inxware Tools

Tutorial

inxware 1.0

Document History

Version	Primary Author(s)	Description of Version	Date Completed
1.0	MSR	Initial issue	15-May-2007
2.0	MSR	Updated to reflect developments	19-Dec-2007
2.1	PMLD	Re-structured	10/10/10

List of Figures

Table of Contents

Introduction

Inxware tools are used to develop software for inxware enabled devices. The tools-suite and evaluation software to run applications on desktop PCs is available from www.inx-systems.com.

Purpose and audience

This document is the tutorial for the Inxware development environment. It provides:

- working instructions of how to build example applications
- an accompanying technical explanation of inxware technology

The tutorials provide a quick way of getting started with Inxware.

Target Supplements

Some devices (Targets) have different capabilities to others. This means some components are not available for all targets. Profiles for different target types are provided with the tools set and checks are made at load time to ensure compatibility. This document does not contain target-specific information and refers only to the standard + GUI profile. Supplements for other profiles can be downloaded from www.inx-systems.com

Document Routemap

Recommended Sections

It is suggested you work through the tutorials in order, from the start. This is because tutorials (other than the first) assume knowledge gained from previous tutorials.

Document conventions

Text-Labelling

- This means that the indicated paragraph describes a feature that may vary under different targets

Menu Choices

When describing a sequence of operations on the user interface, the following example format is used:

[IAB:Component palette] Operators:Add4:Real {Drag To composition window}

The application concerned, and its local context is listed first. Menu selection operations are shown with →. So the above example could be described as:

In the IAB 'Component palette' window, navigate to the node labelled 'Real', which is in the Operators, Add4. Drag this to the composition window.

Similarly:

[IGD:Toolbar] Close Project Icon {LeftClick} would translate as:

Left-click on the 'Close Project' icon in the IGD main toolbar.

Dialog Buttons

[OK] means for example that you should 'left-click' the dialog's 'OK' button. Similarly, [Cancel] would mean to click the 'Cancel' button, and so on.

Explanation Sections

Sections marked like this contained background explanation of how Inxware works, particularly in relation to the given tutorial.

Glossary

MCU	Micro Controller Unit (peripheral enhanced micro-processor)
RTOS	Real-Time Operating System
Assert	The action of sending an event on a *function block's* output *trigger*.
Data Port	A *port* that is responsible for sending or receiving data (as opposed to events).
Function block	A collection of operations on data that are *fired* by input *triggers*, read data on input *data ports*, write data to output *data ports* and *assert* output *triggers*.
Inxware Application Builder (IAB)	The central Inxware PC Tool, using this the user can design and edit Inxware applications
Inxware GUI Designer (IGD)	An Inxware PC Tool for defining precisely the appearance of an application's GUI components
Inxware Component Builder	The ICB is a tool used by programmers to produce inxware interfaces to native function blocks developed in C. This tool is provided separately to the standard inxware tools as an eclipse plugin.

Port	A connector on a *function block*. A port can be an input port or an output port.
Trigger	An event *port*. A trigger may be an input port or an output port.

Working Copies of Tutorial Projects

Location

These can be found in the Inxware projects folder, under ..\examples\tutorials

Introduction to IAB “Hello World”

Introduction

Getting started with IAB is a process of creating or loading a project and editing the graphical representation of system components. This process is split into four sections: Project Creation, Application Development, GUI Layout, and deployment to target.
The application makes use of the GUI tools and simply renders a graphic image on the screen. GUI features are used extensively in these tutorials as they provide an easy interactive environment for the developer to explore how applications behave at runtime.
This tutorial introduces some of the most basic concepts of Inxware. It also represents a good way of testing that your installation of Inxware is working properly.
The tutorial is highly verbose, however most operations are standard desktop application operations and once a developer is familiar with the simple processes in inxware, the developer would be able to create the application in this tutorial in around 60 seconds.

Starting IAB (Tutorial 1A)

Start IAB, via the Windows desktop launcher:

[Windows ‘Start’ button] All Programs->inx->Inxware->Inxware Application Builder {LeftClick}.

IAB Overview

The IAB window should soon appear, as shown in Figure 1.

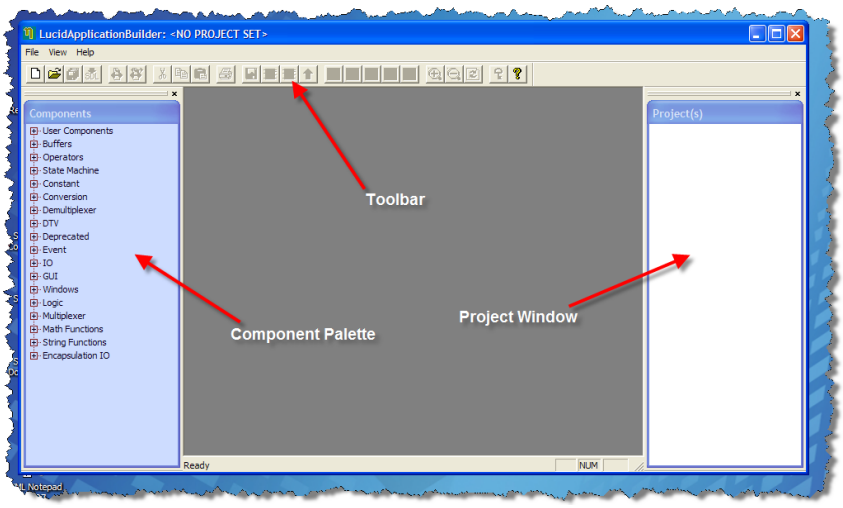

Figure 1: The IAB Tool

The Main Windows

This IAB tool presents the Component Palette on the left: this is a list of components which you can use in your application.
The Project-overview window is on the right: this lists the projects and files associated with your edit session. Because no project(s) are open, the project-overview window is empty.

Create a new Project

IAB is the 'main' tool, in that it is with this that one starts new projects, and defines the application's behaviour. Start a new project as follows:

[IAB Main Menu] File->New project {Left Click}

A dialog appears as shown in Figure 2, allowing creation of a new project in the opening 'default' folder. Clicking on the […] button beside the 'Project Root Directory' field invokes a folder-navigation dialog, as shown in Figure 3. We suggest you create a folder called 'tutorials', with a subfolder called 'tutorials_1' as the project root folder. Create a project called 'tutorial_1A' in the folder.

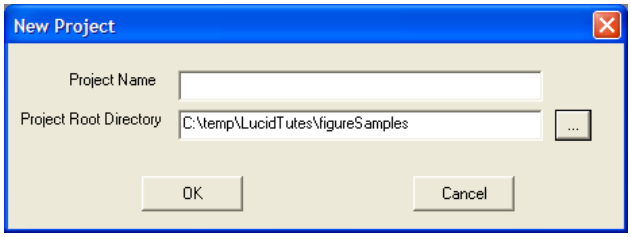

Figure 2: New Project Dialog

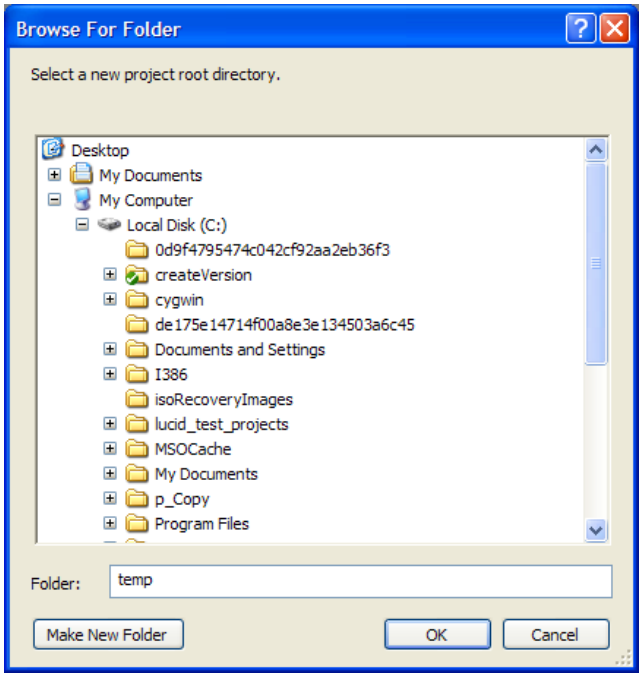

Figure 3: New Project Navigator Dialog.

A composition window will appear centrally in the IAB tool – ref Figure 4. A composition window is where you add components and connect them to create your application.

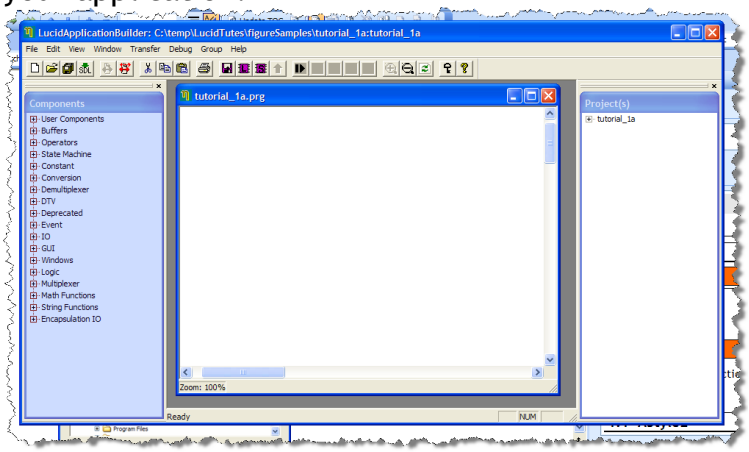
Figure 4: IAB with composition-window showing

Add Function-Blocks

Function blocks are the building blocks of any application. To add function-blocks to the application, drag them from the function-block palette and drop them on the (chosen) composition window.

[IAB:Component Palette] ' Event->Event OR->OR 2' {Drag-and-Drop to composition window}

[IAB:Component Palette] 'GUI->GUI Image'{Drag-and-Drop to composition window}

The IAB window should look like Figure 6.

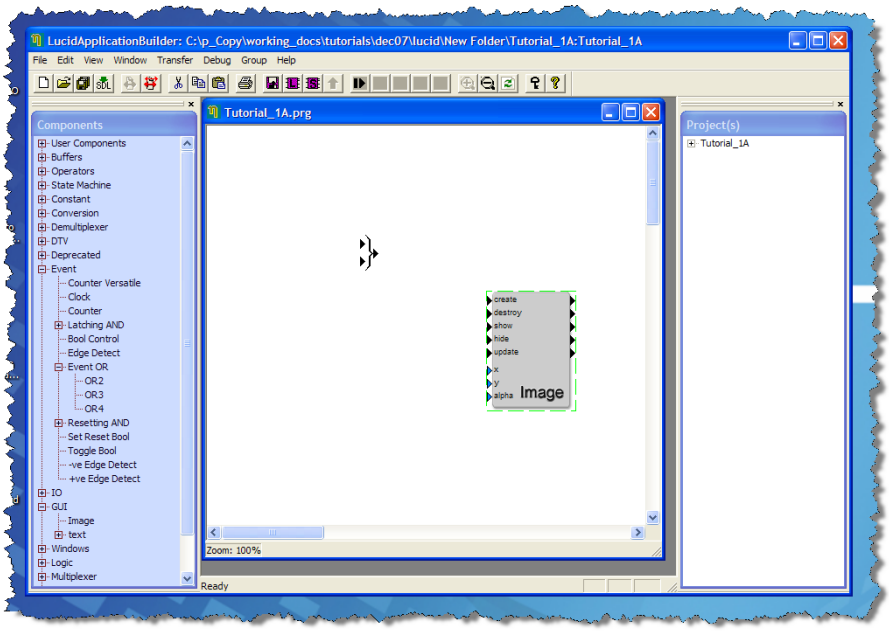

Figure 5: IAB with initial components

The Function-Block Palette.

This window shows a tree-structure, the branches of which reflect the available categories of function block. By expanding the branches of the tree, you can navigate through the sub-categories towards 'leaf' nodes, i.e. items that are rightmost on a particular branch. Only selectable items can be drag-and-dropped.

De-brief

That concludes the opening part of this tutorial.

You can either carry straight on with the next part of this tutorial, or take a break – in which latter case, exit IAB, via.

[IAB Main Menu] File->Exit {Left Click}

Develop the Application (Tutorial 1B)

Introduction

In this tutorial, we further develop the application we started in tutorial 1A, connecting and initialising the relevant function-block ports. We also take note of some more general usability-enhancement facilities.

Open the Application, save it 'As'

Just in case we make any mistakes – and just to make sure you know about the feature – we will copy the former application to another project before we start further work on it. Firstly, we will open the previous application via Windows Explorer, this time as follows:

(Open Windows Explorer),

Navigate to the folder in which you saved Tutorial_1A

Double-click the 'Tutorial_1A.lpj' file.

IAB should start after a few seconds .Now save the application as Tutorial_1B, via:

[IAB Main Menu] File->Save As {Left Click}

and in the ensuing dialog, save as Tutorial_1B under folder 'Tutorials_1'.

Re-position the Function-blocks

We will first see how to move function-blocks around in the editing window.

Move the cursor over a function-block.

Note that the cursor changes to a rectangle to confirm when it is over a function-block.

[Lft-Clk]->Drag when over a function-block.

Note that the position of the function-block is dragged with the cursor.

If necessary, move the function-blocks into the positions shown in Figure 6 by dragging them with the cursor.

