



Unifying MSc Physics and MSc Astrophysics Problem-Based Learning with LabVIEW NXG: A Critical Review

NI Academic Users Forum 2019

Dr Richard James Lewis

Cardiff University School of Physics and Astronomy (PHYSX)



Overview

Plan: transitioning core modules to NXG 2.1

- Summary of MSc activity at Cardiff PHYSX
- Core MSc module structure: then and now

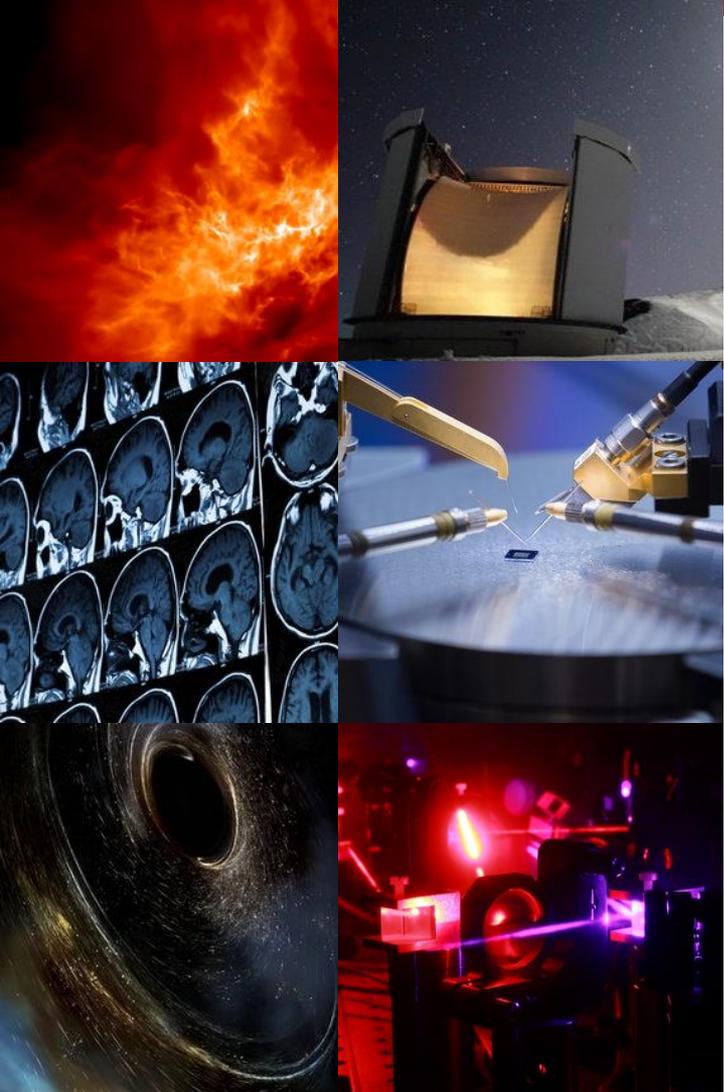
Execution: learning and application

- Impact of NXG on LabVIEW language learning
- Impact of NXG on problem-based learning
- Logistical issues and their solutions

Outcome

- Student reception of new core module
- Future MSc core module design
- Summary of LabVIEW NXG 2.1 critique

Plan: transitioning core modules to NXG 2.1



Summary of MSc activity at Cardiff PHYSX

Programmes

- MSc Physics
- MSc Data-Intensive Physics
- MSc Compound Semiconductor Physics
- MSc Astrophysics
- MSc Data-Intensive Astrophysics
- MSc Gravitational Wave Physics (2019/20)

Cohort of 2018/19

- 24 students (11 Physicists, 13 Astrophysicists)
- 2:1 minimum entrance requirement
- Dedicated co-located MSc teaching facilities
- 20cr common core module with LabVIEW NXG

Transitioning core modules to LabVIEW NXG

Major restructuring for 2018/19

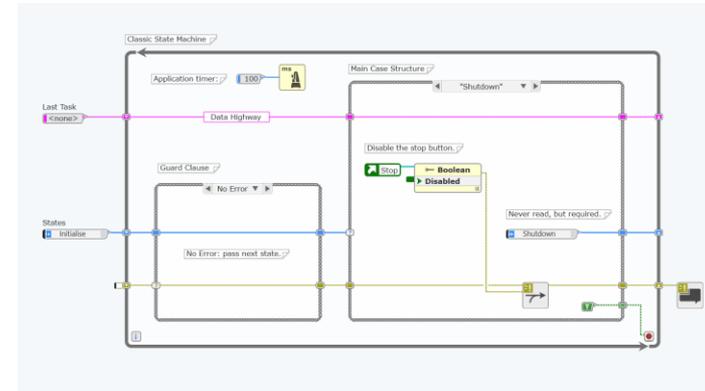
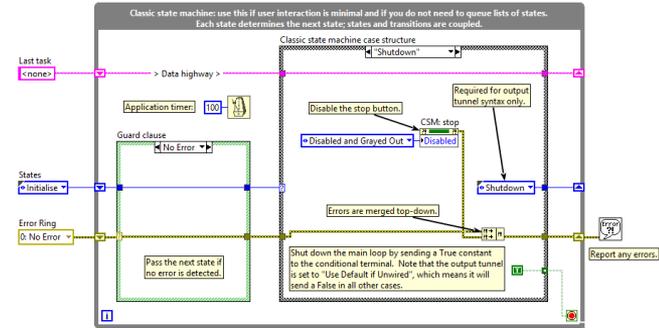
- Merger of core autumn physics and astrophysics modules
- MSc Physics pattern adopted for this 20cr module:
 - Practical introduction to LabVIEW (10cr)
 - Student-lead micro-projects (10cr)
- Transition from LabVIEW 2015 to LabVIEW NXG 2.1*

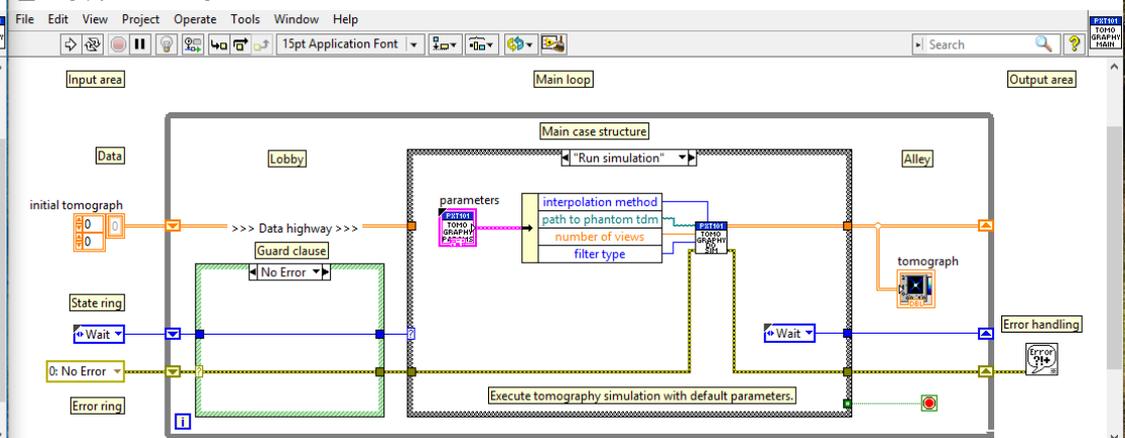
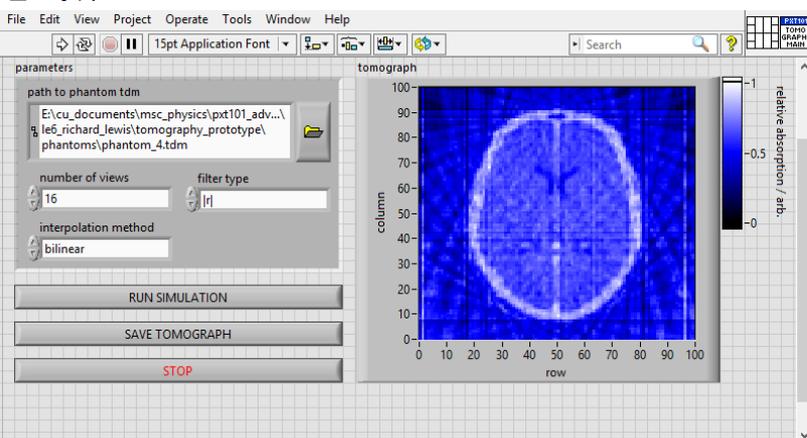
Why LabVIEW NXG?

