dcapp Installation and User Guide

version 1.0

NASA Johnson Space Center

Table of Contents

1.0 Introduction

"dcapp" (pronounced "dee see app") is a displays and controls package designed to run in conjunction with an external Trick-based simulation. It is built upon standard UNIX technologies, like OpenGL and X11. It uses libxml2 for input file parsing, FreeType2 for font handling, and GLUT for window management and event handling. It uses trick_comm to communicate with an external Trick-based simulation and EDGE's remote commanding server (RCS) to communicate with EDGE graphics.

For more information, please contact:

Michael McFarlane
michael.r.mcfarlane@nasa.gov
ER7/Simulation and Graphics Branch
NASA Johnson Space Center

2.0 Installation

2.1 Mandatory Prerequisites

dcapp is designed to run on MacOS and Linux-based machines. For all of the packages described hereafter, be sure to get "development" versions of those packages that include header files. Besides OpenGL and X11, the following packages must be installed:

- libxml2
- FreeType2

libxml2 is an XML file parser that is a standard package on MacOS and most Linux installations, but it can be accessed at http://xmlsoft.org if needed.

FreeType2 is a freely available software library for rendering fonts. It is capable of producing high-quality output (glyph images) of most vector- and bitmap- font formats. It is a standard package on most MacOS and Linux installations, but it can be accessed at http://www.freetype.org if needed.

2.2 Optional Prerequisites

If dcapp is to be run in conjunction with a Trick simulation, then the TRICK_HOME environment variable must be properly set AND a stand-alone version of trick_comm must be successfully built prior to building dcapp. trick_comm is a Trick library that provides an interface to a Trick simulation via the Trick variable server.

Note that Trick 13 does not automatically build the stand-alone version of trick_comm. To build the stand-alone version of trick_comm, install Trick, `cd` to ${TRICK_HOME}, and type "`make stand_alone`".

dcapp can be configured to monitor hardware inputs (dials, switches, etc.) via a controller area network (CAN) bus. CAN is a serial bus protocol used to connect individual systems and sensors over a single- or dual-wire networked data bus. Be sure that the CAN bus software is appropriately built and that the CANBUS_HOME environment variable is set to the directory containing the necessary header and library files.

By default, dcapp utilizes a built-in library for managing windows and mouse/keyboard events. However, GLUT (see http://www.opengl.org/resources/libraries/glut/ for more information) is an alternative that can be incorporated into dcapp by setting the environment variable `UseGLUT` to "`yes`" prior to building dcapp. Note that GLUT is a standard package on most MacOS installations. dcapp has been built successfully with version 3.6 of GLUT.

By default, dcapp utilizes a built-in library for rendering fonts. However, FTGL (see http://sourceforge.net/projects/ftgl/ for more information) is an open-source alternative that can be incorporated into dcapp by setting the environment variable `UseFTGL` to "`yes`" prior to building dcapp. If the FTGL headers and libraries are not contained in a standard directory that the compiler can locate, the user may also set `FTGL_CFLAG` and/or `FTGL_LFLAG` environment variables to specify –I and/or –L flags to help the compiler locate these files. Note that dcapp has been built successfully with version 2.1.3-rc5 of FTGL.

2.3 dcapp

Extract the dcapp package if necessary, `cd` to the top level of the package, and type "`make`". This should build the dcapp executable within the bin_${OSSPEC} subdirectory, where OSSPEC corresponds to $TRICK_HOST_CPU if it is set or `uname -s` if not. You should then add the bin_${OSSPEC} subdirectory to your $PATH environment variable.

3.0 Activation

After following the instructions in section 2, simply type the following on the command line to activate dcapp:

```
dcapp file.xml [-h hostname] [-p port] [-d display] [const=value...]
```

where `file.xml` is a full or partial path to a valid dcapp specfile (see section 4 for more information on dcapp specfiles). Note that the command-line options, outlined below, may be used to override default values and/or values specified in the specfile:

> `-h hostname`: specify the hostname upon which the Trick sim is executing
> `-p port`: specify the port over which communication with the Trick variable server takes place
> `-d display`: specify the X display upon which the dcapp should be rendered
> `const=value...`: the value of any constants defined within the specfile can be overwritten on the command line

For instance, if a user wants to run dcapp with a specfile called myspec.xml communicating with Trick over port 1234 and overriding the constants "WinWidth" and "WinHeight" with "480" and "640" respectively, the user would type the following command:

```
dcapp myspec.xml -p 1234 WinWidth=480 WinHeight=640
```

Note that on MacOS, an alternative to launching dcapp via the command line is to use dcapp.app, which is automatically built during the "`make`" step described in section 2.3. dcapp.app can be launched like any MacOS application (double clicking it, launching it from the Dock, etc.). It brings up a simple user interface that requests the information described above from the user, then proceeds to launch dcapp accordingly.

