create dynamic displays

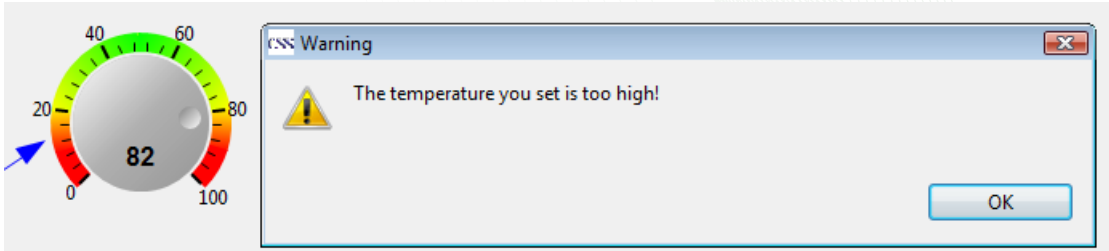
- Easy: PV → Widget Property



Scripts can to "anything"

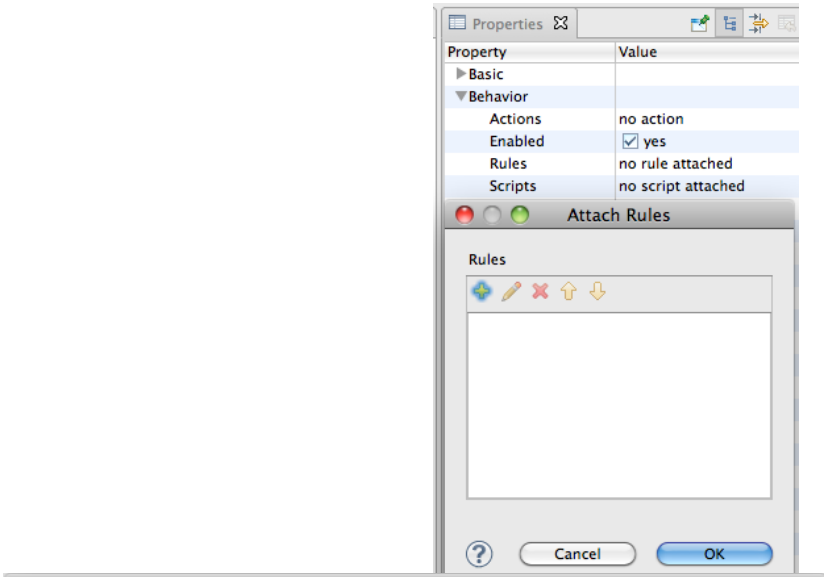
- Read PVs,
  change widget properties,
  open dialog, …
- JavaScript or Python (Jython)

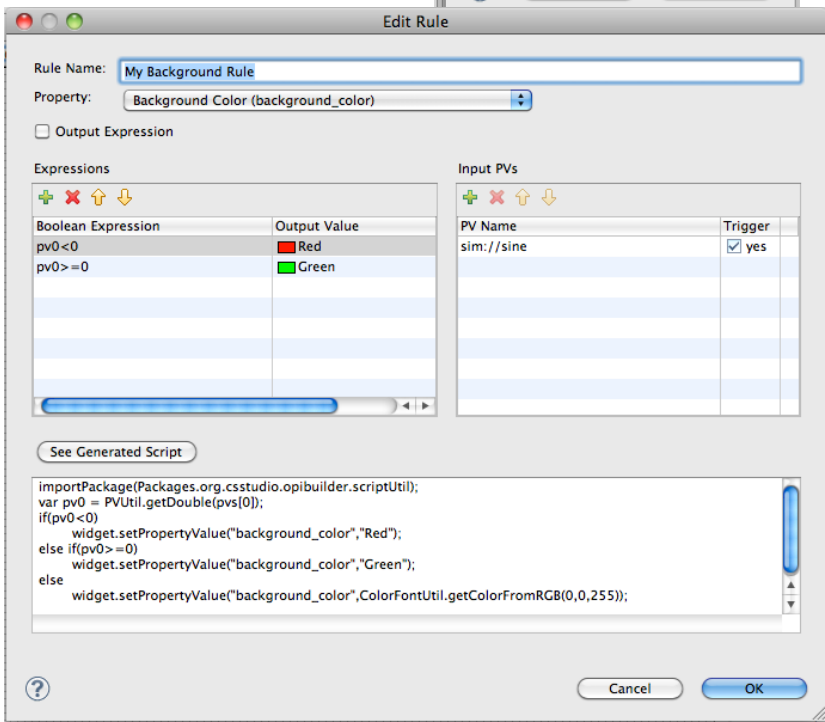




Exercise: Rule to change color of Ellipse

- Create *Ellipse* widget
- Locate its *Behavior*, *Rules* Property
- Click the “no rule attached” value to open the dialog to Attach (or edit) Rules
- Add a rule that changes the background color as shown between Red and Green, triggered by changes in the sim://sine PV
- Press “See Generated Script”, compare with screenshot
- Maybe add another TextUpdate widget to display the same sim://sine PV
- Run the display