- **Pedagogy:** gentler, more logical learning curve
- **Logistics:** useful from day 1, therefore always relevant

What do we want to maintain / enhance?

- Students feel valued as part of our academic community
- Exceptionally high student feedback score average of **93**





Old structure (LabVIEW 2015): drills to application

Week 1: projects, front panel, block diagram, dataflow, Express VIs, AAP

Week 2: arrays, clusters, file I/O, case structures, loops, sub VIs, errors, style

Week 3: functional specifications, efficient VI engineering

Week 4: interfacing with hardware, MAX, VISA, AAP with hardware

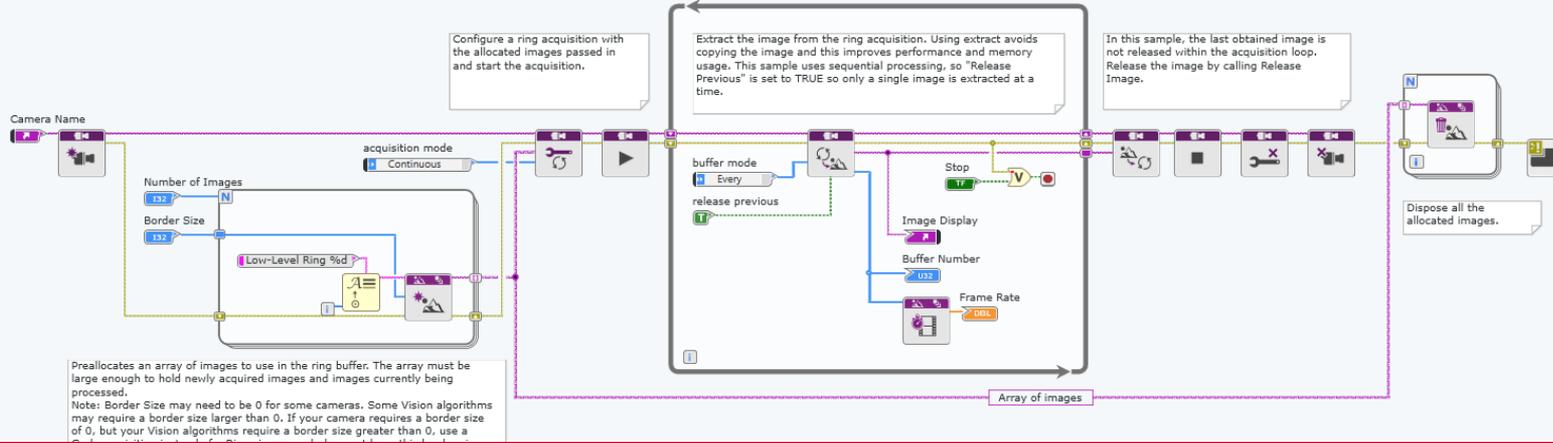
Week 5: development paradigms, type definitions, DAQmx

Week 6: error handling, tunnels, shift registers, classic state machines

Week 7: event structure, event-driven state machines, functional global variables

Week 8: queues, queued state machines

Weeks 9 to 11: applying LabVIEW to micro-projects



Planned (LabVIEW NXG 2.0): DAQ-first, applications throughout

Week 1: a guided tour of LabVIEW NXG, establishing common and linked contexts

Week 2 (AAV and IO I): no coding required, NXG functionality, DAQ on 1 and 2 channels

Week 3 (AAV and IO II): coding drag and drop, DAQ on 1 and 2 channels, images

Week 4 (AAV and IO III): coding from scratch, DAQ on 1 and 2 channels, images

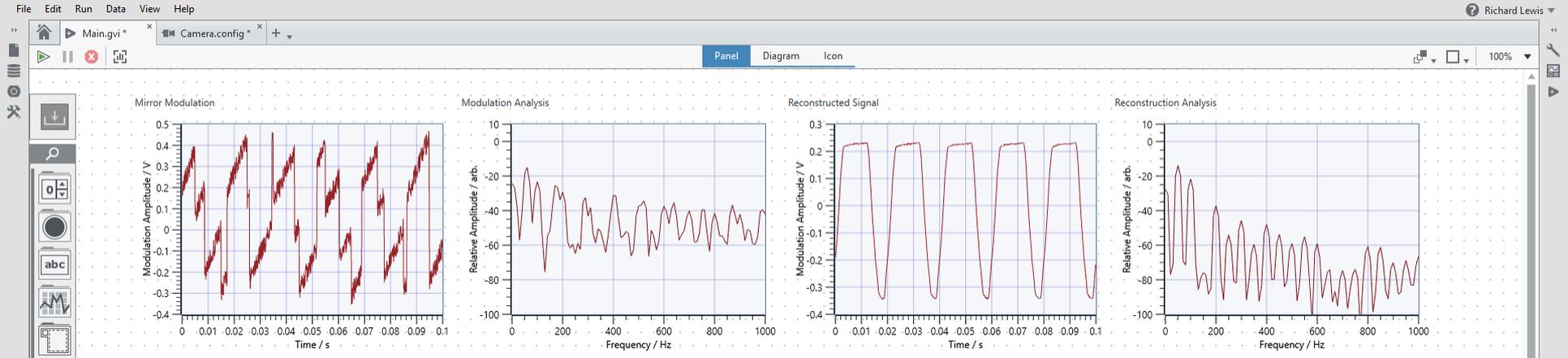
Week 5 (looping and iteration I): looping and iteration, looping AAV code

Week 6 (looping and iteration II): classic state machines

Week 7 (looping and iteration III): event-driven state machines

Week 8 (looping and iteration IV): queued state machines

Weeks 9 to 11: continuing to apply LabVIEW to micro-projects



Result (LabVIEW NXG 2.1): DAQ-first, applications throughout

Week 1: a guided tour of LabVIEW NXG, establishing common and linked contexts

Week 2 (AAV and IO I): no coding required, NXG functionality, DAQ on 1 and 2 channels

Week 3 (AAV and IO II): coding drag and drop, DAQ on 1 and 2 channels, images*

Week 4 (AAV and IO III): coding from scratch, DAQ on 1 and 2 channels, images*

Week 5: module recess

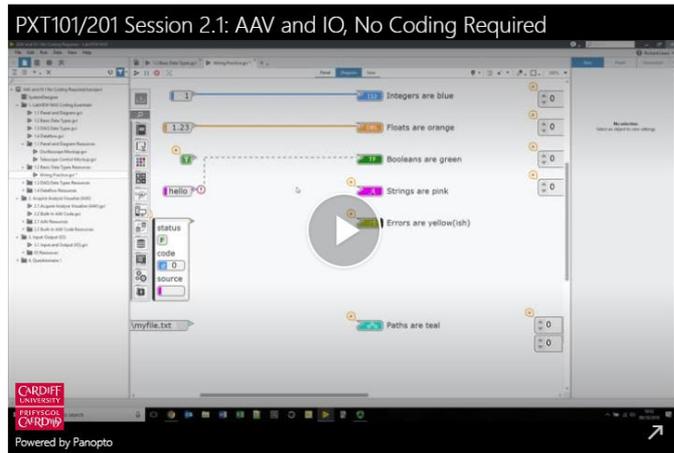
Week 6 (looping and iteration I): looping AAV code and classic state machines