4.0 Specfile

The dcapp specfile is a standard XML file used to customize the features and capabilities of dcapp. See http://www.w3.org/XML/ for more information about XML files, including valid file specifications, definition and usage of character entities, use of comments, etc. The elements contained within the dcapp specfile are detailed in this section.

4.1 Root Element

Element	DCAPP
Parent	(none)
Children	(any)
Attributes	(none)
Description	All dcapp specfiles must contain this root element. All of the other elements, described in the following sections, must be enclosed within this root element.

4.2 Universal Elements

These elements may appear anywhere within the dcapp specfile, and they may be embedded within any element that allows children.

Element	Dummy
Parent	(any)
Children	(any)
Attributes	(none)
Description	This element does nothing besides allowing the user to group sub-elements. This is potentially useful when using XML's <xi:include> element, which requires included files to be "well-formed", which means, among other things, that the file must contain only one element at its root level.

Element	If
Parent	(any)
Children	True, False, (any)
Attributes	Operator, Value, Value1, Value2
Description	This element applies the *Operator* (one of "eq", "ne", "gt", "lt", "ge", or "le") to *Value1* and *Value2* to evaluate a true or false condition. If no *Operator* is defined, then it simply tests *Value* to determine true or false. If the logic evaluates to true, then the sub-elements within the "True" element are processed, otherwise, the sub-elements within the "False" element are processed. If there is no "True" or "False" sub-element defined, the contents of this element are assumed to be contained within a virtual "True" element.

Element	True
Parent	If
Children	(any)
Attributes	(none)
Description	This element simply encloses sub-elements that are to be processed if the logic of the encompassing "If" element resolves to "true".

Element	False
Parent	If
Children	(any)
Attributes	(none)
Description	This element simply encloses sub-elements that are to be processed if the logic of the encompassing "If" element resolves to "false".

Element	Set
Parent	(any)
Children	(none)
Attributes	Variable, Operator, MinimumValue, MaximumValue
Description	This sets the value of *Variable* to a new value defined by the content of

	the element. The *Operator* is "=" by default, but may also be "+=" or "-=" if this element is to be used to increment or decrement *Variable* (usable only if *Variable* is a numeric type). *MinimumValue* and *MaximumValue* may optionally be set to bound the new numeric value.

4.3 Initialization Elements

These elements typically appear near the top of the dcapp specfile. They define the behavior of subsequent elements within the specfile.

4.3.1 Settings Elements

Element	Constant
Parent	DCAPP
Children	(none)
Attributes	Name
Description	This allows a user to create a constant that can be accessed subsequently within the specfile. This is handy for setting values that are used frequently throughout the display. For instance, the user may set: `<Constant Name="FontSize">24</Constant>` The pre-processor will then replace all instances of "#FontSize" in the rest of the specfile with "24".

Element	Variable
Parent	DCAPP
Children	Type, InitialValue
Attributes	(none)
Description	This allows a user to create a variable that can be accessed subsequently within the specfile. The *Type* must be either "Float", "Integer", or "String". For instance, the user may set: `<Variable Type="Integer">MyVar</Variable>` Any subsequent elements may then use the associated value by specifying a value of "@MyVar". Note that if *InitialValue* is not specified, the default value is 0 for float and integer parameters and an empty string ("") for string parameters.

Element	Style
Parent	DCAPP
Children	(any)
Attributes	Name
Description	This allows a user to define a style, which defines attributes for any element that is used subsequently within the specfile. For instance, the user may set: `<Style Name="mystyle">` `    <String Size="28" Color="0 0 1"/>` `</Style>`

	Then, a subsequent “String” element that uses “mystyle” (`<String style=”mystyle”…`) will be blue and use a font size of 28 by default. Note that multiple elements may be defined within a single “Style” element.

Element	Defaults
Parent	DCAPP
Children	(any)
Attributes	(none)

Description	This allows a user to define default attributes for any element that is used subsequently within the specfile. For instance, the user may set:
	`<Defaults>`
	`    <Rectangle LineWidth=”2” LineColor=”1 0 0”/>`
	`</Defaults>`
	Then, all subsequent “Rectangle” elements will be rendered with a red line that is 2 pixels thick by default. Note that multiple elements may be defined within a single “Defaults” element.

4.3.2 Input/Output Elements

Element	TrickIo
Parent	DCAPP
Children	FromTrick, ToTrick
Attributes	Host, Port, DataRate
Description	This construct specifies communication between dcapp and the Trick variable server. *Host* specifies the hostname upon which the Trick simulation is executing. If not specified, the default value is the hostname of the machine upon which dcapp is executing. *Port* specifies the port over which communication with the Trick variable server takes place. If not specified, the default value is 7000. *DataRate* specifies the data rate (in seconds) at which Trick will attempt to communicate with dcapp. If not specified, the default value is 1 second. Note that the values for *Host* and *Port* may be overridden by the command-line arguments outlined in section 3.