Rules vs. Scripts

Rules

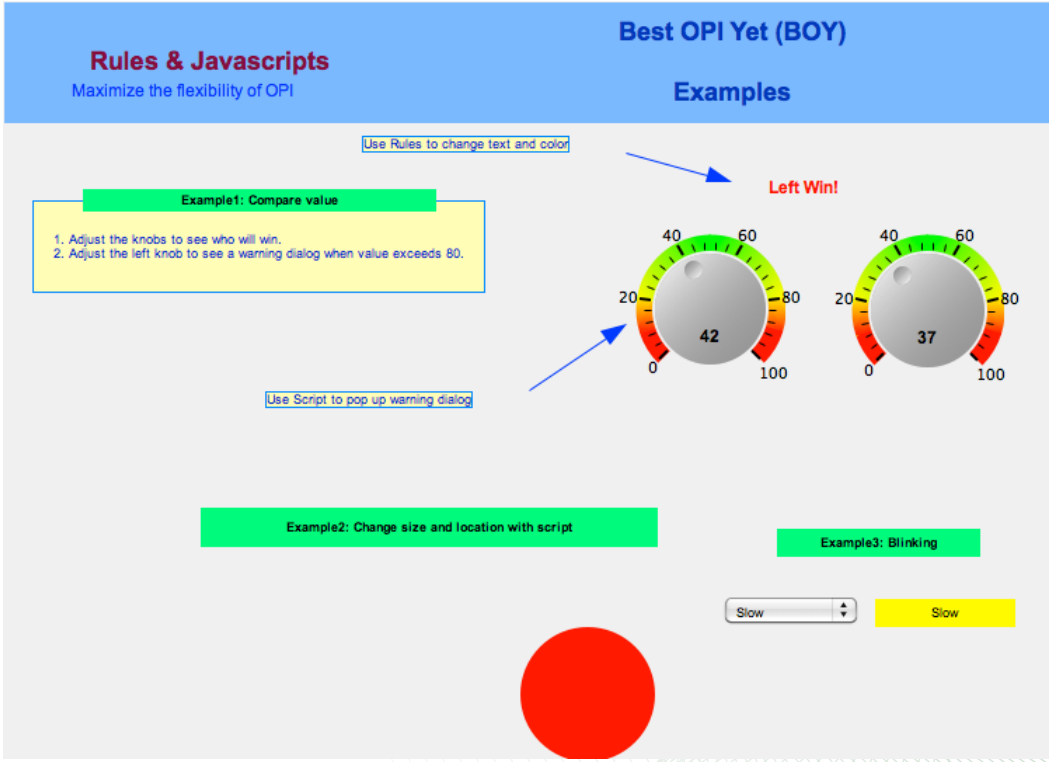
- are simpler: One or more PVs change one property
- are closer to describing Meaning
- are internally converted to scripts, but what's saved in the *.opi file is the Meaning: Property to adjust, expressions for rule, input PVs
- should be preferred to scripts whenever possible

Scripts

- can be pretty much any Java Script of Jython code
- can affect multiple properties, widgets, even add and remove widgets
- should be used with care, because they can be hard to maintain in the long run
  - Use org.cstudio.opibuilder.scriptUtil (PVUtil, ColorFontUtil)
  - Add many source code comments