Week 7 (looping and iteration II): classic and event-driven state machines

Week 8 (looping and iteration III): : event-driven and queued state machines

Weeks 9 to 11: continuing to apply LabVIEW to micro-projects

Execution: learning and application
Impact of NXG on language learning

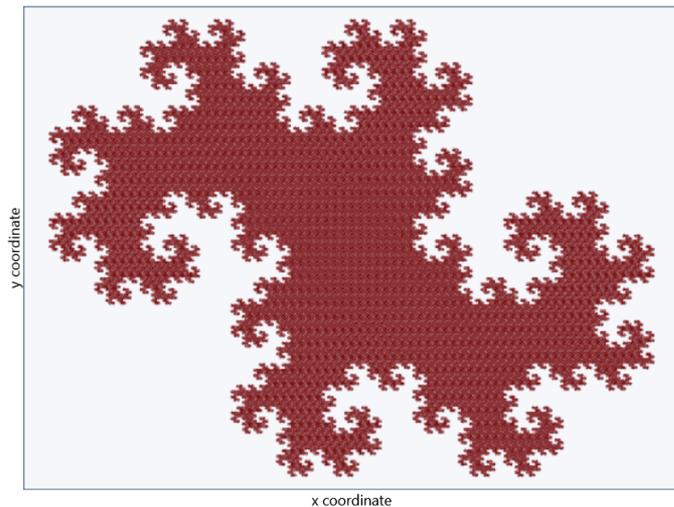


Impact of NXG on language learning

Summary of targets for NXG course

- Everything in projects (no slides)
- Every session recorded on Panopto
- No (nasty) surprises, anticipate student issues
- Physics **and** Astrophysics examples throughout

Dragon Curve



Lots to talk about, so have to be selective:

- Experience of keeping everything in projects
- NXG interface advantages and issues
- (Somewhat) nasty surprises, workarounds

The screenshot displays the LabVIEW NXG environment. On the left, a project tree shows a 'Timetable.gvi' file. The main workspace contains a text area with instructions: 'Run the GVI (green arrow in the toolbar) to load, process, and display the timetable for 2018/19.' Below this are three tables:

PXT101/201 contact sessions

03/10/2018 Week 1: A Guided Tour of LabVIEW NXG
10/10/2018 Week 2: AAV and IO I: No Coding Required
17/10/2018 Week 3: AAV and IO II: Coding Drag-and-Drop
24/10/2018 Week 4: AAV and IO III: Coding from Scratch
31/10/2018 Week 5: *** MODULE RECESS ***
07/11/2018 Week 6: Looping AAV Code
14/11/2018 Week 7: Classic State Machines and their Applications
21/11/2018 Week 8: Event-Driven State Machines and their Applications
28/11/2018 Week 9: Queued State Machines and their Applications
05/12/2018 Week 10: Micro Project Clinic
12/12/2018 Week 11: Micro Project Presentations

NXG assignments out (left) / in (right)

LE1	
LE2	LE1
LE3 (Group)	LE2
LE4 (Group)	LE3 (Group)

uP assignments out (left) / in (right)

LD1 (Formative)	
LD2	LD1 (Formative)
LD3	LD2
LD4	LD3
LD5	LD4
LD6	LD5
LD7	LD6
	uPProject Report

On the right, a block diagram shows the data flow between these tables and other NXG components like 'Format the timetable strings' and 'Display the extract of strings'.

Experience of keeping everything in projects

- Dramatically simplifies distribution logistics, projects enforced, real directories
- Full integration of session questionnaires, exercise scripts, and their solutions
- Live streaming of session capture allows students to review during hands-on
- Sessions much more dynamic than for 2017/18, with far less start-and-stop
- Formatting tools in NXG for free text comments are very simplistic
- Workbooks are much more sophisticated, but not available to end user

Guided Tour of LabVIEW NXG 2 - LabVIEW NXG

File Edit Run Data View Help

1.1 PXT101 201 Timetable.gvi

Panel Diagram Icon

125%

Filter

- A Guided Tour of LabVIEW NXG 2.lvproject
 - SystemDesigner
 - 1. A Guided Tour of LabVIEW NXG 2
 - 1.1 PXT101 201 Timetable.gvi
 - 1.2 Tomography Simulation.gvi
 - 1.3 Analyse Variable Star Light Curve.gvi
 - 1.4 Example Projects and External Resourc...
 - 1.5 Sandbox.gvi
 - 1.1 Timetable Resources
 - 1.2 Tomography Resources
 - Add border to 2D array.gvi
 - Back projection.gvi
 - Filter back projection.gvi
 - Generate phantom image.gvi
 - Generate rotation angles array.gvi
 - Path to phantom.gvi
 - G Types (Tomography)
 - Phantoms
 - 1.3 Variable Star Resources
 - Average JD-Magnitude Array (Day).gvi
 - FFT of Light Curve.gvi
 - Find Fundamental Period.gvi
 - Interpolate to Equally-Spaced JD-Magn...
 - Variable Star Relative Path.gvi
 - G Types (Variable Star)
 - Variable Star Data
- 2. LabVIEW NXG Coding Essentials
 - 2.1 Panel and Diagram.gvi
 - 2.2 Basic Data Types.gvi
 - 2.3 DAQ Data Types.gvi
 - 2.4 Dataflow.gvi
 - 2.5 Acquire Analyse Visualise (AAV).gvi
 - 2.5 AAV Task Definitions
- 3. Questionnaire

1.1 PXT101/201 Timetable.

Run the GVI (green arrow in the toolbar) to load, process, and display the timetable for 2018/19.

PXT101/201 contact sessions

03/10/2018 Week 1: A Guided Tour of LabVIEW NXG
10/10/2018 Week 2: AAV and IO I: No Coding Required
17/10/2018 Week 3: AAV and IO II: Coding Drag-and-Drop
24/10/2018 Week 4: AAV and IO III: Coding from Scratch
31/10/2018 Week 5: *** MODULE RECESS ***
07/11/2018 Week 6: Looping AAV Code
14/11/2018 Week 7: Classic State Machines and their Applications
21/11/2018 Week 8: Event-Driven State Machines and their Applications
28/11/2018 Week 9: Queued State Machines and their Applications
05/12/2018 Week 10: Micro Project Clinic
12/12/2018 Week 11: Micro Project Presentations

NXG assignments out (left) / in (right)

LE1	
LE2	LE1
LE3 (Group)	LE2
LE4 (Group)	LE3 (Group)
	LE4 (Group)

uP assignments out (left) / in (right)

LD1 (Formative)	
LD2	LD1(Formative)
LD3	LD2
LD4	LD3
LD5	LD4
LD6	LD5
LD7	LD6
	LD7
	uProject Report