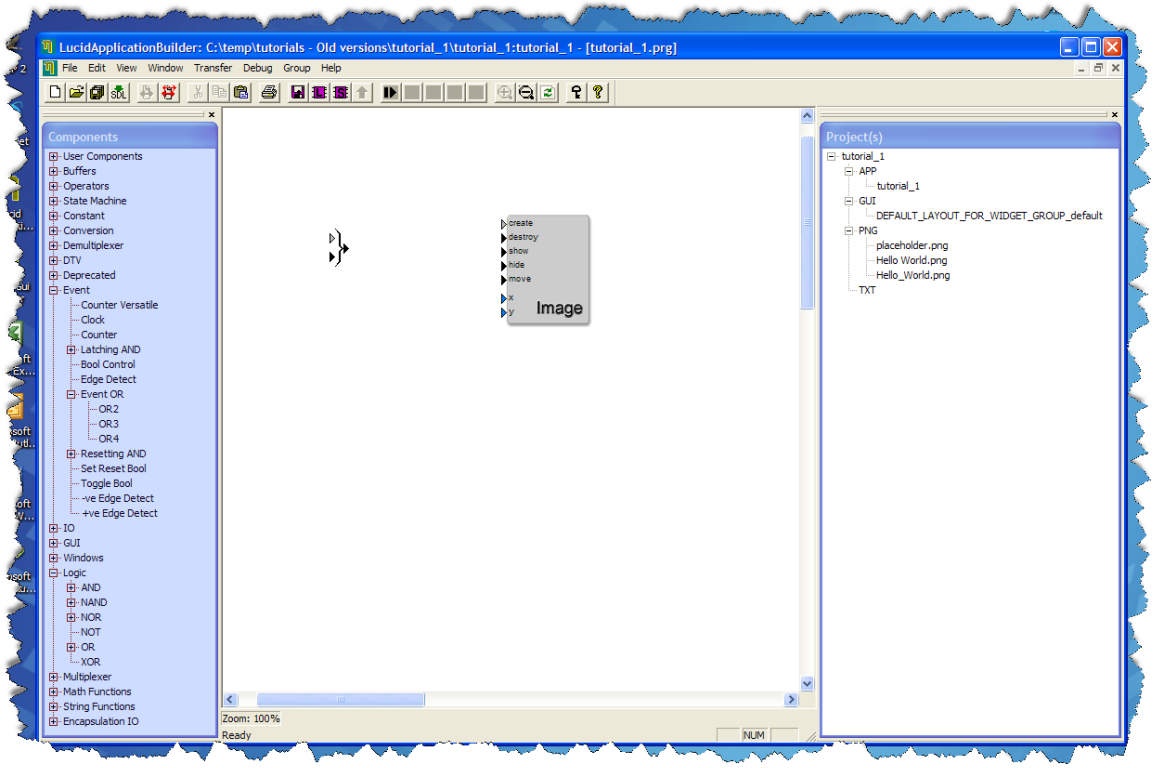

Figure 6: Tutorial 1 components, unconnected

Connect Ports

Move the cursor over the output port of the OR2 block.

Note that the cursor changes into a black triangle to confirm it is over a port.

Left click on the output port of the OR function-block, and drag across to the 'show' port of the Image function-block.

Note that a connection-line stretches from the output port as you drag, and that, when over the 'show' port, the cursor changes back to the 'over port' symbol.

Whilst over the port labelled 'show', release the left button.

Note that a 'routed' (i.e. with horizontal and vertical segments) connection is created.
Your application should look like Figure 7.

Initialise Ports

It is often required that processes are started at application load time, which in turn requires an initial event to be asserted by the system and connected to

function blocks that are required to run on start up. A convenient method of specifying a start event is asserted at start up is to configure the port itself to be asserted at start up. This is achieved as follows:

[OR2: Input port]->initialize {Right Click}.

[Image: 'Create' Input port]->initialize {Right Click}.

Note that the 'initialise' label becomes ticked in each case. In fact, the [RtClk]->initialize toggles the state of initialisation from true to false.

About Initialisation

Initialisation is provided by Inxware to allow you to control what functions execute at start up. Many function blocks require a create, configure, or initialise process to run before they are used for operations.

Save the project

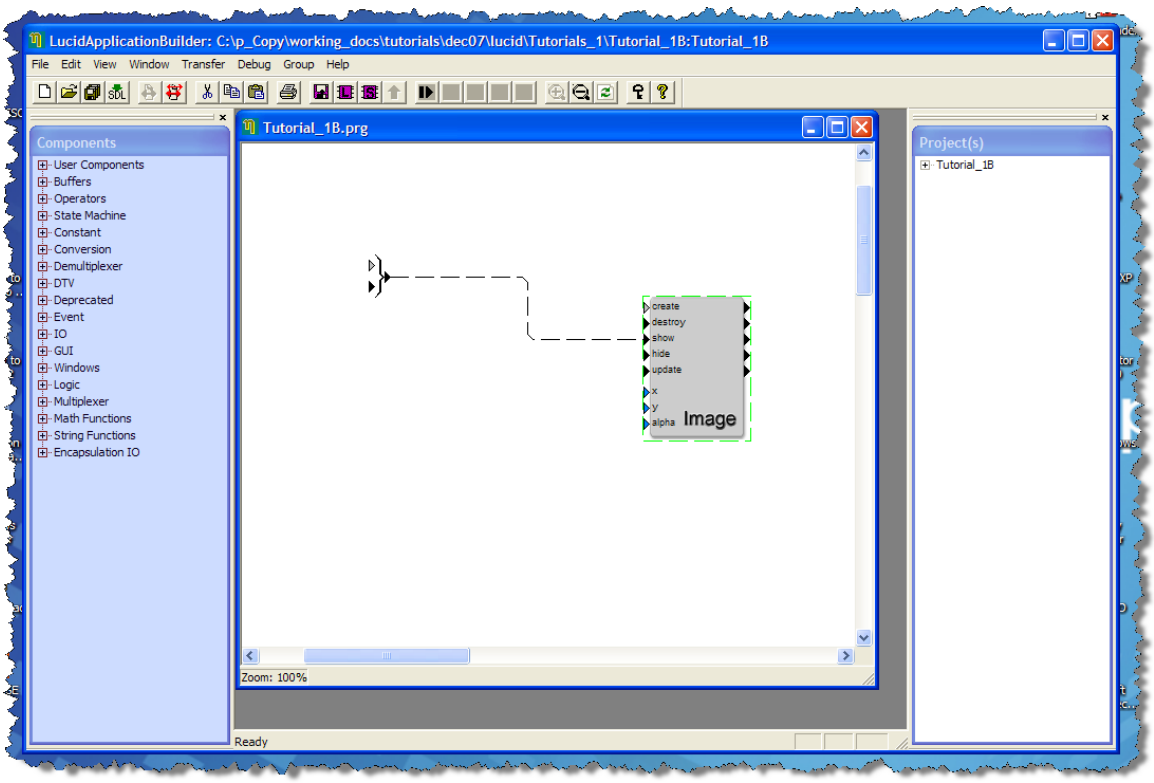

Figure 7: Tutorial 1 components, connected

You now have completed at a functional level what the application will do, but for GUI applications the GUI components must be further configured using the IGD tool to select what image is displayed and to position this within the target's graphical display.

In this step we have skipped defining a tag name for the widget and used the default name generated by IAB. Typically this should be avoided and developers are advised to give EVERY widget a meaningful name.

Layout the image (Tutorial 1C)

Introduction

This tutorial explains how to select images (or any other GUI component) that you may have chosen to display in your application. This involves the use of IGD, and requires an understanding of the dependency between IGD and IAB to fully specify your application's behaviour.

Prerequisites: The SODL ('.sdl') file

Before doing anything else, (and before we forget), we will get IAB write out the SODL file, which is scanned by IGD for active GUI components. GUI components are only active of one or more of their input events are connected or initialised.

Start IAB, open project 'Tutorial_1B', and save it as 'Tutorial_1C'.

Write SODL, either by:

[IAB:Main Menu] File->Write SODL {Left Click}, or

[IAB:Toolbar] Write SODL icon {Left Click} (see Figure 8)

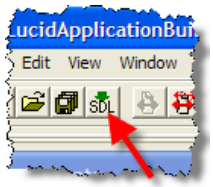

Figure 8: The IAB 'Write SODL' button

This causes the SODL to be compiled into a single description file within the project for reference by helper tools such as IGD.

IAB and IGD

IAB is the "hub" tool used in inxware to integrate components and define functional details of how applications behave. It does not however provide specialised environments for configuring more complex types of components such as GUI widgets. To aid the developed in configuring such components in specific environments (e.g. graphical layout), "helper tool" plugins are provided to aid developers.

IGD is the GUI helper tool for inxware. IAB retains control of the dynamic functional aspects of the "widget", but IGD is responsible for the aesthetic

("skinning") features of the component. Aesthetic parameters include colour, transparency, position, size and/or associated graphics files.
IGD is a slave to IAB creating the existence of GUI widgets, however applications cannot be deployed to devices if they contain GUI widgets that have not been configured by IGD. The connection between widgets represented in IAB (as function blocks) and IGD (as screen elements) is established by the use of descriptive "tag" names defined in IAB for each widget. Grouping of widgets is also encouraged with further" tagging" as will be seen in the following sections.

Starting IGD

Start IGD, via

[Windows 'Start']->All Programs->inx->Inxware->Inxware GUI Builder {Left Click}

IGD can also be started from IAB by clicking on the GUI entry in the project window. However this is host operating system dependent and is not recommended here to ensure the process completed as expected.

With no project open, IGD should look as shown in Figure 9.

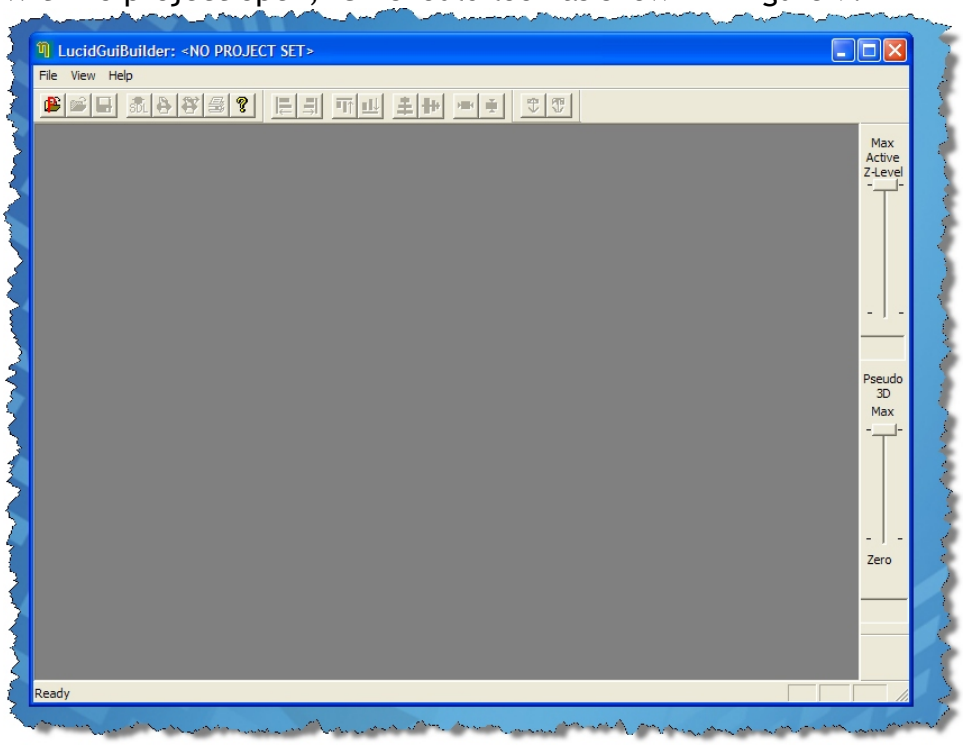

Figure 9: The IGD Tool

Open the Project

[IGD Main Menu] File->Open Project {Left Click}

The Project dialog should appear.

In the dialog, navigate to folder tutorial_1C, and open tutorial_1C.lpj.

NB: the '.lpj' (project) file serves as the 'openable' for both IAB and IGD.

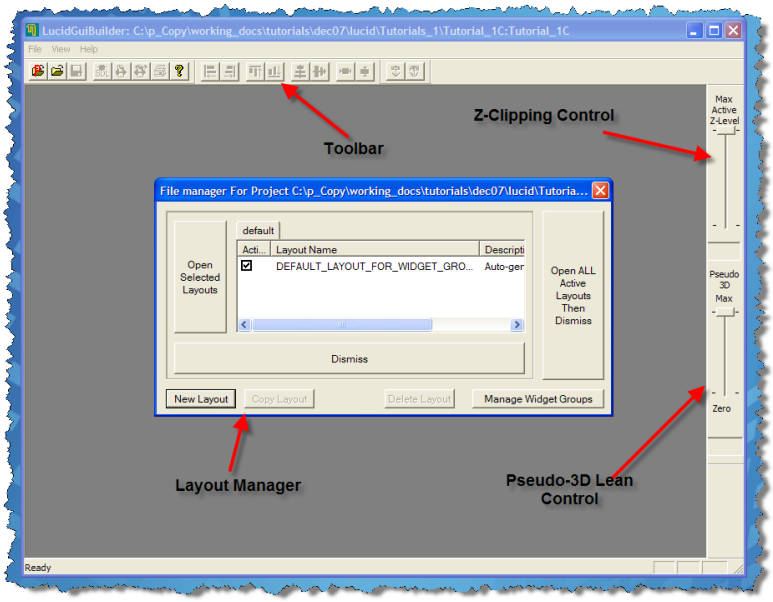

Figure 10: The IGD Tool with Layout manager open

The layout manager opens for the project, as shown in Figure 10. The Layout manager shows the layouts for the project, grouped by 'tabbed-window' into their respective widget-groups. (NB. Widget-groups are discussed later, in Tutorial). For now, we can open the one layout for the project as follows:

[Layout Manager: Open ALL Active Layouts Then Dismiss] {Left Click}

The dialog is dismissed, and a layout opens, similar to as shown in Figure 11.