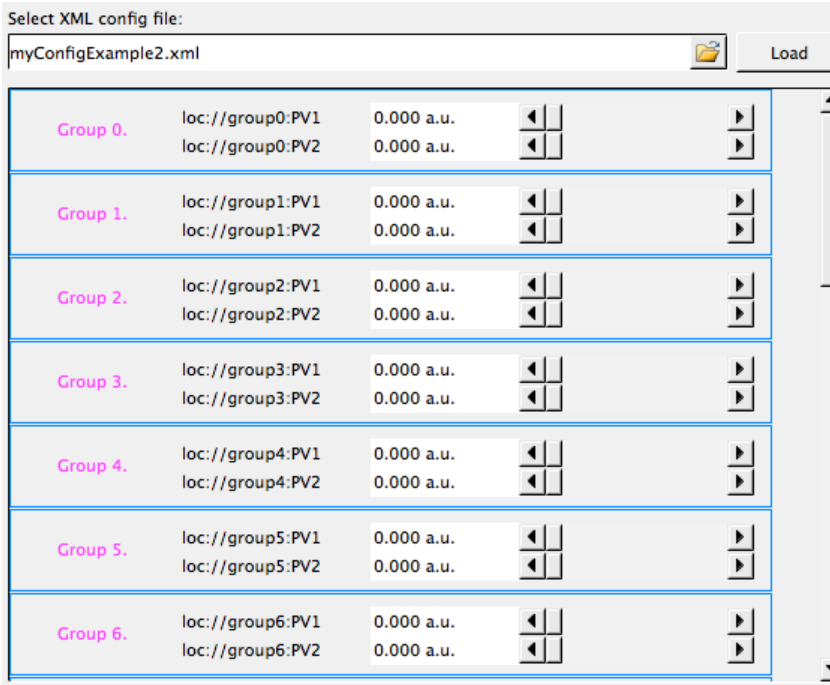
Rules, Scripts in OPI Examples

- Open BOY Examples/5_3_Rules_Script.opi, first in Runtime, then in Edit mode
- Check the rules behind the “Left Win!” text above the two knobs
- Check the Script attached to the left Knob
- Check the Script attached to the moving circle
  - How does it change its color?



Script-generated Displays

- Open
  BOY Examples/Miscellaneous/DynamicLoadWidgets/LoadWidgetsExample.opi
  in Runtime mode



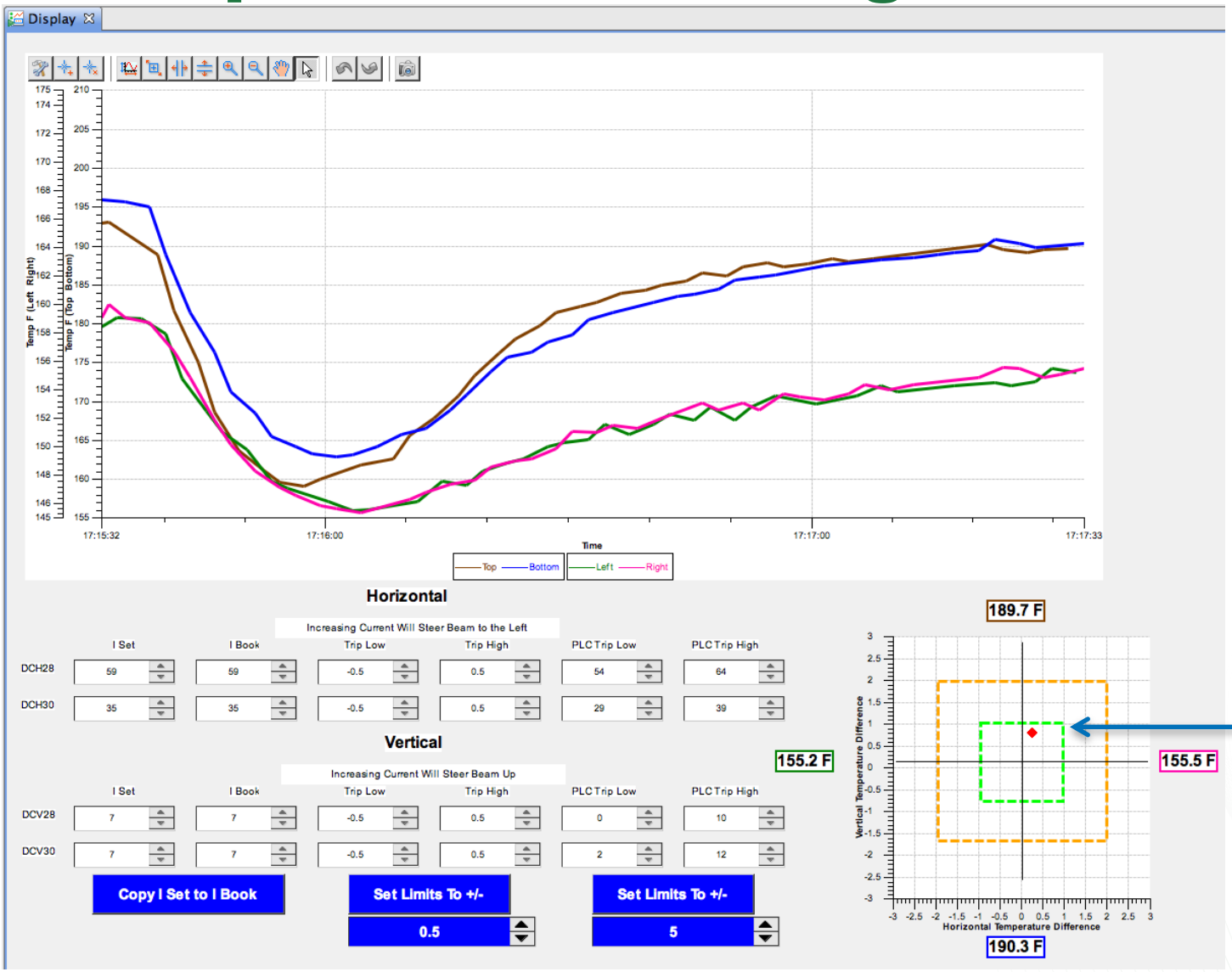
- Enter “myConfigExample.xml”, press “Load”.
  Enter “myConfigExample2.xml”, press “Load”.
  - Notice a difference?