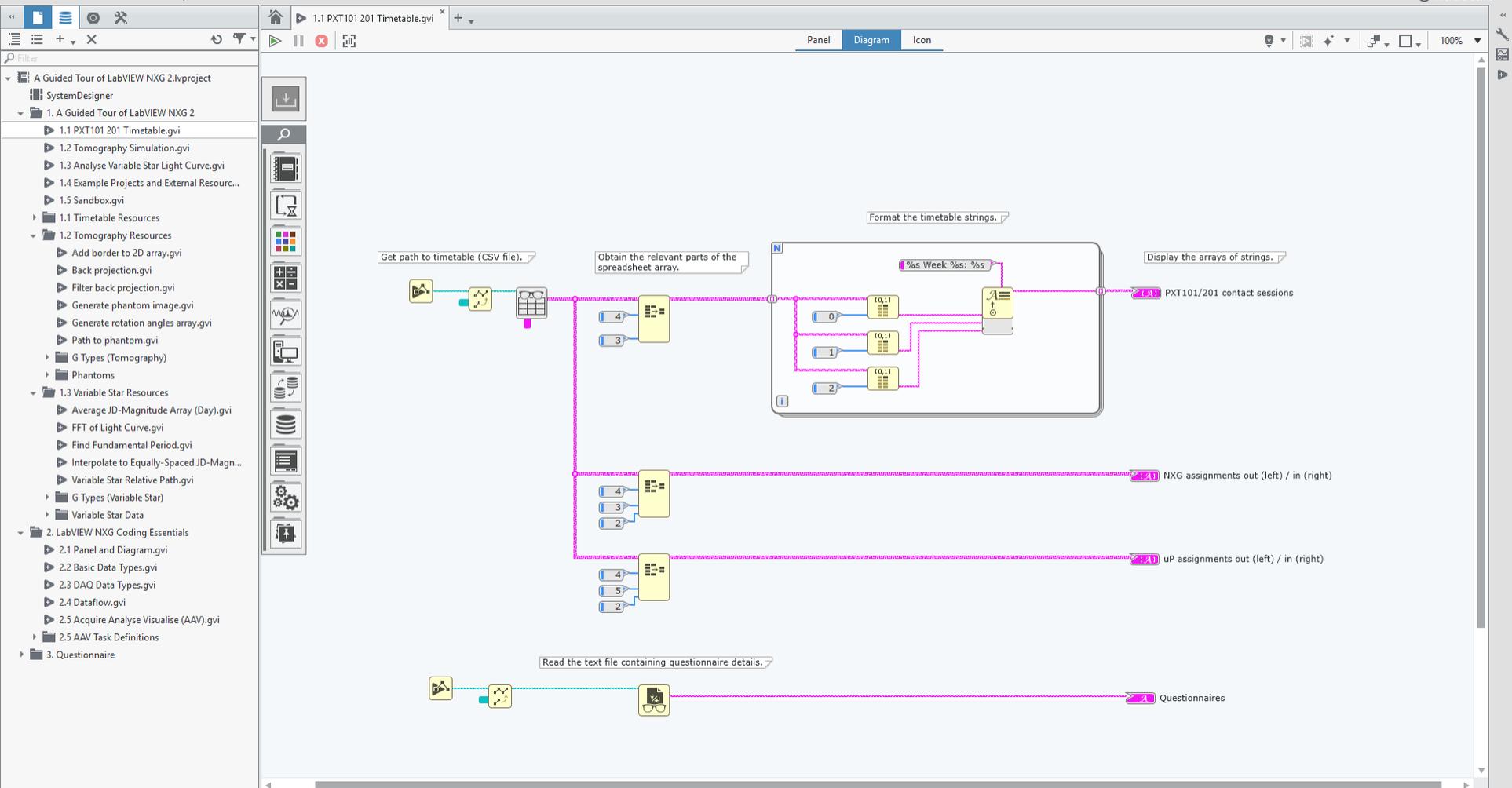
Questionnaires

- Every week there will be a laboratory questionnaire embedded in the contact session project as a VI;
- The questionnaire will consist of two or three questions directly related to the contact session;
- You can answer these as we go through the contact session (you will be given time to do this);
- Every questionnaire will require you to write some simple LabVIEW code in the VI and / or answer a question by writing a comment;
- Discuss the questionnaire in your laboratory pairs and come up with your answers together;
- When done, email the VI to Richard (LewisR54@cardiff.ac.uk) before you leave the session;
- Make sure that you include both your names in a comment in the VI (so I know whose VI I'm marking!);
- The questionnaires are worth 10% of the mark for PXT101 / PXT201.

The questionnaire for this session can be found in the navigation pane:

A Guided Tour of LabVIEW NXG 2 \ 3. Questionnaire \ Questionnaire 1.gvi

A contact session can be contained entirely within a LabVIEW NXG project. The use of external materials and other applications has been minimised to maintain session flow.



Embracing the “everything in a project” concept allows for unexpected teaching and learning opportunities. Here we can use the course timetable to demonstrate dataflow!

File Edit Run Data View Help

3.1 Bottom-Up Development.gvi

Panel Diagram Icon

150%

Filter

- AAV and IO III Coding from Scratch...
- SystemDesigner
- 1. Questionnaire 2 Solutions
- 2. Software Development Best...
- 2.1 Software Development...
- Functional Specification
- Old Functional Specificatio...
- 3. Bottom-Up Development
 - 3.1 Bottom-Up Developme...
 - 3.2 Completed Application...
 - 3.3 BU Application.gvi
 - 3.2 Completed Application...
 - GTypes
 - subVIs
- 4. Top-Down Development
- 5. Blackbody Simulation
 - 5.1 Blackbody Simulation.gvi
 - GTypes
 - SubVIs
 - Tests
- 6. Questionnaire
- 7. Lab Exercise 2

3.1 Bottom-Up Development.

In general terms, bottom-up development means:

- Thoroughly planning your application with a functional specification and supporting documents;
- Selecting an appropriate architecture for your application ("script", state machine, etc);
- Writing subroutines / functions / classes that encapsulate the functionality that you will require;
- Thoroughly testing elements of functionality with combinations of these subroutines / functions / classes;
- Writing the boiler-plate code for the main application;
- Building up the functionality by inserting calls to subroutines / functions / classes directly into the boiler plate code, thoroughly testing as you go.

In LabVIEW terms, bottom-up development means:

- Thoroughly planning your application with a functional specification and supporting documents;
- Selecting an appropriate architecture for your application ("script", state machine, etc);
- Generating a LabVIEW project with the appropriate application template;
- Encapsulating required functionality in fully-functional and documented subVIs / classes / type defs;
- Testing elements of functionality with combinations of these subVIs / classes / type defs;
- Generating the main VI, together with all states (if a state machine), logic, loops, etc required;
- Building up the functionality by inserting subVIs / LabVIEW classes / type definitions directly into the main VI, thoroughly testing as you go.

We will now develop a simple calculator application bottom-up...