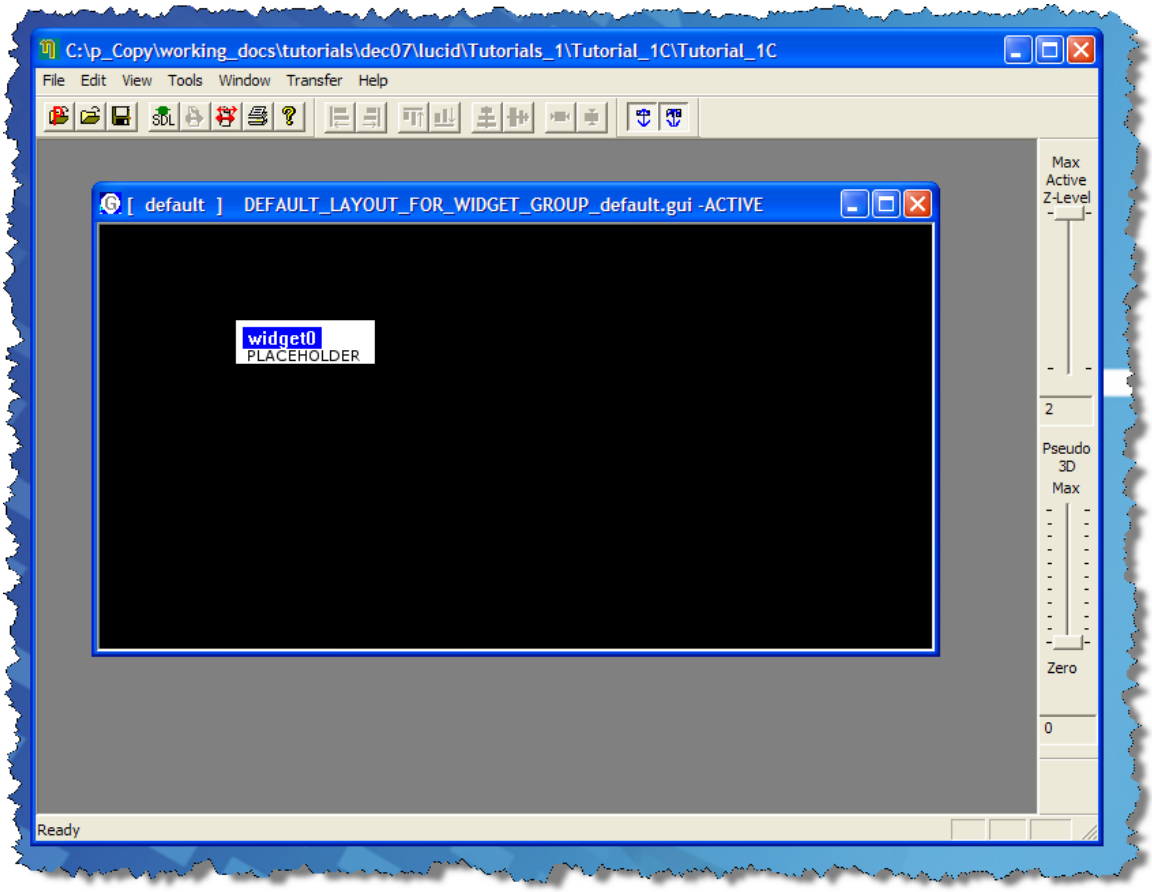

Figure 11: Tutorial 1 IGD window

Why is a 'placeholder' image displayed?

To make it easy for the user to 'get started' with IGD on a new project, IAB assigns a 'placeholder' image to each image function-block. In this way, when a layout file is opened for the first time, IGD can display at least something. This gives the user something to click on and edit!

Why can I sometimes not see some or all the placeholders in IGD?

When IAB writes-out the SODL file, ***it doesn't include any widgets that are not initialised****. A widget is deemed 'initialised' if its 'create' port either has a connection or is 'initialised'.*

Secondly, of the global set of widgets that have been created, a given layout shows only those that belong to its widget-group: Check this - there may be none!

If a placeholder is missing, check your IAB application to see if this is why.

Do I have to start afresh with placeholders in dummy positions each time I open a layout?

No .

Whenever IGD opens a layout, it checks the SODL for the list of widgets in the widget group, against its own list.

If a widget is in SODL and in the layout, it will be displayed in its edited form.

If a widget is in SODL and NOT in the layout, it will be displayed in 'placeholder' form.

If a widget is not in SODL and is in the layout, it will be deleted from the layout

Update SODL anytime

If there is no SODL file present at all

If, for this project, you had never written SODL from IAB, then the IGD window would have flashed a warning message to the effect that it couldn't read the SODL file, and subsequently would not have shown the 'dummy' widget.

If the SODL file is present, but out of date

In this case, you would have received no warning upon opening the layout. However, SODL can be brought up-to-date at any time simply by using IAB to write-out an (up-to-date) SODL file, then getting IGD to parse it, as follows:

[IAB:Main Menu]File->Write SODL {Left Click} (or use the shortcut)

[IGD:Main Menu] SODL->Load SODL {Left Click} (or use the shortcut)

This emphasises that IGD is the slave of IAB, and any time you add or delete gui widgets in IAB, you should write then read sodl with IAB and IGD respectively.

Select Bitmap for image.

Rather than the 'dummy' placeholder image, we wish to choose a more suitable image to be displayed by EHS for the function-block. To do this, we must first import the image into the project from an external location on the host's filesystem. Once imported this can the image can be associated with one or more GUI image widgets. To do this for the current widget:
Double-click the dummy image, or

[Rt-Clck]->Properties

A properties dialog appears, similar to as shown in Figure 12.

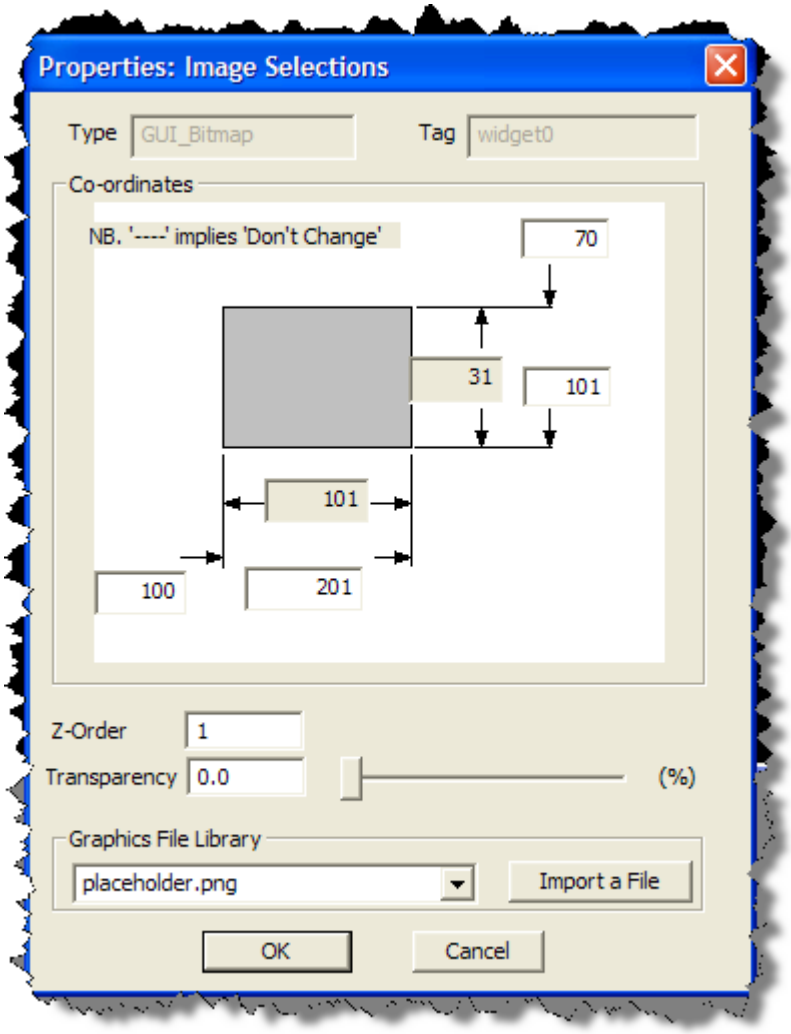

Figure 12: IGD Image-Properties Dialog

Click [Import a File] button

In the ensuing dialog, navigate to '[installation dir]\examples\exampleBitmaps',

select 'HelloWorld.png'

'OK' each dialog.

The IGD window should resemble Figure 13.

Figure 13: IGD Window showing bitmap

Tidying Up

Save your project in IAB

Save your project in IGD via:

[IGD: Main Menu] File->Save {Left Click}

then close each tool, via

[IAB: Main Menu] File->Exit {Left Click}

and similarly for IAB

Transferring the application to the target (Tutorial 1D)

Introduction

In this tutorial, we transfer our application to a target inxware runtime (EHS) running on the host PC. EHS for windows is provided with the inxware installation, but must be started independently form the windows application menu.

About EHS

The acronym 'EHS' stands for 'Event-handling system'.

Overall Functions

EHS has two overall functions, as follows:

Application runtime environment

EHS is the framework which runs on the target platform, and which follows the 'instructions' in the SODL file to effectively enact the IAB application.

Development Tool and Application update Communicator

EHS communicates (typically via TCP/IP) with IAB and IGD to:

- receive application data which are to be run, and also to
- communicate debugging information and status information to the tools.

Starting Windows EHS

To start EHS the desktop application launcher is used:

[Windows Start]->All Programs->inx->inxware->ehs {Left Click}.

A window for EHS instance should appear, which maybe blank if run for the first time. Otherwise the last application that was transferred to the EHS installation will be present and will start up.

Minimise the EHS window for the time being.

Start IAB, open project Tutorial_1C, and save it as Tutorial_1D.

Start IGD, and open project Tutorial_1D.

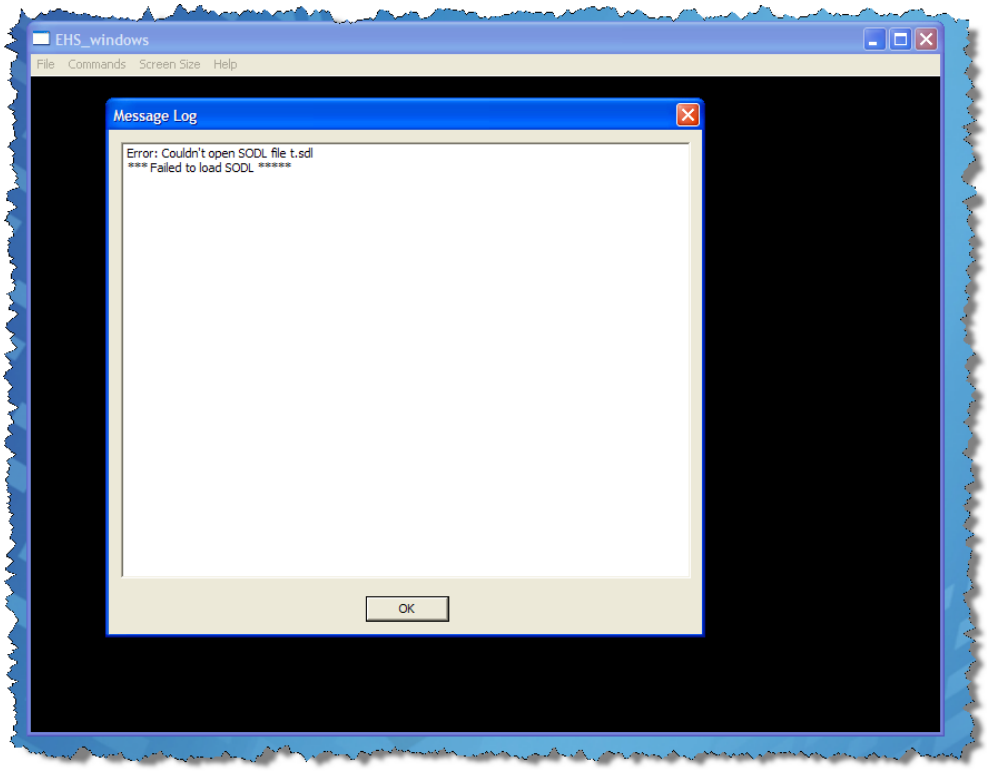

Figure 14:Windows EHS

Transfer Files to target

We will now transfer our application's files to the target. To do this, invoke the transfer process, as follows:

[IAB: Main Menu] Transfer->Transfer manage {Left click}, or

Left-click the button shown in Figure 15.

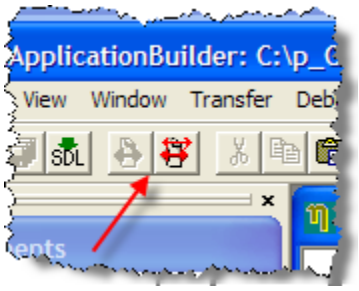

Figure 15:Transfer-To-Target Shortcut

The Transfer dialog (Figure 16) appears. The dialog first informs that the SODL file is being written, then that the files that need to be transferred are being listed. You should see the progress bars on the dialog reporting the transfer of the application files to the target.

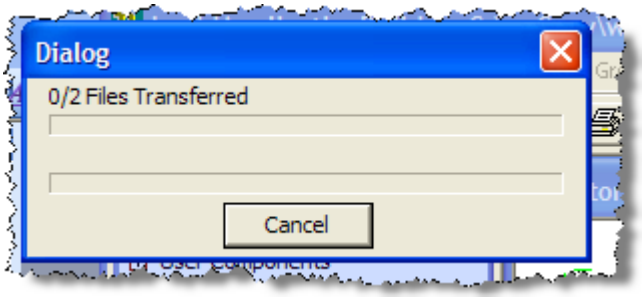

Figure 16: IAB Target-Transfer Dialog

About the Transfer Process and Dialog.

The transfer process can be invoked either to transfer all files, or just the files that have been changed. This choice is made at the time of starting the process.

The first time a project is opened, only the 'Transfer All' option will be available for the first transfer. This is to help Inxware ensure that all necessary files are sent to the target.

By the time the transfer dialog (belonging to IAB) has been displayed, the transfer process has already started. The dialog serves both to report the progress through the process, and to allow the user to cancel the process.

The appearance and behaviour of the transfer dialog is identical between IAB and IGD , even though each tool does technically have its own instance of the dialog.

See How it Runs.

Immediately following the transfer EHS will exit any currently running application and run the new application.

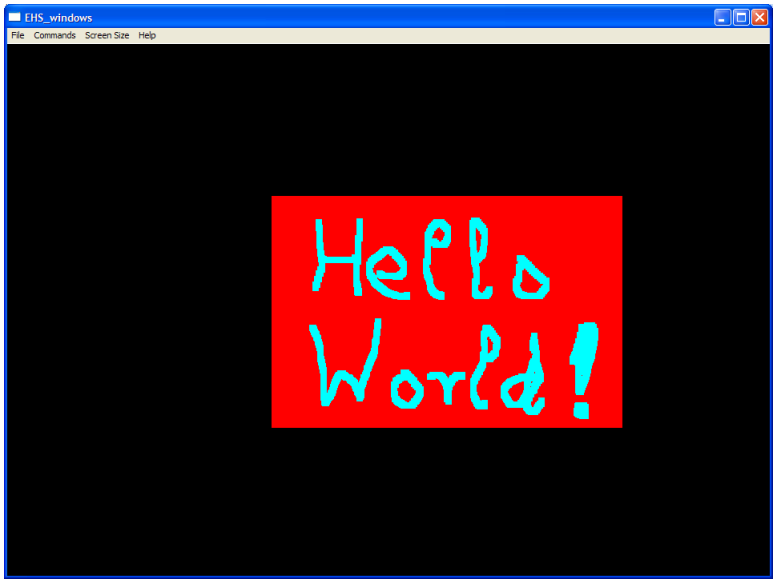

Figure 17: The Running ‘Hello World’ Application

Application Analysis

IAB Application

The OR block has two event inputs on its left one output port on its right side represented by black triangles. The OR event block is one of the simplest components of a small set of components that the developer uses to define the behaviour of an application. The function of this component is to assert an event on its output port when either of the inputs receive an event. So far we haven’t connected anything to the second event input port so this component merely passes on the initialisation event to the image display widget’s “show” port – i.e. the equivalent of initialising the “show” event directly.
It is generally not necessary to connect all ports of a component for it to operate.
This application also highlights the issue of race conditions in event-based system definitions. This example identifies a scenario where the timing of receiving the image widgets’s “create” port and “show” port events is not clearly determined. In general function blocks are defined to behave robustly when sequencing is not a concern functionally as in this case where the sequence of creating the function block and enabling it’s visibility is not important as long as the user understands that the behaviour defined for the particular component. In this case the GUI image is not displayed until both create and visibility events are received. In general components are defined using a “non-trusted client” model, where components can handle all combinations of use-cases robustly.
In this case there is no special event “logic” required to synchronise one event occurring before the other. Subsequent tutorials will identify cases where sequencing is important, and more explicit “wiring” of events maybe appropriate.