- Open SubPanel.opi in Edit mode,
  change it slightly by setting the color
  of the “Group…” label to violet,
  save, then press “Load”
  on LoadWidgetsExample.opi
  - See how it’s using the current version
    of SubPanel.opi?

Investigate how this is done!

- What PV is attached to the text field
  where you enter the *.xml file names?
- What PV is attached to the “Load” button?
- Note the script attached to the big Grouping Container that appears empty in edit mode, but is dynamically populated with copies of SubPanel.opi in runtime mode.
- Read that script together with myConfigExample.xml.
  Writing such a script requires knowledge of the BOY widget model.
  You don’t have to write such a script, but you should be able to understand what it does.

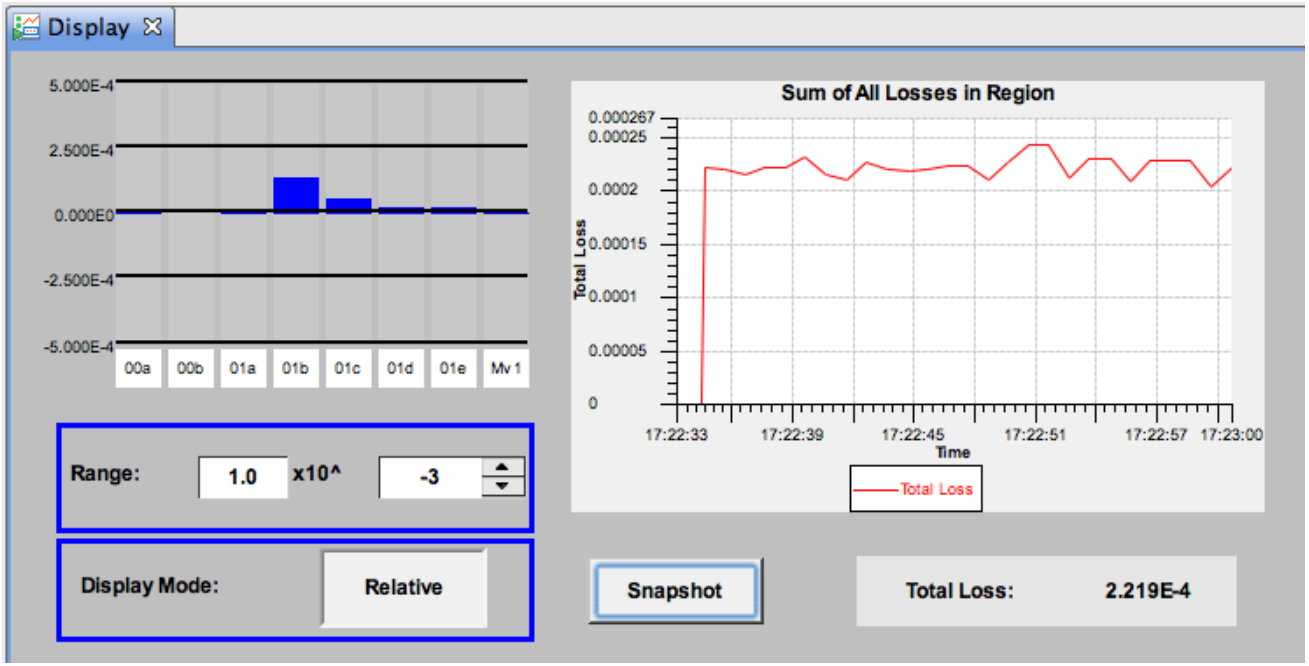
Example: SNS "Steering" Tool



Try to get spot into the green, at least into orange

Tim Southern, Nick Luciano

Scripts can replace custom Applications!