Errors and Warnings

Richard Lewis

Item	Panel	Document
ABC	Name	Text
	Type	Text

Visual style

Segoe UI 9 B / U

Text color

Layout

Auto size

Width 744 Height 112

Programming assistance

Create reference

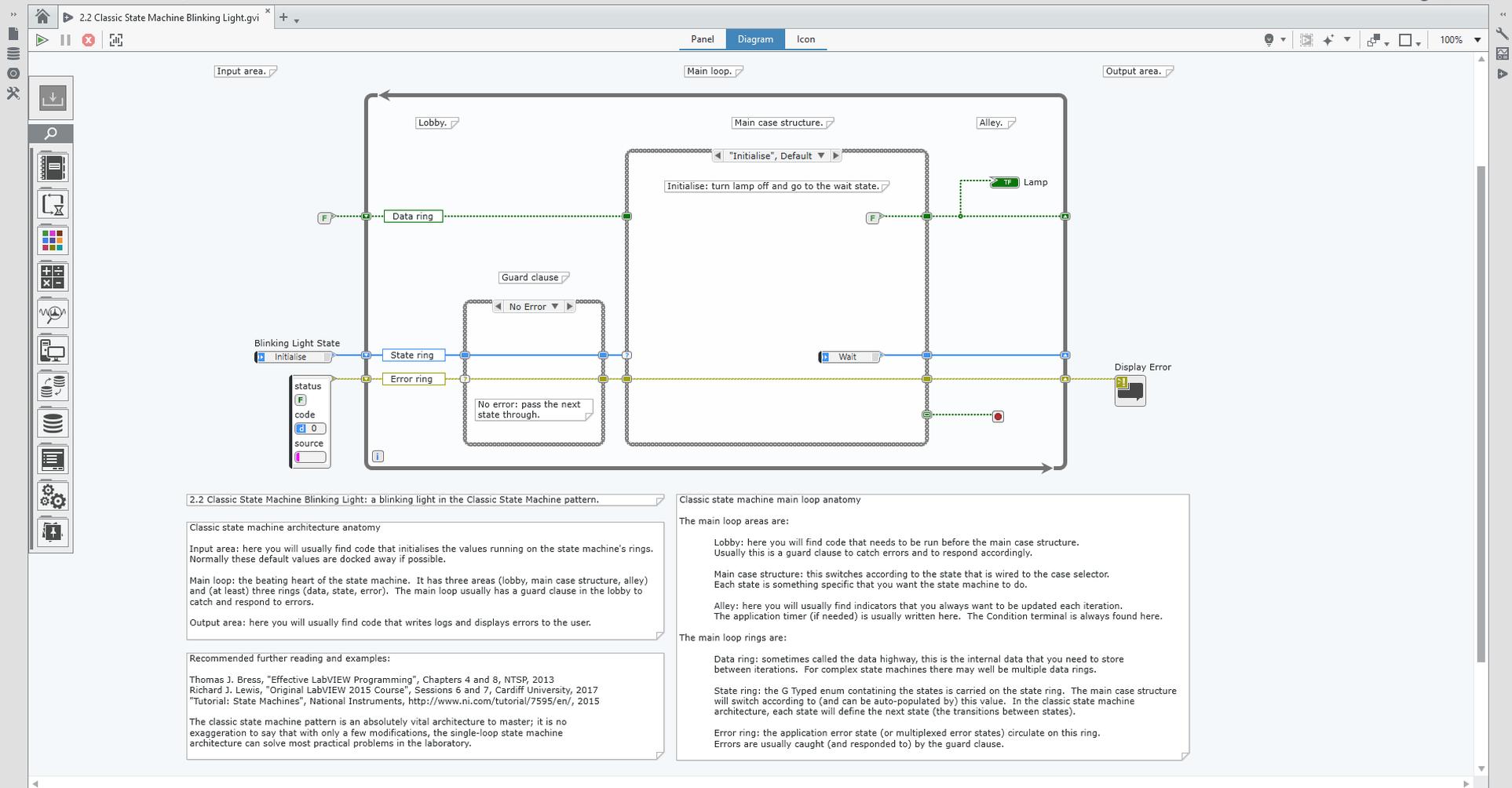
Context help description

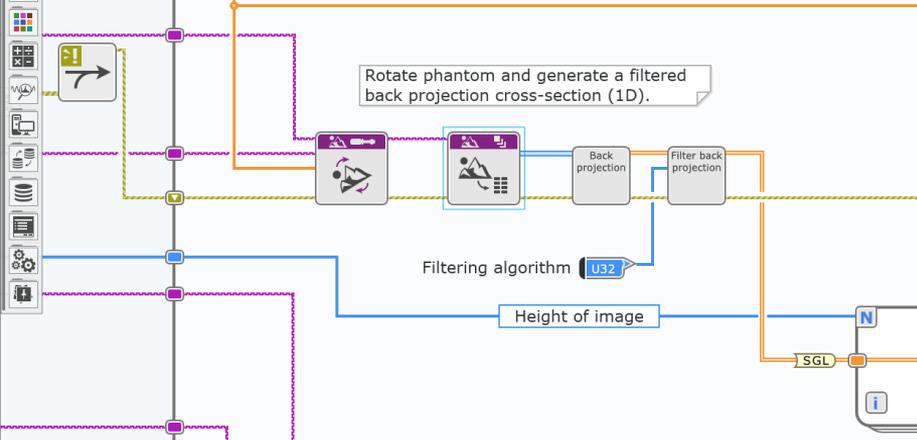
Tooltip

Documentation

Context Help Online manual

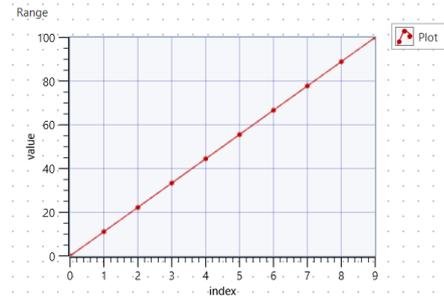
The text formatting tools in NXG 2.1 apply to entire text blocks. In the Panel shown here are no fewer than six text blocks.





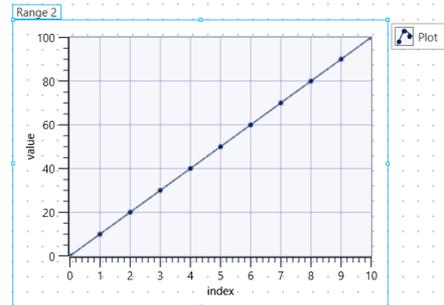
The advantage of this pattern is that it will always include the end points. The main disadvantage is that it may require some mental arithmetic (or a calculator) to specify a required step size.

See "Generate Range (Number of Steps).vi" for an example of this functionality encapsulated in a sub VI.



The advantage of this pattern is that it will always include the end points. The main disadvantage is that it may require some mental arithmetic (or a calculator) to specify a required step size.

See "Generate Range (Step Size).vi" for an example of this functionality encapsulated in a sub VI.



NXG interface advantages and issues

- NXG UI is vastly superior to LabVIEW UI in almost every usability aspect
- Panel and Diagram zoom enhances delivery, recording, and development
- Docking constants and the properties pane really help with clarity
- The ability to zoom necessitated a rethink of what “good style” means
- NXG is noticeably slower and less stable compared to LabVIEW
- Some consistency issues: analysis panel only available for waveforms, e.g.

Guided Tour of LabVIEW NXG 2 - LabVIEW NXG

File Edit Run Data View Help

1.2 Tomography Simulation.gvi

Panel Diagram Icon

200%

Item Panel Document

Image to Array.gvi

Terminals

- image in
- optional recta... Create constant
- error in
- SQL pixel values
- image out
- U8 pixel values
- U16 pixel values
- U16 pixel values
- error out

Constants

Create all

Dock Undock

Visual style

Show label

View

- Icon
- List

Documentation

Context Help Online manual

Angles

Rotate phantom and generate a filtered back projection cross-section (1D).

Filtering algorithm U32

Height of image

Back projection

Filter back projection

SGL

Generate fi

The zoom function is a killer feature on the Diagram. It is particularly useful in session recordings to remove visual clutter.

i. The "start, end, number of steps" pattern

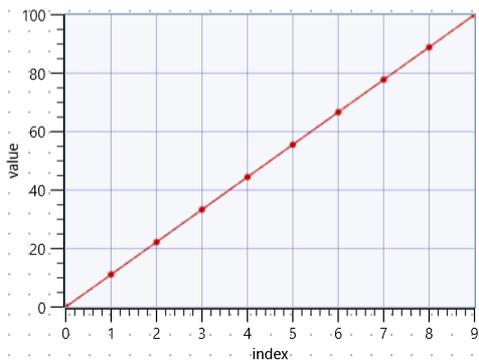
Start value Number of steps

End value

The advantage of this pattern is that it will always include the end points. The main disadvantage is that it may require some mental arithmetic (or a calculator) to specify a required step size.

See "Generate Range (Number of Steps).vi" for an example of this functionality encapsulated in a sub VI.

Range



ii. The "start, end, step size" pattern.