IGD Layout

The IGD layout allows interactive adjustment of the position and bitmaps of GUI components, and saves these details in a layout file. Try dragging the image around in IGD, then transfer to target using IGD’s transfer buttons (similarly to IAB’s). You should see the EHS display reflect the new position of the image in its display.

Tidying Up

In IAB, save the project as ‘Tutorial_1D’.
In IGD, Close the project, then close IGD itself.

Exploring the IAB Development Environment (Tutorial 2)

Introduction

This tutorial introduces the features of Inxware Application for editing project diagrams. This tutorial does not add any further functionality to the application, but provides tips on how to manage diagram components in the workspace

Start IAB.

Open project for tutorial_1D.

Save the project as ./Tutorial_2, i.e. at the same level as Tutorials_1.

Cursor Changes

Function Blocks

Move the cursor over a function-block.

The cursor changes into a black rectangle with shift arrows, indicating the function block can be moved to another position.

While over the function block, left-click and drag with the mouse.

Groups of function blocks can be selecte by dragging a rectangle around the groups of functions blocks. These can be moved “ensemble” , including connections or cut and paste as described later in this section.

Function Block Parameters

Each function block may have a set of configuration parameters declared in the component’s description file. IAB displays these along with user manual information by Double clicking the function block (or alternatively selecting properties from a right-click pull-down menu).
Parameters are primitive number, string or Boolean flag types and are pre-checked for range and format as specified in the component description file.

Line Segments

Move the cursor over a connecting-line segment – i.e. away from any corner in the line

The cursor changes into a cross pattern indicating the line can be moved orthogonally to its direction.

While over the line segment, left-click and drag with the mouse.

A new 'corner' appears in the line, whose position is dragged by the mouse.

While over the line segment, right-click.

A dialog pops-up, as show in Figure 18.

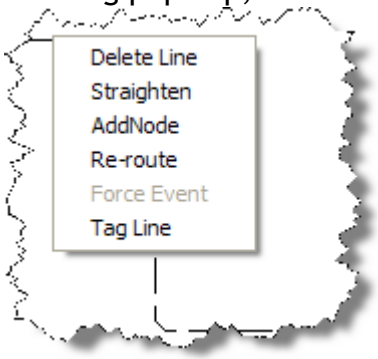

Figure 18: IAB Right-click popup
Delete Line deletes the connection.
Straighten renders the line straight
AddNode adds a corner (although this is done automatically by left-click / drag on a line segment).
Re-route recalculates a likely to be convenient path from port to port, via horizontal / vertical segments.
Tag Line replace the line with an implicit connection, implied between two ports which share a pair of tags.

Line Corners

Move the cursor over a connecting-line corner

The cursor changes into a 3-segment line with a 'blob'

While over the line corner, left-click and drag with the mouse.

The corner's position is dragged by the mouse.

Drag the corner so that the two line-segments forming it are roughly at right-angles.

A chamfer appears at the corner when a right-angle is approximated.

NB. The chamfer at corners helps to distinguish visually between a case where two lines cross-over and where they branch-away from being superimposed.

While over the line corner, right-click with the mouse.

A dialog pops up, as shown in Figure 19. This gives the option of deleting the node (ie the corner), in which case the two kinked line segments will become 1.

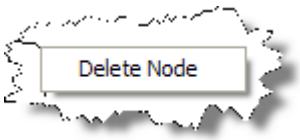

Figure 19: IAB 'Delete Node' popup

Ports

Move the cursor over a connection-port

The cursor changes into a black triangle

While over the port, left-click and drag with the mouse.

A connecting-line stretches between the port and the present cursor position.

While over the port, right-click with the mouse.

A pull down menu will appear (as described in tutorial 1 for initialising a port) , which will contain the option to tag a port. This allows a lines to be replaced with text-tags. This is useful if a port output is used in many parts of the diagram or when the connection must pass a long distance in the diagram and clarity of it's path is lost.

NB. Tagging should be used minimally - if your diagram is very complex then it should be decomposed into subsystems (see later tutorials on how to do this. It may indicate that your software structure requires an "architecture" review.

Copy, Cut, Paste, Delete

These have the customary effect windows effect, that is:
Copy copies the selected item(s) into the clipboard.
Cut removes the selected item from the application, and then copies the item into the clipboard.
Paste copies the item from the clipboard into the application
Delete simply removes the item from the application.
The shortcuts for Cut, Copy and Paste are Ctrl-X, Ctrl-C and Ctrl-V respectively.

Undo / Redo

Perform any edit such as creating or deleting a connection.

Click Edit->Undo (or Ctrl-Z)

Note that the action is undone.

Click Edit->Redo (or Ctrl-Y)

Note that the action is reinstated.

Connection Editing

This allows a connection to be moved from one set of input/output ports to another (ref Figure 20).

Move the cursor to the input or output end of connection 'A'.

Left-click and drag the connection to connection 'B', then release the mouse button.

Note that a dialog pops-up, that 'This is an illegal connection'. This is because the output (source) port is an even port, and the input (destination) port is of type integer.

Move the cursor to the input or output end of connection 'A'.

Left-click and drag the connection to connection 'C', then release the mouse button.

The connection is now associated with the new port.

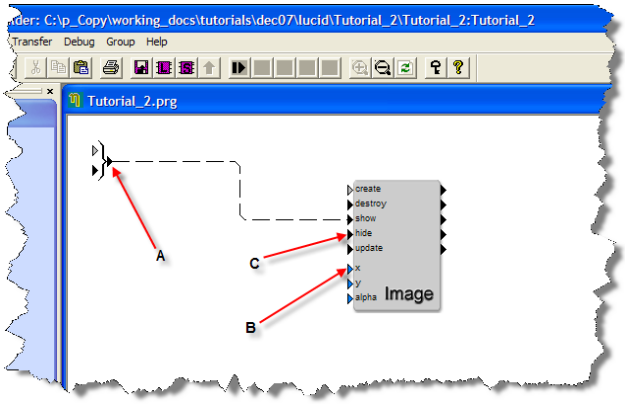

Figure 20: Demo-connection Ports

Component-substitution

Sometimes you may find that you would like to replace one function-block with another similar type. Examples include increasing the number of ports for scalable multi-input-output functions, upgrading previous version of native components with new versions.

To do this:

Move the cursor over to the component palette.

Drag an Event->Event OR-> OR 3, from the palette, and hover it over the OR2 already present.

The block already present (the OR2 in this case) becomes bordered with a green rectangle.

Release the left mouse-button

The function-block already present is substituted with the dragged block.

Drag an Event->Event OR-> OR2, from the palette, and hover it over the OR3 now present.

The block already present (the OR3 in this case) does not become bordered with a green rectangle. So, an OR2 can be substituted by an OR3, but not vice-versa. This is because an OR2 has fewer ports than an OR3, and the problem potentially arises as to how ***automatically*** to connect the OR3's possible 3 output connections to the OR2's to ports.
If in fact one wants to substitute (say) an OR 3 with an OR 2, one would do this manually, by dragging the OR 2 into the composition window, then dragging the connections from the OR 3 to the OR 2.

The project-overview window

The project-overview tree

Now that we have some data in the project, it is worth while examining the project-overview window. This is now populated with data that reflects the current project.

Firstly, expand the data-tree fully by clicking the '+' symbols on the tree. The categories on the tree are as follows:

APP

This contains the name of the current project and a tee identifying subsystems that maybe part of the project. Sub-systems can be opened directly from the tree view by double clicking on the navigated branch.

GUI

This contains a list of layouts for the project. Layouts are what IGD deals with: they are used to define the position of layouts on the screen. IGD can be launched on windows operating system directly from the navigated branch.

PNG

This list the image files (these must be of type 'PNG') that exist in the project. (Known bug: Other formats are supported, but are not visible in this window).

TXT

This lists text files that have been added to the project. Text files are stored by Inxware, and sent to the target when the user requests this, but are otherwise not processed by Inxware.

Hiding the Component and Project Windows

Firstly, and temporarily, we will pretend that we need more space for our project window. In the top-right corner of the Component-palette there is a standard windows 'Dismiss' ('x') symbol.

[Lft-Clk] the Dismiss button of the Component-palette window.

Repeat this for the project-overview window.

Note that the windows are dismissed. This leaves more space for the project window.

To re-instate them use:

IAB: [View]->Project Bar

IAB: [View]->Component Palette

Note that windows are restored.

Opening a data file from the project overview

Start IGD.

Note that IGD has no project set.

[Double-Clk] the GUI->'DEFAULT_LAYOUT_FOR_WIDGET_GROUP_default'

Note that IGD is now set to the current project, and displays the current layouts.

Non-IGD applications

Double-clicking any icon in the project-overview tree will invoke an instance of the application associated (in your PC) with that icon type.

IGD

In particular, double-clicking a 'layout' icon will:

Bring an instance of IGD that is already running the current project to the foreground, or, if there is no such instance, will

Open a new instance of IGD, having initialised it by opening that project.

Project Resource-Locking

There are certain project resources that are read-from and written-to by both IAB and IGD. During critical operations only one instance of an Inxware tool can have control of these resources at one time. Collisions do not occur in normal use-cases of inxware, but should a warning be issues this section explains the causes and remedies.

Start IGD, and open project Tutorial_2.

Note that there is no problem in general having IAB and IGD edit the same project!

IAB: double-click an image function block, and from the 'Properties' dialog, invoke the Widget-group manager dialog.

IGD: Invoke the layout manager, and attempt to invoke the widget-group manager.

Note that Inxware will not allow simultaneous access to this dialog from more than one instance of a tool.

IAB: Close the Widget-group manager dialog

IGD: Invoke the layout manager, and attempt to invoke the widget-group manager.

Note that Inxware now allows access to this dialog.

Tools->Options->Native View-background

toggles the colour of the display background between black and gray.
NB. The idea of this is that the user may wish to see the widgets against target background (black) or against gray (while he edits the layout with IGD).

Adding Functionality to “Hello World” (Tutorial 3)

Introduction

This tutorial shows further use of a few more basic Inxware components to enable animation of the image’s location. The implementation of the required functionality is not the most elegant solution available in inxware, but shows how simple components can be combined to implement more complex behaviour. The application will allow for user input from the keyboard that will move the displated image in the x, y directions on screen.

Adding further Components

Open the application built in tutorial 1 or Create a new project called Tutorial_3.

Add components to your application as shown in Figure.

The four components represented by the black and white icons with a single event input, single Boolean input and two event outputs in the diagram or “Bool Control” components from the “event” menu. Searching for these components will give you a view of the components available in the component menu. ***Connect the components as shown.***

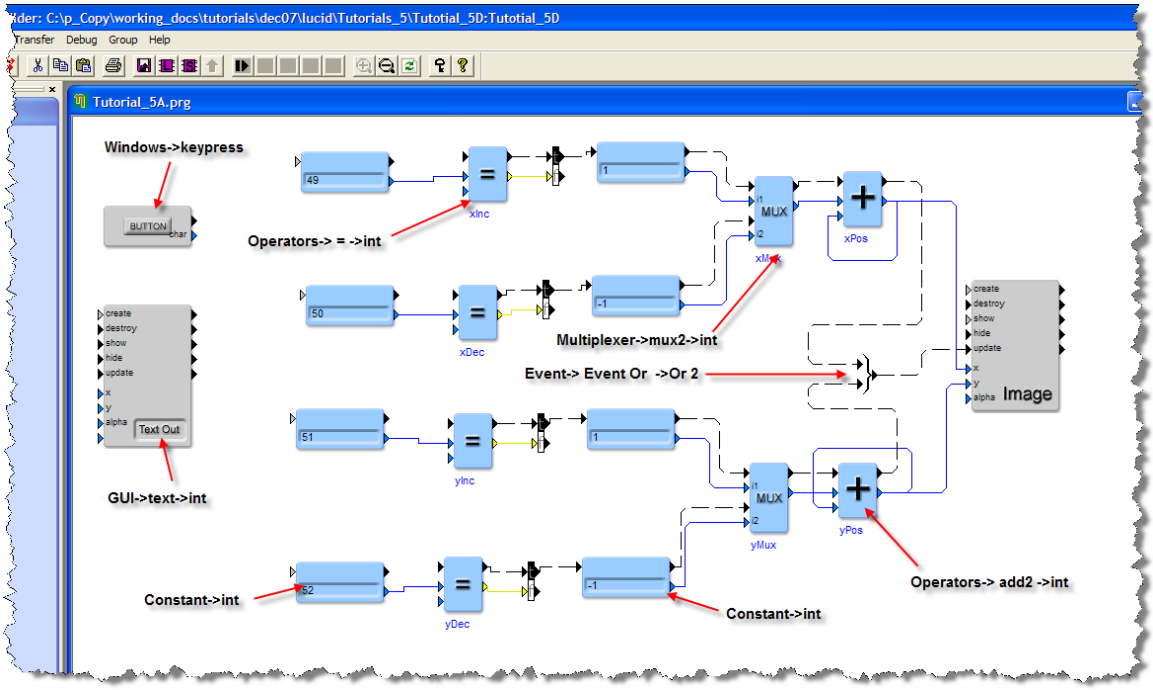

Figure 22: Tutorial_3 application: (state ‘A’)

Working from left to right, components are as follows:

Components->Windows->Keypress

Components ->GUI->GUI_Output->int

Components ->Constant->Int ***(x4)***

Components ->Operators-> = -> Int ***(x4)***

Components ->Multiplexer->Mux2->int ***(x2)***

Components ->Constant->Int ***(x4)***

Components ->Operators-> Add2 -> Int ***(x2)***

Components ->Event->Event Or->OR2

Components ->GUI>GUI_Image

When the components have been added, double-click each one with a tag-name, and set the tag as shown in the figure.