Display how beam loss is increased or reduced relative to a “snapshot”

SNS operation group:
Tim Southern, Nick Luciano

Good

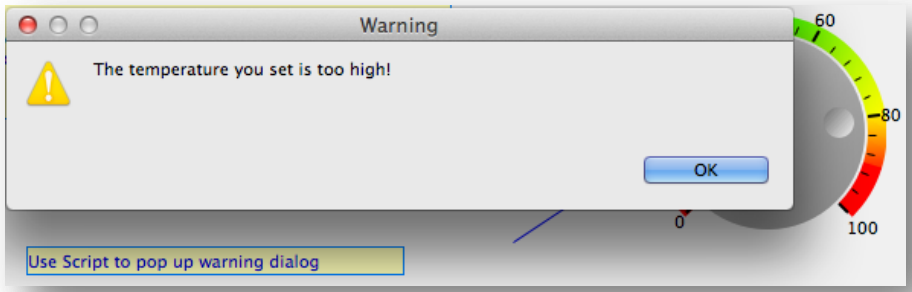
- Add Widget, enter PV, done
- TextUpdate widget with enum, string PV
- LED widget with PV
- Rule used once to highlight a special state.
  Otherwise, update or create new widget.

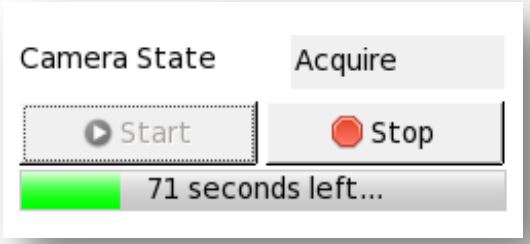
Bad

- Add 5 widgets, use dynamic visibility, local PVs, scripts
- Various overlapping Text widgets with rule to change visibility
- Circle widget, filled with rule-based color
- Rule attached to every widget to display alarm severity, because you don’t like the alarm sensitive border

What not to do in scripts

- Check allowed values
  - Record's DRVH, DRVL
- Perform interlocks
  - CALC records
- "Timer" Displays
  - CALC ..
- Start threads to "Ramp Power Supply Voltage"
  - CALC records, Sequencer
- Write experiment data to file
  - Archive tool, area detector, SCAN record, ..

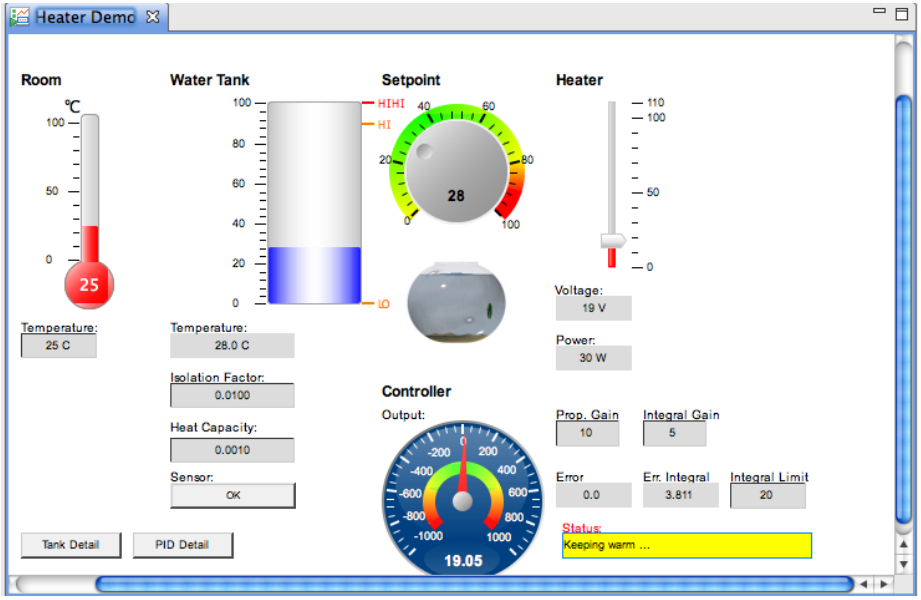




Summary

There is a lot you *can* do in BOY

- Macros, Rules, Scripts, …



Remember
the Main Idea:

Simply Things are Easy

1. Add widget
2. Enter PV Name
3. Run