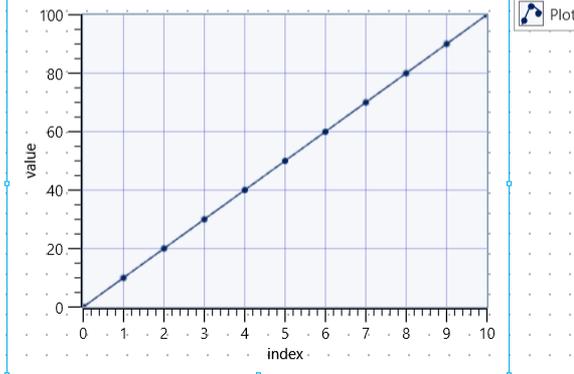
Start value 2 Step size

End value 2

The advantage of this pattern is that you can specify a required step size. The main disadvantage is that the step size you specify may not include the ending number. You have to decide how to handle such cases; the usual approach is to only go as far as the largest number before the ending number, unless the step size divides the range exactly.

See "Generate Range (Step Size).vi" for an example of this functionality encapsulated in a sub VI.

Range 2



Item Panel Document

Name Range 2
Type Cartesian Graph

Parts

- Plot legend
- Cursor legend
- Scale legend
- Graph tools

Plots

Metadata always overrides plot names

Visual style

- Anti-aliased

Label

- Show label
- Label color
- Font: Segoe UI, 9, B, /, U, -
- Label placement: Top left

Content font

- Font: Segoe UI, 9, B, /

Layout

Width: 392, Height: 296

Plot area

- Auto size
- Left: 50, Top: 8
- Right: 7, Bottom: 43

Behavior

Object: Control, Indicator

The zoom function works equally well on the Panel. It is particularly useful in session recordings to remove visual clutter.

2.1 File Input.gvi

Filter

- AAV and IO II Coding Drag and Drop.lvproj...
 - SystemDesigner
 - 1. Questionnaire 1 Solutions
 - 2. Working with Files (IO)
 - 2.1 File Input.gvi
 - 2.2 File Output.gvi
 - 2.3 TDMS IO.gvi
 - IO Resources
 - 3. Loops and Flow Control
 - 4. Creating VIs and subVIs, LabVIEW NX...
 - 5. Applications of AAV and File IO
 - 5.1 Analyse Variable Star Light Curve...
 - 5.1 Variable Star Resources
 - Average JD-Magnitude Array (Da...
 - FFT of Light Curve.gvi
 - Find Fundamental Period.gvi
 - Interpolate to Equally-Spaced JD...
 - Save FFT Data to File.gvi
 - Variable Star Relative Path.gvi
 - G Types (Variable Star)
 - Variable Star Data
 - 6. Questionnaire
 - 7. Laboratory Exercise 1

2.1 File Input

LabVIEW NXG supports a number of ways to get data into and out of your applications. The methods below get increasingly low-level as you go down the Diagram. We'll use here the example of accessing a small file containing comma delimited values (CSV, files are usually file_name.csv). The process for writing is similar, see 2.2 File Output for details.

Enable one subdiagram at a time to execute that method's portion of code when the VI runs. Remember that you can zoom the Diagram with CTRL+MOUSEWHEEL.

Method 1: use "Read Delimited Spreadsheet" without any file specified, which will need to ask the user to choose the file. Loads the entire file into memory in one go.
High-level, slow, heavy on memory usage, but quick to write.

Method 2: supply a path programmatically to "Read Delimited Spreadsheet". The user doesn't have to do anything. Loads the entire file into memory in one go.
High-level, slow, heavy on memory usage, but quick to write.

Method 3: separate the open, read, and close file actions, but load the entire file in one go. The file path is programmatically supplied either as a path or as a file reference.
This is an example of the open-act-close paradigm.
Mid-level, speed dependent on how much is being written, variable memory usage, allows modifying of the file before the file is closed, requires the user to convert the data out of string format.

Method 4: separate the open, read, and close file actions, and load a specified number of characters from the file. Usually you would use a loop to repeatedly perform this action. The open and close nodes sit outside of the loop, so the file is only opened and closed once.
This is an example of the open-act-close paradigm.
Mid-level, fast, light on memory usage, allows modifying of the file before the file is closed, requires the user to convert the data out of string format.

Richard Lewis

Item Panel Document

Read Delimited Spreadsheet (DBL).gvi

Function configuration

DBL

Terminals

- file
- error in Create constant
- format %3f
- delimiter
- new file
- all rows
- error out

Constants

Create all

Dock Undock

Visual style

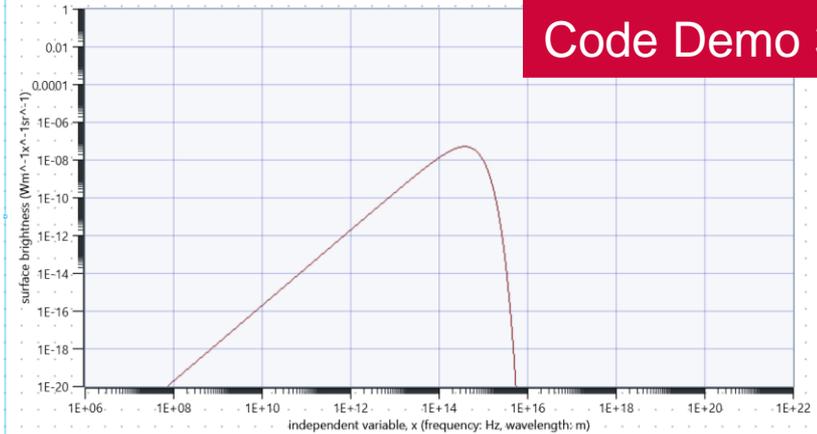
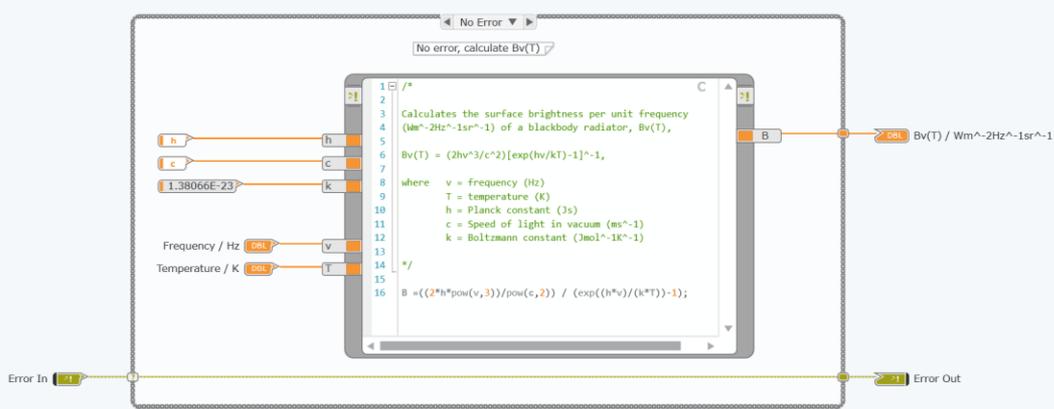
Show label

View Icon List

Documentation

Context Help Online manual

The zoom function aids exploration from the student's perspective, allowing clearer focus on individual areas of code.



(Somewhat) nasty surprises: some workarounds required

- Analysis panels (from direct acquisition / capture) unreliable across PCs
- Cannot distribute compiled C Node code (crashes target PCs)
- G Types do not always work as expected (i.e. as strict type definitions)
- LabVIEW NXG 2.0 code will not load at all on LabVIEW NXG 2.1 machines
- Have reported the bugs to NI, will provide code to reproduce the errors

Untitled Project* - LabVIEW NXG

File Edit Run View Help

Project Files Captured Data Debugging Tool Launcher

SystemDesigner x Analog Input.task x Input 0 x Analysis panel: FFT Spectrum x

Capture 1* 01/16/2019 18:15:27

Time domain Input 0 (16/01/2019 1...)