Create some 'Tags'

Firstly, connect the button (top-left of the screen) to the int '=' function-block as shown in Figure 21.

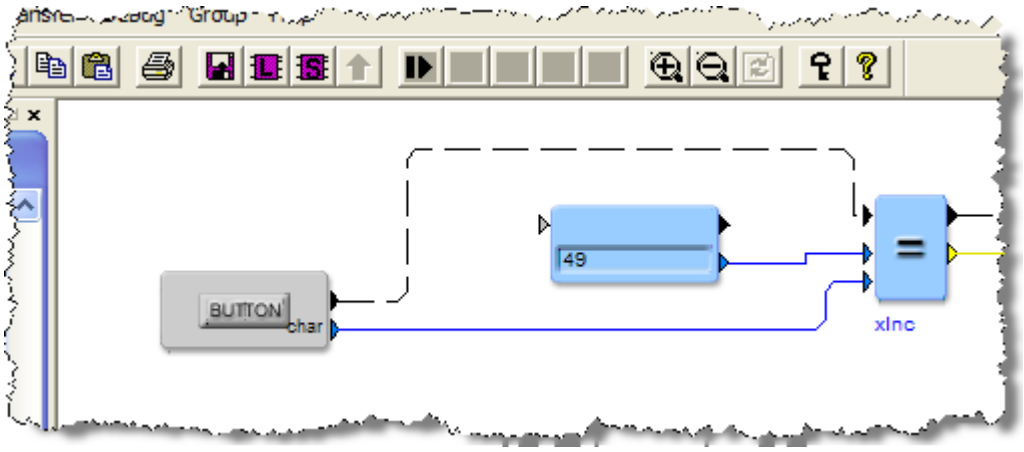

Figure 21: Tutorial_3 application part-view: (state 'B')

Right-click on the event output port of the keyPress component, and click 'Add tag'.

Enter 'keyPress' into the tagname field

Right-click on the blue (signifying 'integer') connection between the button and the 'xInc' function block, and click 'Tag Line'.

Enter 'keyVal' for the tag name.

This part of the application should now look as shown in Figure 22.
[NOTE: in later versions of inxware the input event port for constant value components has been deprecated. The adjustment to this tutorial is to make the relevant connection to the multiplexer directly. You may also find further ports on the image and test functions blocks, which can be ignored.]

A port that is tagged is deemed to be connected - albeit with an 'invisible' connector - to the output port with a tag of the same name.

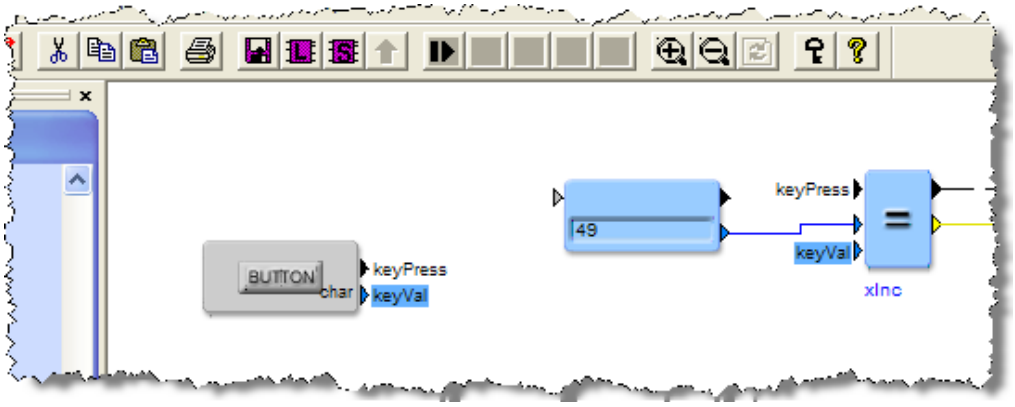

Figure 22: Tutorial_3 application part-view: (state 'C')

Then right-click one input-port of each comparator-block, select 'add tag', and choose 'keyVal' in the drop-down menu.
On the GUI->Text and the 4 function-blocks 'xInc', 'xDec', 'yInc', 'yDec',
Right-click->Add tag,
and set the tags as shown in Figure 23.

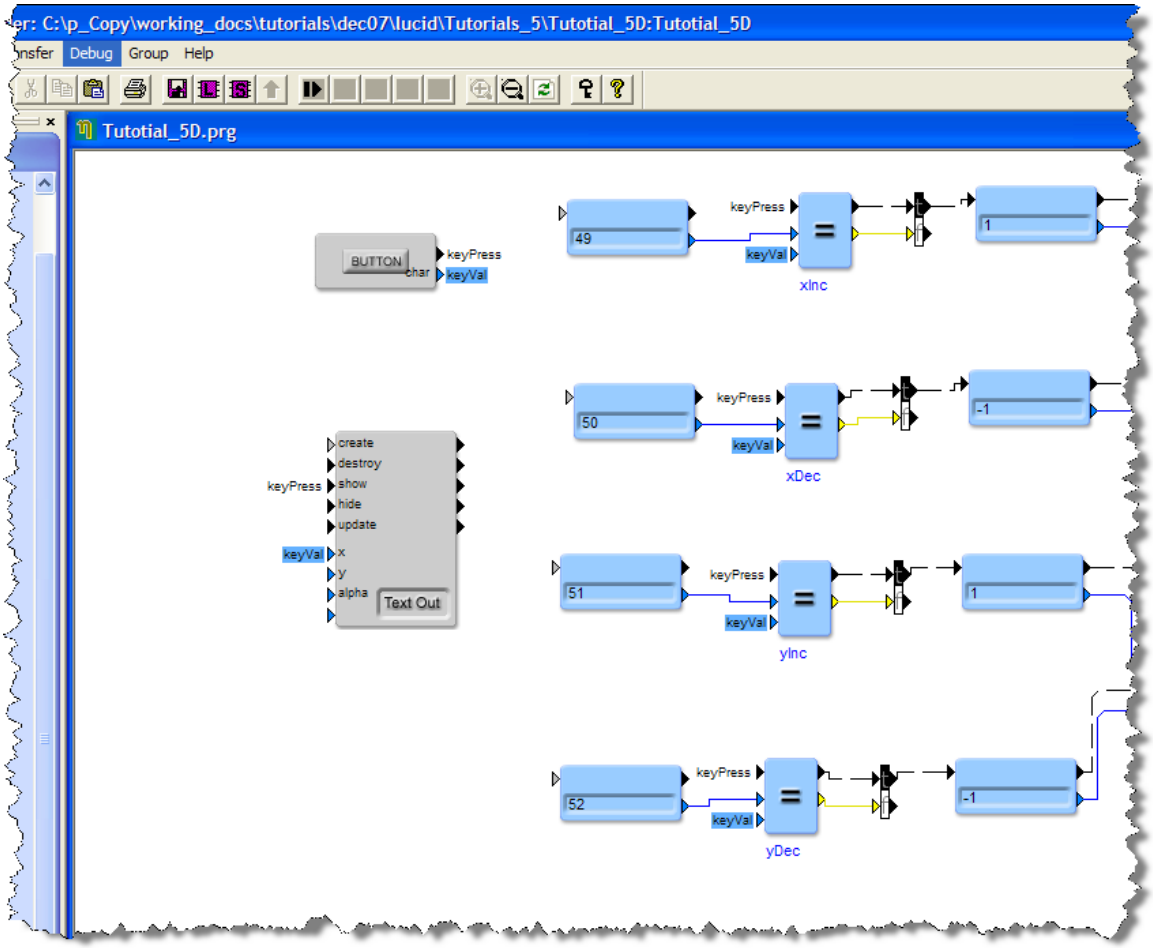

Figure 23: Tutorial_3 application part-view: (state 'D')

Initialise the layout, Transfer to Target

When all connections are made

write SODL in IAB and start IGD to place the new Textbox GUI widget as in Tutorial 1.

Ensure EHS is running.

In either IAB or IGD, transfer the application to the target.

Test Behaviour

With the EHS window focused, hold-down for a few seconds at a time, individually, any of the keys '1', '2', '3', '4'.

Note that the text widget gets updated by the 'value' of the key pressed, and the image widget moves around the screen.

Application Analysis

The implementation of this application uses low level simple operations to decode keyboard events, trigger specific values to be selected and passed to a moveable GUI widget.
The constant integer values of 49 – 52 are the ASCII key values used, together with the integer comparators, to filter key press events: only if the key press value is equal to the constant int will the key press event pass through the Boolean control.
Whichever Boolean control passes the event on to its output, this also passes its event into the 1 or -1 constant, and thence into the multiplexer. Thus for example if:
key '1' is pressed (value 49) the value +1 passes to the output port of the upper multiplexer.
If key '2' is pressed (value 50) the value -1 passes to the output port of the upper multiplexer, and so on.
Thus the value of the addition block for x or y is either incremented or decremented according to which key was pressed. These values are used to control the position of the image on the screen.
This implementation could alternatively be implemented using a switch construct or index selector function block.

Debugging (Tutorial 4)

Applications running on a remote device or locally on a the workstation can be debugged over the same network connection that is used to deploy software to the devices and runtimes. Debugging maybe required if the application is not functioning as expected or to aid understanding of how an application operates in its real-time environment.

Debugging is carried out graphically, where data values can be displayed as an overlay on the data connections, and the occurrence of events. Time stamped data can be logged and visualised using 3rd –party tools such as Time Doctor, however this is rarely necessary.

Start debugger

Start the debugging tool by clicking the 'Start' shortcut (ref Figure 24).

The 'Start debugger' icon becomes unavailable. The 'Stop' and 'Pause' icons become available (Figure 25).

Pause debugger

Pause the debugger, by clicking on the 'Pause' icon.

The 'step' and 'continue' icons become available.

Generate some events

Click on the EHS window to give it the focus.

Press once any one of keys 1 – 4.

Step through the event processing

Click the 'step' shortcut at discrete intervals, giving yourself a moment between clicks to peruse the application.

Note that the connection lines change colour at intervals. This is to indicate that they are processing an event.
Note also that the yellow framed values over-layed on input ports are updated dynamically as they change.

Continue

At any stage after pressing a given key, you may decide you have ‘seen enough’. In this case, pressing the ‘continue’ icon tells the debugger to process all pending events.

Press ‘Continue’

Press ‘Pause’

Give EHS the focus, and press another key.

Step through to see events following from this key press handled, as before.

Exiting the debugger

At any time you can stop the debugger by clicking the ‘Stop’ icon.

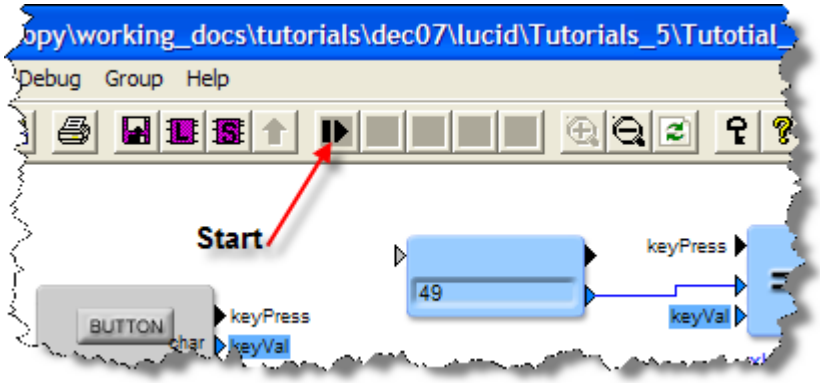

Figure 24: ‘Start Debugger’ shortcut: (Initial Menu)

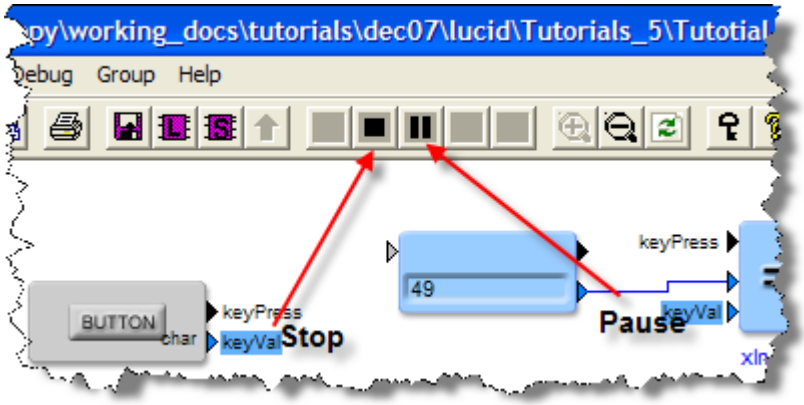

Figure 25: ‘Start Debugger’ shortcut: ‘Started’ Menu

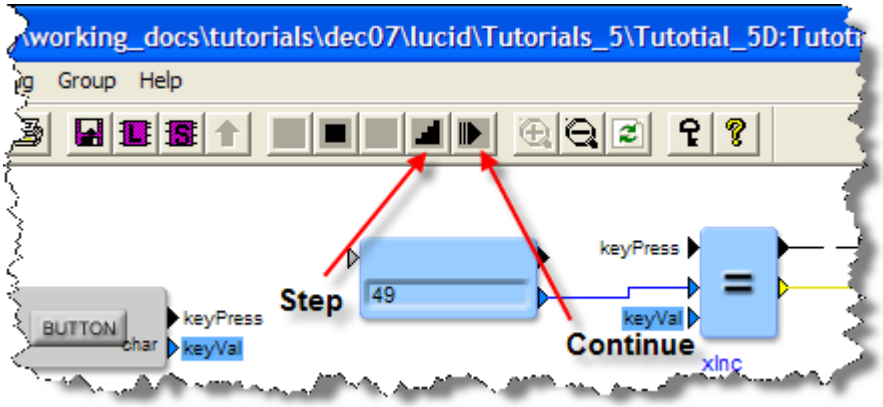

Figure 26: 'Start Debugger' shortcut: 'Paused' Menu

Encapsulation (Tutorial 5)

Introduction

This tutorial introduces ‘encapsulation’ of groups of function blocks into a single function block with it’s own ports indicating what data and events the sub-system requires and makes available. Encapsulation is an important part of inxware application development for more complex applications than shown in these tutorials. The times when encapsulation is required are typically:

- the complication of the inner details are abstracted away from the user, and:
- re-use is facilitated by creating re-usable and inter-project sharable components.

The simplest form of encapsulation subsumes the set of target function blocks into a single function block, but does not create an entry in the component chooser menu, so that it can be easily u re-used elsewhere (though any function block can be copied and pasted multiple times). The next level is a light variation where the subsystem is created and simultaneously entered into the component chooser library. Any subsystem can however be added to the component chooser’s library at a later date.

Build Application

Open project Tutorial_5

The composition window appears as in Figure 27.

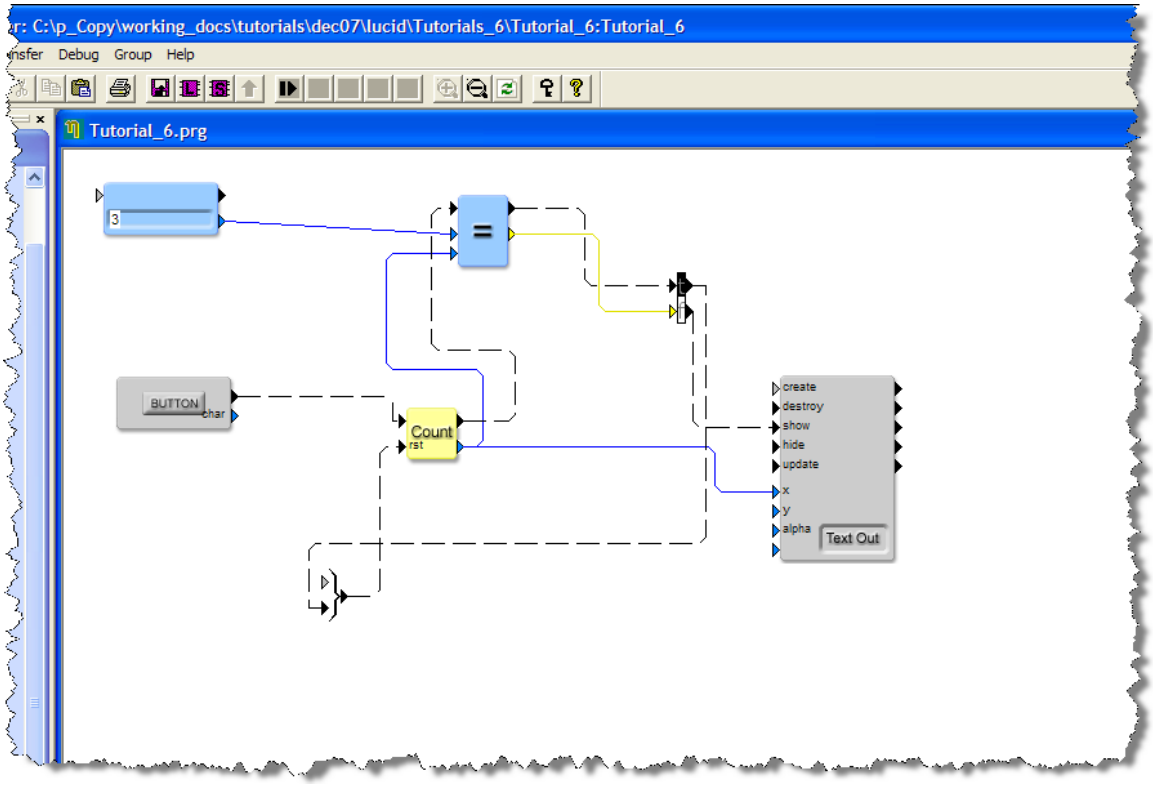

Figure 27: Tutorial 6 Application (state 'A')

File->Save Project As->tutorial_6B

File->Save Project As->tutorial_6A

This saves the present project for future uses, and allows us to carry on with the current status as tutorial_6A.

Create a Simple Sub-system

Click and drag a rectangle around the four central components, as shown in Figure 28.

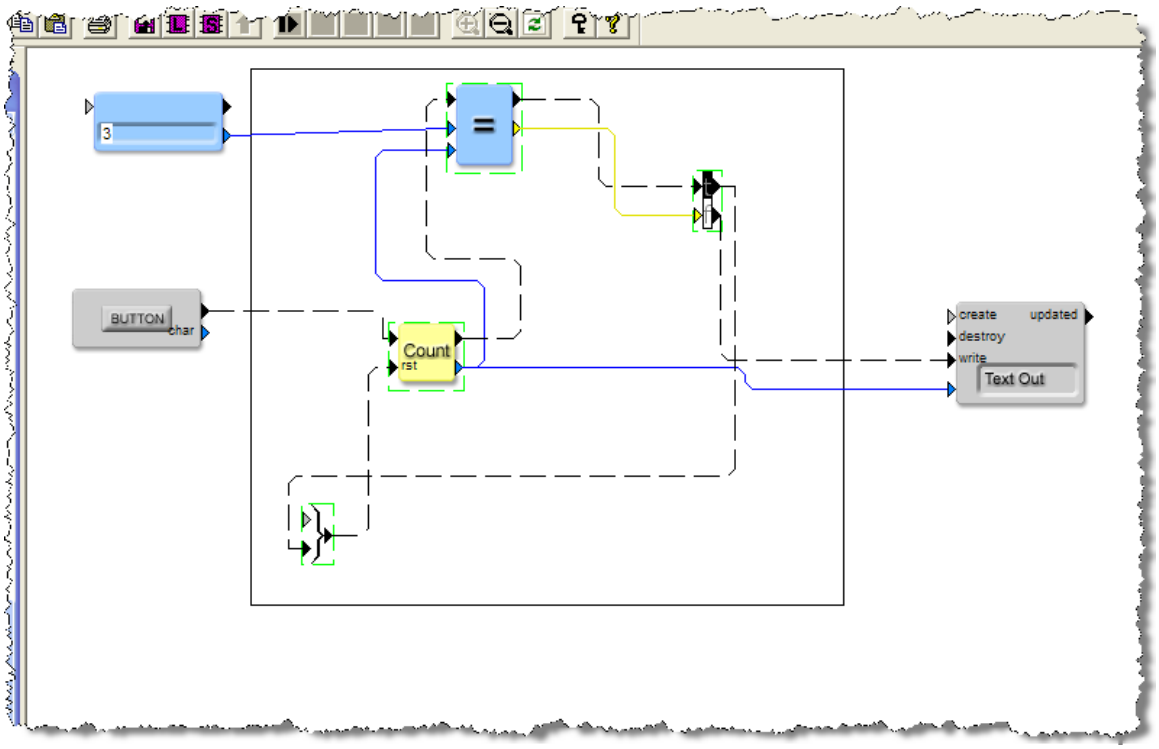

Figure 28: Central components selected

File->Encapsulate->Sub Block

A dialog appears as shown in Figure 29.

Fill-in details as shown (if you can -don't worry if this is not possible at this stage because of multiple ports, labelling can be completed later!), then [OK] the dialog to create the encapsulated function block in the workspace.

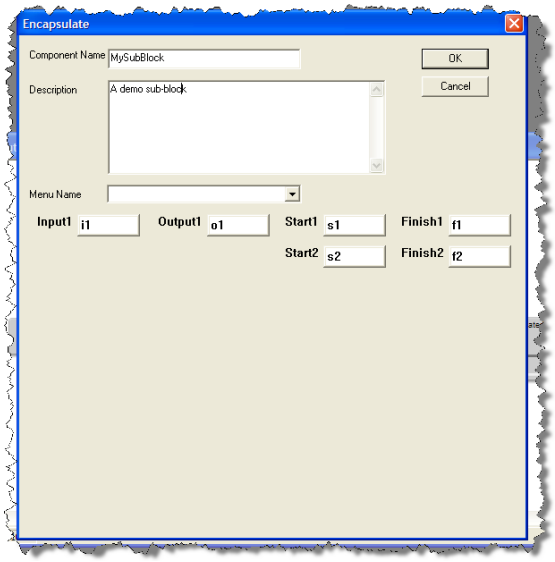

Figure 29: Encapsulation Dialog

The project window should look similar to that shown in Figure 30. If necessary, right-click the purple block and select 'Reroute All' to orthogonalise the connecting lines.

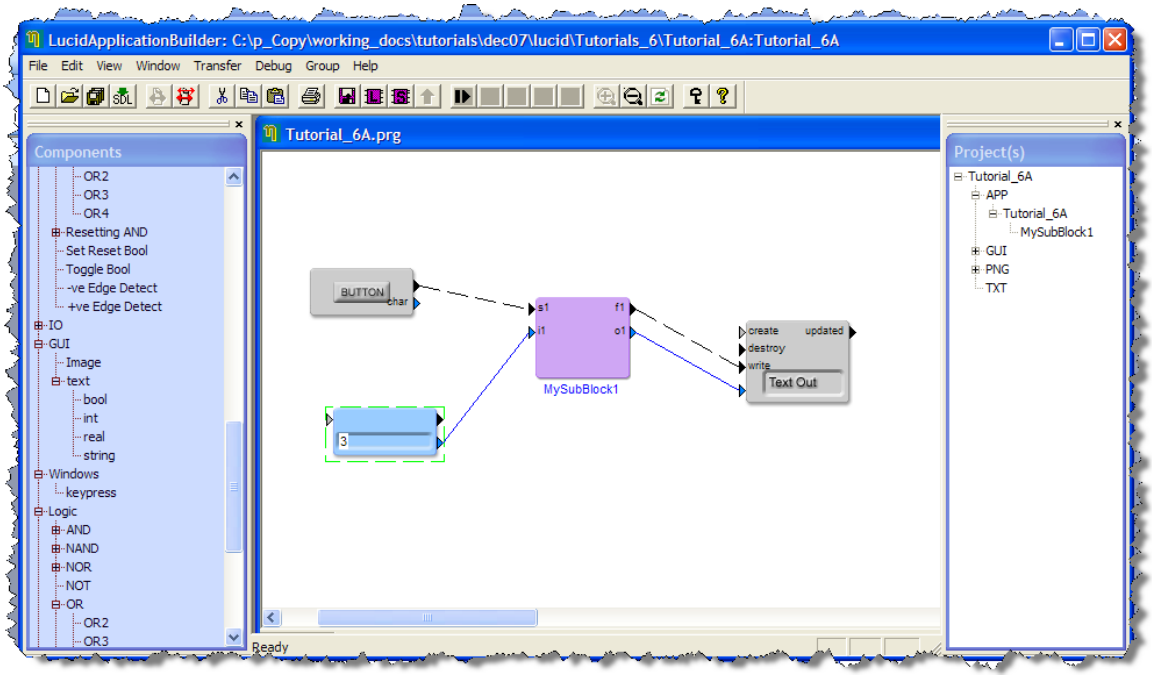

Figure 30: Tutorial 6 Sub-block Encapsulation

Double-click the purple encapsulation block.

Note that it should expand to reveal its contents as shown in Figure 31.

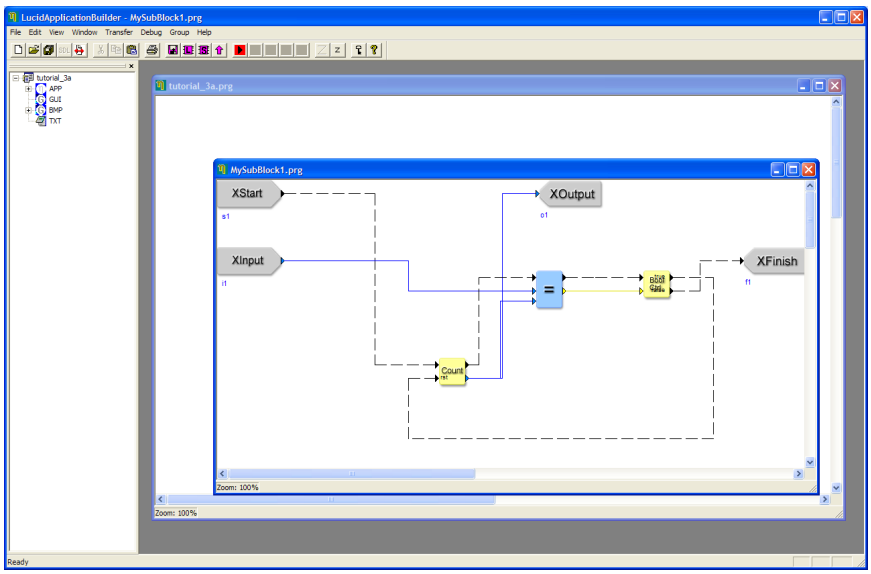

Figure 31: Tutorial 6 Encapsulation - expanded

Click anywhere in the 'tutorial_6' project window.

Note that the expanded function block should contract back into a purple block.

It is now most convenient to label the ports by right clicking on each port and selecting to change the port name from the pull down menu. Alternatively you can enter the encapsulated block and label each function

block by right clicking on each "x-port" icon that represents entry and exit ports of the susbsystem.

RtClk in the component chooser window to invoke the component palette, and expand 'User Components'.

Note that 'MySubBlock' is NOT present as this component was created as a single instance.

Creating a Subsystem for inter/intra Project Re-Use

File->Close project

File->Open Project->Tutorial_5B

File->Encapsulate->Library (subset)

This achieves a similar effect to the 'sub-block' option. however, this encapsulation will be available to other projects, as can be found by:

RtClk in the project window to invoke the component palette, and expand 'User Components'.

Note that 'MySubBlock' IS listed in this case.

Advanced users can export the subsystems as packages to be installed in pier's copies of inxware tools.

Creating an Empty Subsystem Block to be Populated

When applications are developed "top-down" or the user wishes to create a new composite component with a specific interface rather than an application The same result as the previous bottom-up methods may be achieved by creating an empty function block with pre-defined ports that need subsequent connection. This also be useful where a place-holder for a piece of functionality is required, but the developer is not ready to implement this immediately.

This is achieved using "File->encapsulation->sub-block" menu or shortcut button which presents a dialog box that the developer can use to define the top level naming and a description.

Ports are added later either externally by right clicking on the new function block's body and selecting "Add Port" or internally by dragging the appropriate port type from the component chooser under "Software Structure->Function Blocks".

Further GUI Development (Tutorial 6)

Introduction

This tutorial introduces the features of Inxware GUI Builder that make it more easily usable, and more productive.

Moving / Resizing – By Cursor-Drag

Start IAB.

Open project 'Tutorial_6A',(under a folder 'Tutorials_6').