Spectral magnitude Capture & view data

Spectral phase Capture & view data

Window type: Hanning

Input signal (black line)

Spectrum (blue line)

Configured code

Copy to clipboard

The analysis panel is not available for all data types. Attempting to invoke the analysis panel occasionally crashes NXG 2.1

5.1 Blackbody Simulation.gvi + Bv(T)_frequency.gvi +

Panel Diagram Icon

Calculates the surface brightness per unit frequency ($Wm^{-2}Hz^{-1}sr^{-1}$) of a blackbody radiator, $Bv(T)$:

$$Bv(T) = (2hv^3/c^2)[\exp(hv/kT)-1]^{-1}$$

No Error

No error, calculate $Bv(T)$

h c 1.38066E-23

Frequency / Hz DBL v

Temperature / K DBL T

```

1 /*
2
3 Calculates the surface brightness per unit frequency
4 ( $Wm^{-2}Hz^{-1}sr^{-1}$ ) of a blackbody radiator,  $Bv(T)$ ,
5
6  $Bv(T) = (2hv^3/c^2)[\exp(hv/kT)-1]^{-1}$ ,
7
8 where v = frequency (Hz)
9       T = temperature (K)
10      h = Planck constant (Js)
11      c = Speed of light in vacuum ( $ms^{-1}$ )
12      k = Boltzmann constant ( $Jmol^{-1}K^{-1}$ )
13
14 */
15
16 B = ((2*h*pow(v,3))/pow(c,2)) / (exp((h*v)/(k*T))-1);

```

B DBL $Bv(T) / Wm^{-2}Hz^{-1}sr^{-1}$

Error In Error Out

Execution: learning and application
Impact of NXG on problem-based learning



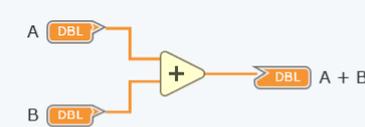
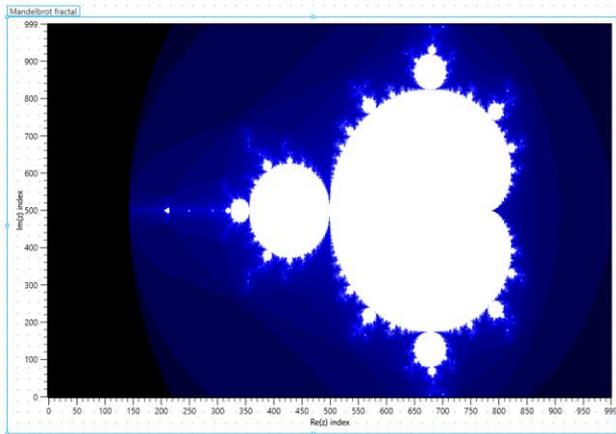
Impact of NXG on problem-based learning

Aims: LabVIEW NXG course contact sessions

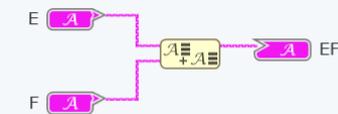
- Start useful, stay useful
- Flatten the learning curve
- Remain relevant to Physics and Astrophysics
- Leverage NXG's strengths (UI, projects, etc.)
- Retain old module's USPs (funcspects, etc.)

Aims: Student-lead micro-projects

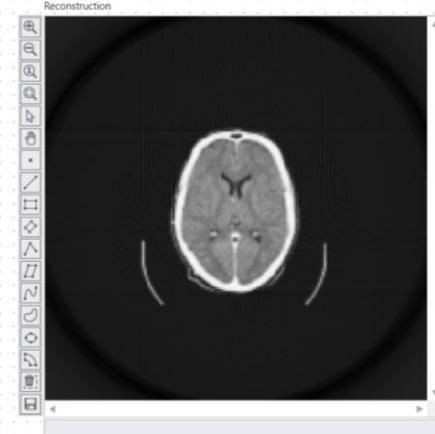
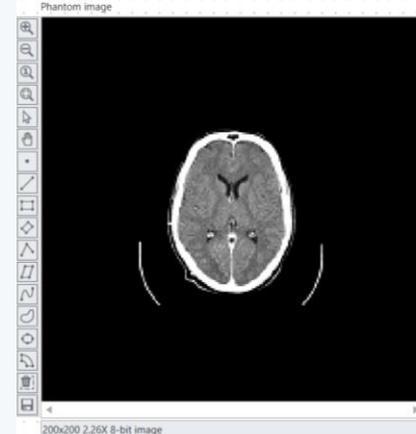
- Start useful, stay useful
- Allow more sophisticated applications earlier
- Provide a common language for related projects to encourage collaboration
- Retain old module's USPs (community, etc.)



Operation 1: calculate the sum of A and B

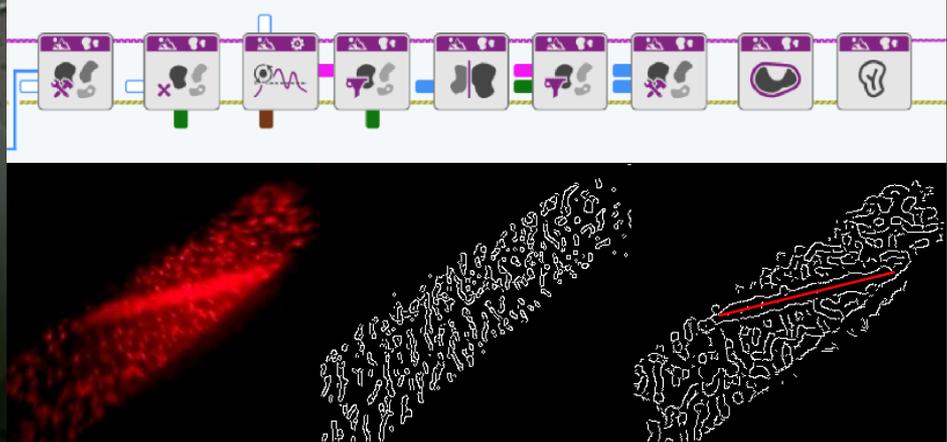


Operation 3: concatenate strings E and F

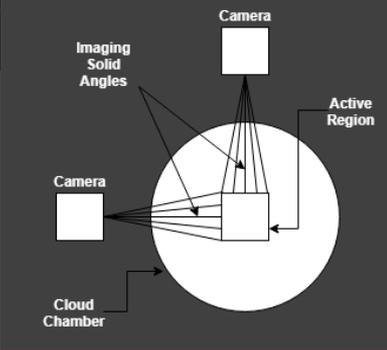


Impact of NXG in contact session problem-based learning

- Flashes of brilliance when NXG's strengths all came together
- Freely-flowing sessions dramatically accelerated learning, made teaching more efficient
- Staying “in project” allowed students to concentrate more effectively on their tasks
- Full retention of original module's USPs, group work in particular enhanced
- Instability and bugs introduced interruptions in otherwise freely-flowing, dynamic sessions
- DAQ-first concept partially thwarted by crashes, but the concept itself is sound