Write SODL

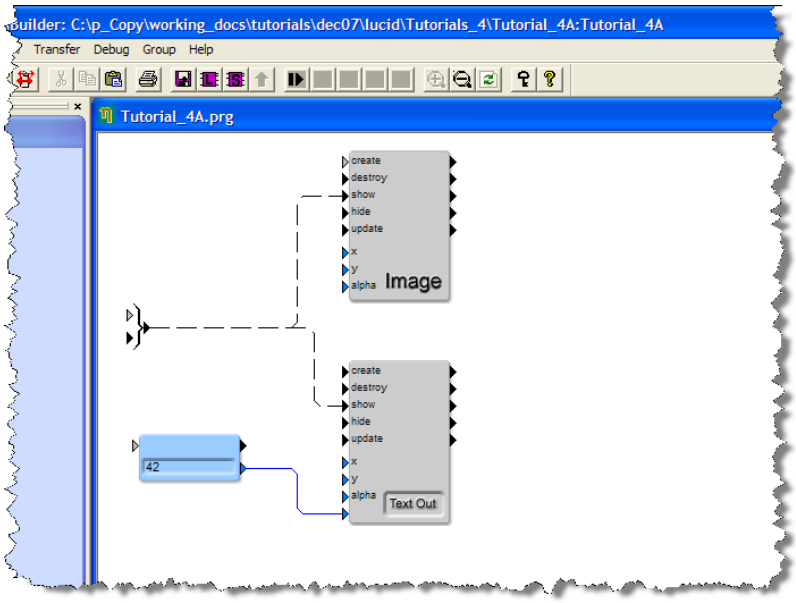

Figure 32: Composition

Textbox Icons

Start IGD.

Open the default layout

Select a text widget

Move the cursor over and around the text-widget.

Take care to move the cursor around the edges and corners of the widget, as well as the interior.

The cursor changes into a NS , EW, NESW, NWSE [1] as it is moved around the periphery.
The cursor changes into a 'All Directions' (a vertical / horizontal cross) when over the interior of the widget.

With the cursor over various regions (interior, top, etc) of the widget, left-click and drag with the mouse.

Dragging an edge or corner of the widget resizes it. Dragging the interior moves it.

Image Icons

Move the cursor over and around the image widget.

Take care to move the cursor around the edges and corners of the widget, as well as the interior.

Note that:
The cursor changes into a 'No Entry' sign as it is moved around the periphery.
The cursor changes into a 'All Directions' (a vertical / horizontal cross) when over the interior of the widget.
Dragging the interior of the widget moves it.

Image vs. text Icons

An image widget takes its size from the image it represents. Changing the image automatically resizes the image: it would be meaningless for the user to be able to change the size.

A text widget is resizable at the user's whim (notwithstanding considerations of whether there is enough space for the text to be displayed!).

Hence the difference in the behaviour.

Moving / Resizing – By Dialog

A text widget can be moved and/or resized, Image widgets just moved, by opening its 'Properties' dialog, via a:

double left-click, or by:

Right-click->Properties

Some important things to note:

[1] NS = 'North-South', NESW = 'North-East / South-West', etc

For a text-widget dialog:

Setting the left-edge position updates the right-edge to keep the width the same.

Setting the width sets the right-hand position to keep the left-hand edge position the same.

Similarly with bottom edge, vertical extent and top edge position.

For an image-widget dialog

The width and vertical-extent fields are non-editable, but populated with reference values for convenience.

For multiple-selections.

A selection of more than one text widget can be group-edited.

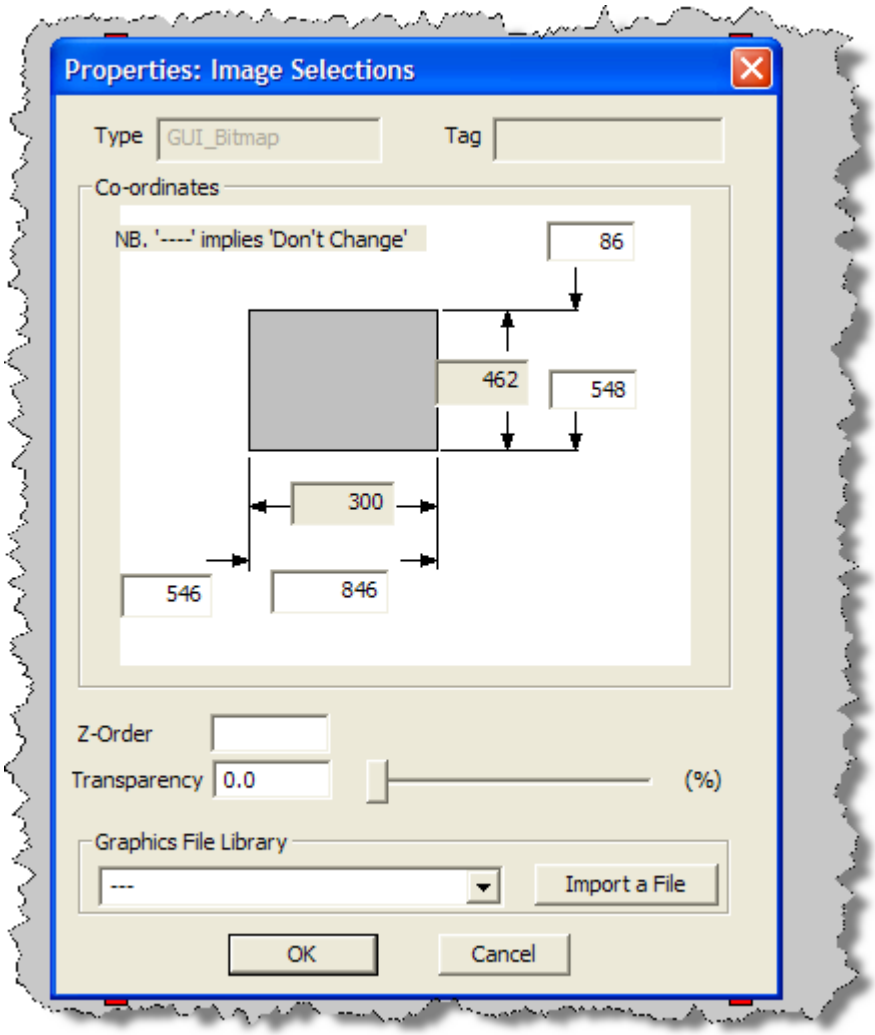

Figure 33: Text Icon Properties Dialog

With reference to Figure 33:

Any field for which all the widgets share a given value will be populated with that value. In this case, all the edge and width values are the same for each widget.

Any field for which the widgets do not all share a given value will be populated with '---'. For example, all the widgets have a different image file, so the 'image file' field will have a '---' field.

If such a field is edited with a non '---' value, all widgets will be updated to that value. In this case, if the image file drop-down were to be set to 'jack-hearts.png', all the widgets would be updated (upon OK) to show that image.

For any field with a '---' value, all widgets will not have that attribute updated.

***NB**.If you edit a '---' field with a value of (say) 301, then realise that you want all the selected widgets to keep their value for this field, then you can type '---' **by hand** into the field of the dialog.*

Z-ordering

Introduction

Z-ordering allows you to control which widgets appear above other widgets. Although any widget is of course drawn (physically) 'on the screen', an widget at a higher z-level will be drawn over (i.e. will obscure) another lower widget, if and when they overlap.

Hands On!

Close the present tutorial ([IAB]File->Close Project)

Open tutorial 'Tutorial_6B' ([IAB]File->Open Project{Navigate to tutorial_4B, double-click '.lpj' file})

Z- clipping control

Ensure the z-clip slider is at max z.

Slide the z-clip control up / down

Note the following:
Upper widgets progressively disappear as the slider is moved down, and vice-versa.
There are 'notches' on the slider corresponding to the z-levels where widgets actually are.

Pseudo-3D Control

Leave the z-clip control at max z.

Slide the z-lean control up / down.

The widgets move progressively into pseudo-3D positions, and back again.

The slider controls only the display of the widgets: they can be redisplayed simply by adjustment of the slider.

Task

We will now arrange the images in their correct order, that is, with the King uppermost, then the Queen, and so on.

Slide the z-clip control down until the King appears on top of the display.

Rt-click-> Properties the King widget, and set the z- level to 20.

Slide the z-clip control up to 'max.

The King appears on top of the display.

Slide the z-clip control down until the Queen appears on top of the display.

Rt-click-> Properties the Queen widget, and set the z-level to 19.

Slide the z-clip control up to 'max'.

The King appears on top of the display.

Slide the z-clip control to 1 below 'max'.

The Queen appears on top of the display.

Repeat this cycle until all the cards are correctly z-ordered.

Note that the z-level of each widget is marked with a 'tick' mark on the z-level slider.

Edit the King and Queen widget to have a z-level of 30, 29 respectively.

Note that the tick marks show the gap in z-level now present

Z-Reordering

Tools->Z-Ordering->Renumber (Consecutively)
to renumber all z-levels starting from 0

Tools->Z-Ordering->Renumber (keeping any gaps)

to renumbers all z-levels starting from 0, and compresses any gap in the ordering of more than 1 down to 1.

Eg:
Z values of:
2,3,4, 11,12, 20, 30,31
would be renumbered to:
0,1,2, 4,5, 7, 9,10.

Gaps in z-ordering

A good thing about this is that the user can form logical groups of widgets, each group separated by a gap.

Note that the z-lean is calculated as if the max gap allowed is 1: this allows the user to set a max z-level of (say) 1000 without having that widget's z-lean become over-exaggerated.

Undo / Redo

Perform any edit such as moving a widget.

Click Edit->Undo

Note that the action is undone.

Click Edit->Redo

Note that the action is reinstated.

NB. *The shortcuts for undo, redo are Ctrl-Z, Ctrl-Y respectively.*

Multiple selections

Introduction

IGD allows the user to control which items are selected, and which is the 'master' selection. The significance of the 'master' selection will be described later.

Scramble the Icons

Set the z-clipping control to max z.

Drag the top widget away from the stack of widgets, into its own position.

Repeat this for the next widget of the stack, so that no widgets overlap each other's position.

Pick a Master

left-click any widget

Note that it gains a border with solid red rectangles on the corners. These solid red handles indicate that it is the 'master' selection.

Reset the Master

left-click any other widget

Note that the widget just clicked becomes the (master) selection, whilst the formerly-selected widget becomes deselected.

Unselect everything

left-click empty space

Note that all items become de-selected.

Select Master and Slaves

left-click an widget

Ctrl-left-click several further (as yet unselected) widgets; note which widget was clicked subsequent to the master.

Note that selections subsequent to the first (master) gain a border with hollow red rectangles on the corners. These hollowness of these red handles indicate that they are 'non-master' (ie slave) selections.

Toggle a non-master selection

Ctrl-left-click the last widget selected.

Note that it becomes deselected

Toggle the master selection

Ctrl-left-click the master widget.

Note that the master becomes deselected, and that 'master' status is bestowed upon the widget which was the 2nd to be selected.

Group-select – 'Completely within'

Left-click on space, and drag a marquee around a selection of widgets

Ensuring that at least one widget is only partially within the marquee, release the left mouse button to the lower-right of the start point

Note that widgets completely within the marquee get selected.

Group-select – 'At least partially within'

Ctrl-left-click on space, and drag a marquee around a selection of widgets

Ensuring that at least one widget is only partially within the marquee, release the left mouse button to the lower-right of the start point

Note that widgets that are either completely or partially within the marquee get selected.

Selection Toggling

Note that repeating and with the Ctrl key pressed renders a similar result, but that the selections are toggled, rather than set outright.

Alignment

IGD allows the user to align widget edges in a left-right and up/down sense. Note that the slave selections are aligned with the master.

Select several widgets, noting which is the master.

click Tools->Alignment, to bring up the options shown in Figure 34.

Experiment with these options (which are hopefully self-explanatory).

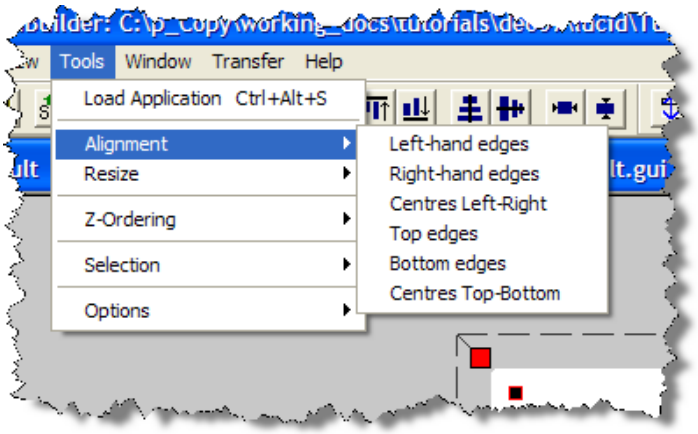

Figure 34: Icon Alignment Menu

Comparative Resizing

Introduction

IGD allows the user to set the size of a given widget or widgets to be the same as a user-nominated 'master' widget.
It should be born in mind, however, that resizing works only for text widgets: image widgets can never be resized - except by changing the image they represent.

Tools->Resize brings up the dialog shown in Figure 35

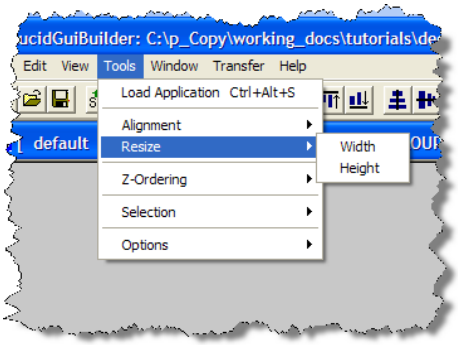

Figure 35: Icon Resize Menu

Miscellaneous

Tools->Options->Display Widget Labels

toggles display of these on/off.

Advanced GUIs (Tutorial 7)

Introduction

This tutorial introduces the concept of widget-groups that help developers deal with UI widgets in subsets in IGD, where otherwise clutter would hinder the layout process. This section also indicates the ability to produce multiple layouts "Skins" for a UI to suite different target types, products or deployments.

Overview

A widget-group is a group of widgets! The significance of widget-groups is as follows:

- Any GUI layout must (upon creation, by the rules of Inxware) be associated with a given widget-group as chosen by the user.
- Thereafter, that layout displays (and hence can edit the details of) only the widgets of that widget-group with which it is associated.

An Example Scenario

Suppose that in a digital TV application, you have a 'home' screen, with options on that 'home' screen to invoke a 'tuning' screen. In that case, you might create a 'home' and a 'tuning' widget group. Then, you could create a layout for each widget-group, and when editing the (say) 'tuning' layout, be able to concentrate on *its* widgets, without being cluttered by 'home' widgets. Note that you can the example in this tutorial is very simple, NOT a TV-tuning screen!

Many layouts-per-widget-group allowed

Note furthermore that you can have more than 1 layout for a given widget-group, e.g. 'My_home_4-3', 'My_home_WideScreen' might be names for layouts of the 'home' widget-group – with the 'home' in the layout-name acting as a clue to the widget-group.

Setup the project

Set up IAB Components

We will create two widget-groups, with each group containing at first one widget.

Start IAB, opening a new project called 'Tutorial_3', at the same level as 'Tutorial_2'

Drag an image function-block into the composition window.

Left-Click the image function-block, press Ctrl-C to copy it, then Ctrl-V to paste another instance into the project window.

A dialog will appear after the paste, asking which widget-group you want the pasted widget to be allocated to. ***Select 'default' from the drop-down.***
Drag an 'OR 2' function-block.

Move the function-blocks, and connect / set the ports as shown in Figure 36.

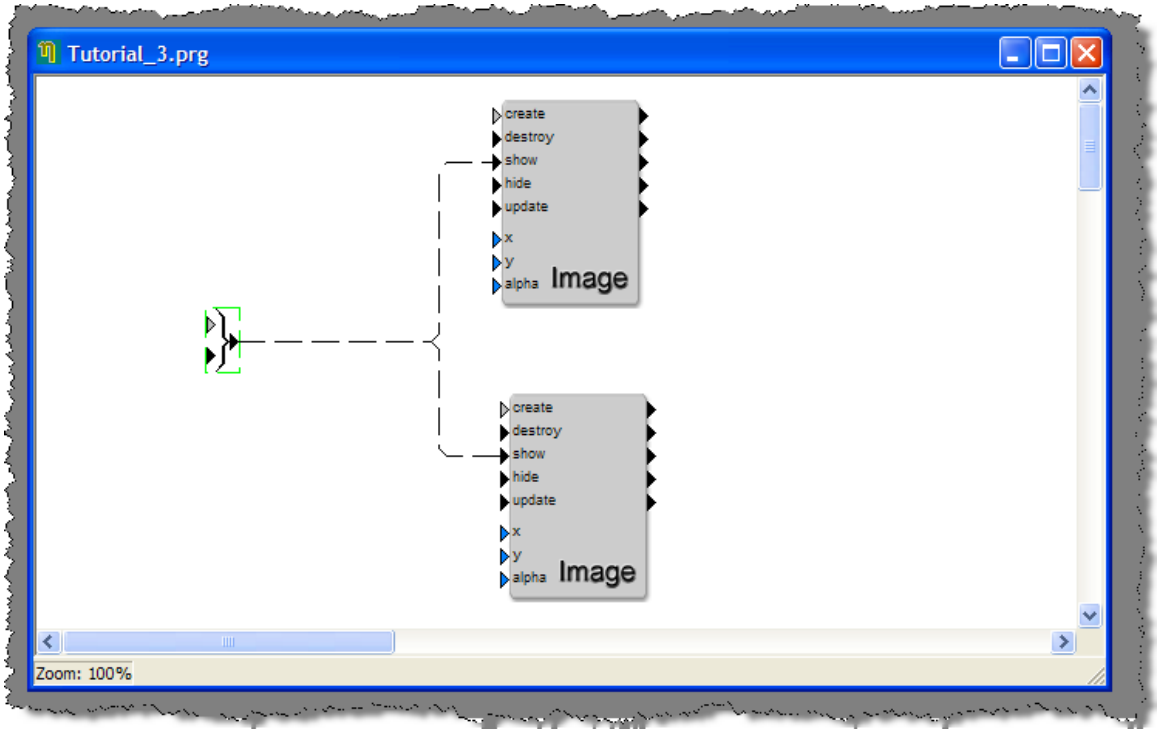

Figure 36: Tutorial 2 Application

Create a new widget-group

Double-click the lower widget to bring up its properties dialog, as shown in Figure 37.

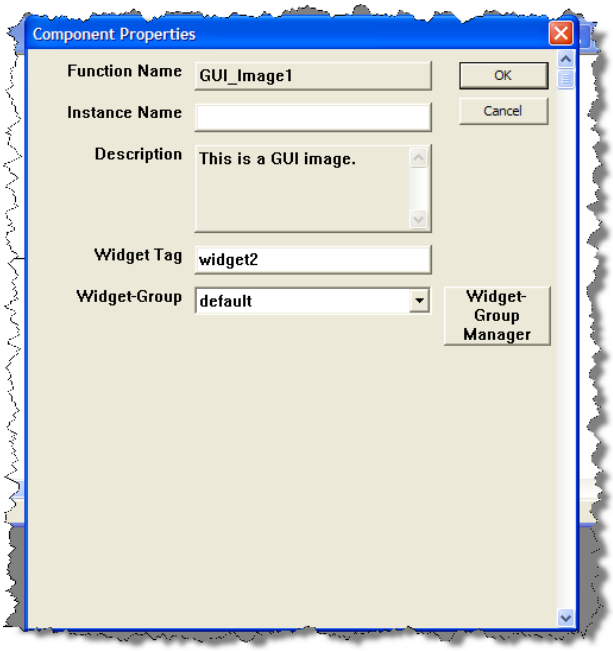

Figure 37: IAB Component Properties Dialog

Click the 'Widget-Group Manager' button to bring up the widget-group manager dialog, similar to that shown in Figure 38.

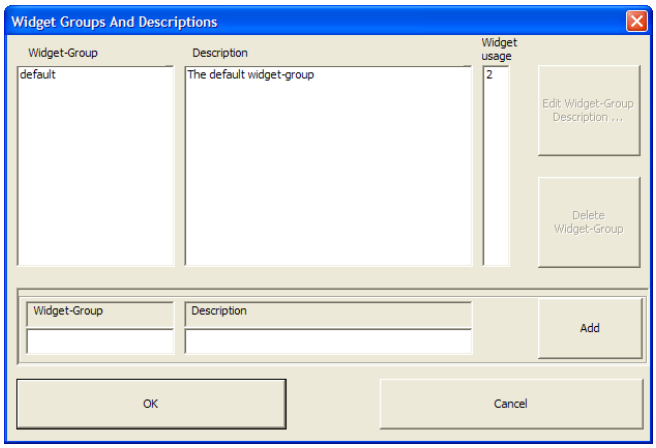

Figure 38: Widget-Group manager Dialog
Note that the Widget-usage of the 'default' widget-group is 2. This means that 2 widgets belong to this group.

Create a new widget-group

Type 'Another Group' and 'For an Example' into the 'Widget-Group'and 'Description' fields respectively

The dialog should appear as shown in Figure 39.

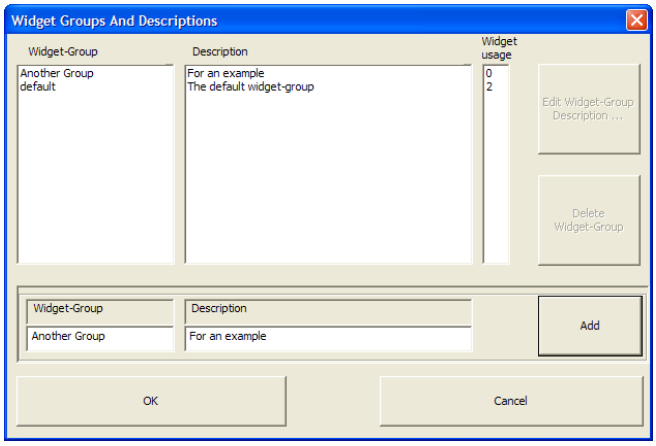

Figure 39: Widget-Group manager Dialog

Left-click the 'Add' button

[OK] the dialog.

Allocate the 2nd widget to the new group

In IAB, double-click the lower widget.

In the ensuing 'Properties' dialog, click the combo-box on the 'Widget-Group' field of the widget's properties dialog

Set the widget-group to the one you just created i.e. 'Another Group'.

Check Widget-usage

Invoke the Widget-Group manager (by double-clicking either widget in IAB, then clicking 'Widget-Group manager' button on the ensuing dialog).

Note the new 'usage' details of the widget-groups, in that each widget group is reported as being 'used' by just one widget.

Dismiss the dialogs.

Clashing Layouts

Create a second layout for widget-group 'default', as follows:

[IAB] File->Write SODL {Left Click}

[IAB] double-click a layout in the Project-Overview window.

This invokes an instance of IGD, pre-set to the present project.

In IGD: Invoke the Layout Manager (File->Layout Manager)

Left-click [New Layout].

In the ensuing dialog, enter 'Default2' as the name of the layout, and 'default' as its widget-group.

OK the 'new layout' dialog.

Open the layouts

Open ***all*** the project layouts, as follows:

Default widget-group

In the Layout manager, select the 'default' tab,

Left-click one layout, then

Ctrl-Left-click the other layout.

Click the 'Open selected layouts' button.

'Another Group' widget-group

Then select the 'Another Group' tab, and open the layout (again, by selecting it)

Click the 'Open selected layouts' button.

Dismiss the dialog

Note that the text in the title bar of the 'default' widget-group layouts is suffixed with a 'clash' warning.

Close the 'Another Group' layout..

Note that the clash warning on the default group layouts remains.

Close one of the 'Default' windows.

Note that the clash warning is removed.

Having more than one layout for a widget-groups open at the same time is known as a 'clash'. A clash is potentially confusing for the user, as each clashing layout provides a different definition of where the widgets should be placed on the screen.

The 'Active layout' concept

Discussion

As alluded to above, it is perfectly possible - and often desirable - to create within IGD more than one layout file for a given widget-group. So that the user can inform Inxware which layout should be transferred to the target, Inxware provides the concept of the 'active layout'.

For each widget-group, one and only one layout can and must be nominated as active. For each widget-group, when a transfer-to-target is invokes, it is the active layout which will be eligible[2] for transfer to target.

Re-examine the Layout Manager Dialog

Open the Layout manager in IGD.

Select the tab of the 'Another Group' widget-group.

Note that you cannot uncheck the 'active' checkbox. This is because at least one of the widget-group's layouts must be active, and as this is the only layout for this widget-group, it must be the active one.

Now select the 'Default' widget-group tab.

Note that because this has more than one layout, it is possible to change which layout is active.

Play with the positions of the widgets in the various layouts, of both the active and non-active layouts.

In between rearrangements of widget positions, etc, transfer to target.

Note from the widget-position changes manifested on the target that only the active layout gets sent across to the target.

File 'PC names' vs 'Target names'

Every file that can be transferred to the target, be it a layout, an image file, text file, whatever, has potentially two different incarnations. The first is on the PC (ie within Inxware). On the PC, the file can have a nice (meaningful to the user) long windows file name with upper / lower case, spaces etc. However, as a layout is transferred to the target, to cater for the target's (potential) limitations, it must be given an (8.3) target name.

[2] It will be sent either if a 'Transfer All' has been invoked, or
if a 'transfer Changed' AND the layout is out-of-date

***Conclusion**: Do not expect files on the target to have the same name as the corresponding file on the PC.*

Furthermore, and to keep things simple, Inxware specifies that for any given widget-group, there is the same target name for any associated layouts. Thus a layout for a given widget-group will, when transferred to target, automatically overwrite previously-transferred layouts for its own widget-group.

Appendix

The 'SODL' file – an Overview

SODL and EHS

The way Inxware works is as follows:

- IAB is used to edit a application which will behave as required. The application defines a set of event-processing instructions that dictate how events are processed, by what components, etc.
- The 'essence' of these instructions is contained in the SODL file, which is writable by IAB, on command.
- EHS, the 'event-handling system', is the framework which Inxware runs on the target platform. EHS enacts the programmatic instructions in the SODL file.

So, the SODL file is needed on the target by EHS , and must be transferred there before the application can run (- more on the 'transfer' process later).

IGD Main Window Tools

This IGD tool presents on the right two slider controls. Without going into detail (which is provided later), we shall briefly discuss the main components. These are:

The z-clipping control

z-clipping is a designer's aid that allows the upper layers of widgets to be removed from the view to allow temporary access to widgets beneath.

The pseudo-3D lean control

This allows the developer an alternative method of separating overlapping widgets by visualising the widgets with a 3D perspective related to their z-position.

Application Resource Compilation Errors

Sometimes a situation can arise where EHS cannot run the application transferred as the resources have not been properly synchronised in the tools. EHS, as fail-safe, detects and reports errors as the files are transferred, and if it does so, it sends back error notification to the Inxware tool that was doing the transferring, where errors are displayed in a popup dialog.

For the purposes of a demonstration, an easy error to inject is that of not having the correct layouts for the application. To do this:

In IAB, copy the image function-block and paste the copy into composition window.

Having pasted, and after the 'widget-group' dialog has popped-up, use the dialog to create a new widget group to which to allocate the pasted widget.

Now, from IAB, transfer to target.
An error dialog should appear as shown in Figure 40.

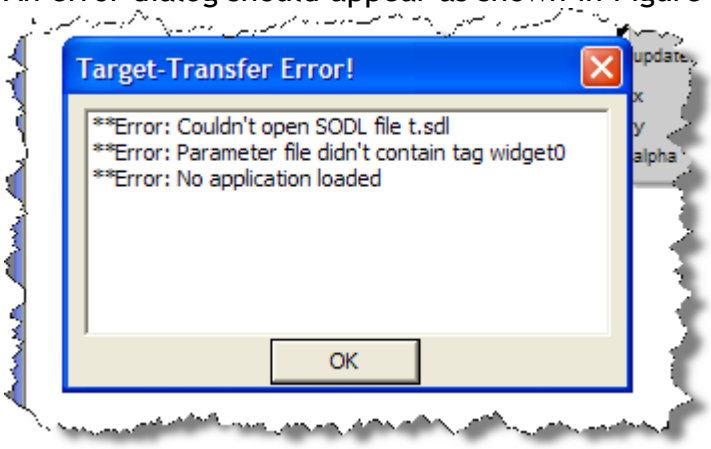

Figure 40: Transfer-Error Dialog

The 2nd widget belonged to a new widget group. IGD had not read-in an updated SODL to be able to produce a new layout for this widget-group. EHS detected that no layout had been received onto the target for this widget, and issued some error messages that were displayed in the popup.