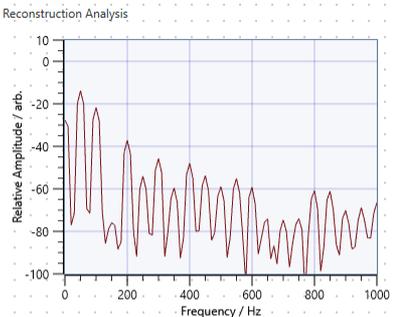
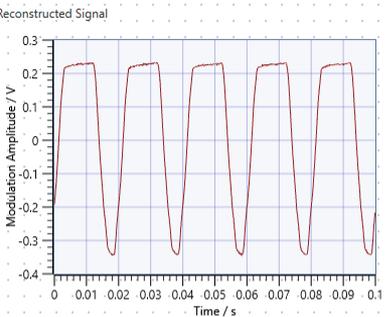
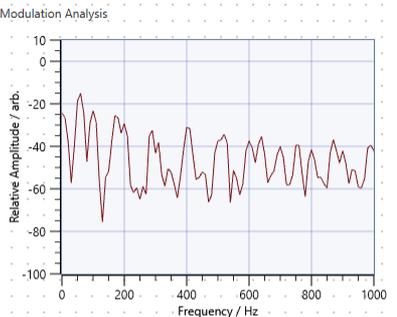
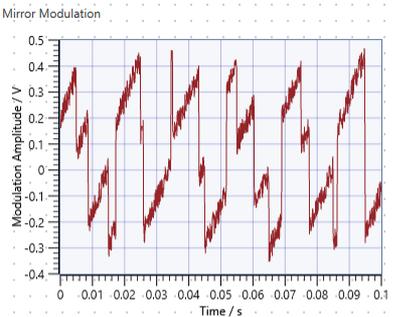


Code Demo 4



Impact of NXG in student-lead micro-projects

- This is where NXG really shone: a noticeable increase in efficiency
- Image analysis in particular was found to be much more accessible
- Students explored the language in much more depth than in previous years
- Students progressed much further in the micro-projects than in previous years
- One micro-project formed the basis of an exhibition at a Royal Society Event...



Waveform Parameters

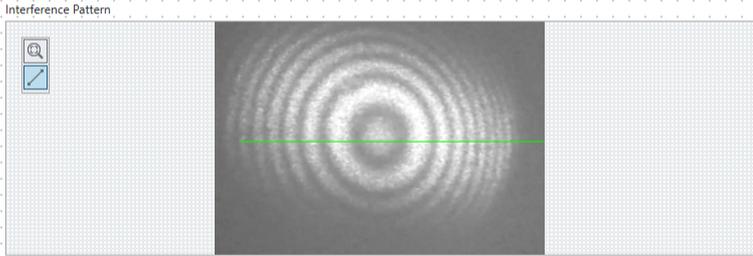
Sine	Square	Triangle	Sawtooth
Amplitude: 0.1, 0.2, 0.3, 0.4			
Frequency: 10, 500, 1000			
Phase: 0, 100, 200, 300, 360			



Stop Application

Noise level

0.1, 0.2, 0.3, 0.4



Execution: learning and application
(Other) logistical issues and their solutions

(Other) logistical issues and their solutions

Point release 2.0 to 2.1 broke compatibility(!)

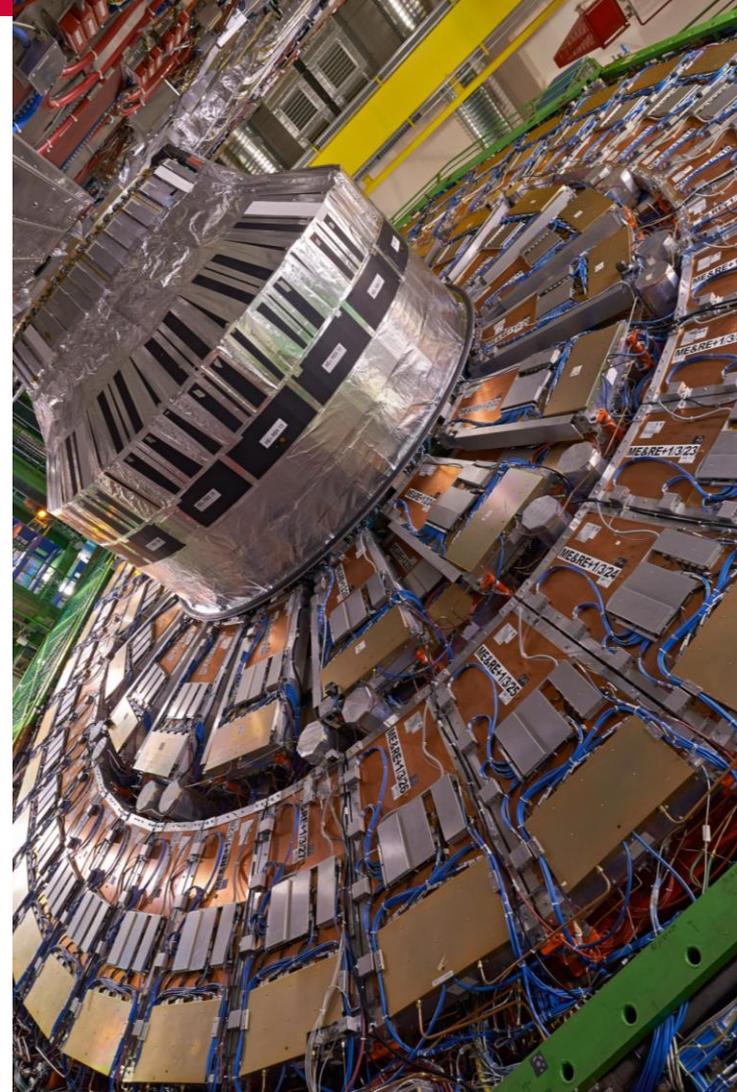
- No easy way around this; code was re-written
- PCs in PHYSX migrated to 2.1

Some code elements crashed on lab PCs

- Hard to pin down; some PCs worse than other
- Drive towards “safe” implementations
- Avoided these elements in assessments

Freezes, crashes, and bugs

- Very rarely show-stopping in practice
- Bugs reported to NI



Outcome



Student reception of new core module

Student module evaluation scores

Academic Year	LabVIEW Version	Score / 100
2018/19	NXG 2.1	86
2017/18	2015	94
2016/17	2015	90
2015/16	2015	94

Student feedback

- Overall very positive, all students felt they got something positive from the module (projects)
- Astrophysicists see the value of LabVIEW, but some would have preferred to develop their Python instead
- LabVIEW generally valued, especially on projects

Future MSc core module design (2019/20)

Scaling (>30) and broadening of MSc student cohort

- MSc Gravitational Wave Physics comes online for 2019/20
- Additional micro-projects and relevant examples required
- Currently reviewing language teaching structure for 2019/20

Will we use LabVIEW NXG for 2019/20 or revert to 2015?

- Contingent on outcome of module review
- Will evaluate NXG 3.0 for stability improvements
- Now have full resources (inc. videos) for 2015 and NXG

Will we use NXG for the advanced programming module?

- Main problem is the lack of RIO support in NXG
- More difficult to migrate advanced materials to NXG
- Retain LabVIEW 2015 for now (considering 2018 SP1)





Summary: critique of LabVIEW NXG 2.1 for teaching and micro-projects

- **Usability:** excellent for micro-projects, more variable for teaching, but good.
- **Reliability:** NXG can be slow, crashy, and buggy. Often masks true potential.
- **Materials:** can stay in-project, so more efficient. Really need workbooks!
- **Student reception:** excellent for micro-projects, overall positive for teaching.

- **Overall:** NXG 2.1 has its issues, but the potential for brilliance is there.



Contact details

Dr Richard James Lewis

Director of Postgraduate Studies
School of Physics and Astronomy
Cardiff University

Tel: +44(0)29 2087 5433

Email: LewisR54@cardiff.ac.uk

URL: <https://tinyurl.com/y94cfmk5>

Case studies, presentations

Transitioning MSc Physics Teaching to LabVIEW NXG 2.0:
From Drills to DAQ-First (NI AUF 2018)

Reflections on LabVIEW as a Common Language for
Community and Skill Building (NIDays 2017)

LabVIEW as a Common Language for Community and Skill
Building (NI AUF 2016, NIWeek 2017)

MSc Physics Students Take Ownership of their Learning with
LabVIEW (NI EIA 2016)

Bringing the Research Group Ethos into Taught Masters
Learning (VICE/PHEC 2016)

www.cardiff.ac.uk/physics-astronomy