Element	FromTrick
Parent	TrickIo
Children	TrickVariable
Attributes	(none)
Description	This contains a list of the “TrickVariable” elements that are used to over-write dcapp data with data from the attached Trick simulation.

Element	ToTrick
Parent	TrickIo
Children	TrickVariable
Attributes	(none)

Description	This contains a list of the “TrickVariable” elements that are used to over-write Trick simulation data with data from dcapp.

Element	TrickVariable
Parent	FromTrick, ToTrick
Children	(none)
Attributes	Name, Units
Description	This element attaches a dcapp “Variable” to the variable in the attached Trick simulation defined by *Name*. The user may optionally define the *Units* of the data within dcapp, which the Trick variable server will use to convert the data, if necessary. The *Units* must be a unit string recognizable by Trick. For instance: `<TrickVariable Name="trickobj.var">MyVar</TrickVariable>`

Element	EdgeIo
Parent	DCAPP
Children	FromEdge, ToEdge
Attributes	Host, Port, DataRate
Description	This construct specifies communication between dcapp and EDGE via EDGE’s remote commanding server server. *Host* specifies the hostname upon which EDGE is executing. If not specified, the default value is the hostname of the machine upon which dcapp is executing. *Port* specifies the port over which communication with EDGE takes place. If not specified, the default value is 5451. *DataRate* specifies the data rate (in seconds) at which EDGE will be polled by dcapp. If not specified, the default value is 1 second.

Element	FromEdge
Parent	EdgeIo
Children	EdgeVariable
Attributes	(none)
Description	This contains a list of the “EdgeVariable” elements that are used to over-write dcapp data with data from the attached EDGE instance. For instance: `<EdgeVariable RcsCommand="doug.node Light set –lit_int">LightCmd</EdgeVariable>`

Element	ToEdge
Parent	EdgeIo
Children	EdgeVariable
Attributes	(none)
Description	This contains a list of the “EdgeVariable” elements that are used to over-write EDGE data with data from dcapp.

Element	EdgeVariable
Parent	FromEdge, ToEdge

Children	(none)
Attributes	RcsCommand
Description	This element attaches a dcapp “Variable” to the variable in the attached EDGE instance defined by *RcsCommand*.

Element	CAN
Parent	DCAPP
Children	(none)
Attributes	Network, ButtonID, ControlID
Description	This element assigns bezel keys to data associated with a CAN bus based upon *Network*, *ButtonID*, and *ControlID* of the unit associated with this instance of dcapp. The bezel keys are processed via the “Button” and/or “BezelEvent” elements.

Element	UEI
Parent	DCAPP
Children	(none)
Attributes	Host, Port, BezelID
Description	This element assigns bezel keys to data associated with a UEI controller based upon the *Host* and *Port* of the UEI and the *BezelID* of the unit associated with this instance of dcapp. The bezel keys are processed via the “Button” and/or “BezelEvent” elements.

4.3.3 Logic Element

Element	DisplayLogic
Parent	DCAPP
Children	(none)
Attributes	(none)
Description	The content of this element specifies a shared object file to be linked into dcapp at execution time. See section 5.3 for more information about the format and content of this file.

4.4 Display Setup

Element	Window
Parent	DCAPP
Children	Panels
Attributes	X, Y, Width, Height, FullScreen, XDisplay, ForceUpdate
Description	This defines the position (*X* and *Y*) and size (*Width* and *Height*) of the window containing the dcapp displays. If *FullScreen* is set to “true”, “yes”, or “on”, the window will be rendered full screen regardless of *X*, *Y*, *Width*, and *Height* settings. If dcapp is being executed in an X11 windowing system, the user can specify *XDisplay* to run dcapp on an alternate display. By default, dcapp only updates when it senses an

	event (a mouse event, input data change, etc.), but the user may set *ForceUpdate* to specify an interval, in seconds, after which dcapp will automatically update.

Element	Panels
Parent	Window
Children	Panel
Attributes	ActiveDisplay
Description	This serves as a container for the individual display panels within a dcapp instance. The *ActiveDisplay* attribute allows the user to assign a variable to determine which display is active at any given time. If the value of this variable corresponds to the *DisplayIndex* of a given panel (see below), then that panel becomes the active display.

Element	Panel
Parent	Panels
Children	(display primitives)
Attributes	DisplayIndex, BackgroundColor, VirtualWidth, VirtualHeight
Description	This contains all of the display primitives for a given display panel. The *DisplayIndex* attribute is used to define when this display is the active display. *BackgroundColor* specifies the background color for the panel. See section 5.1 for information on specifying color. If not specified, the default color is black ("0 0 0"). *VirtualWidth* and *VirtualHeight* define the user-specified geometry of the display panel, which is used to render the position and size of the display primitives. If not specified, the default geometry is 100x100 units.

4.5 Display Primitives

The display primitives are the building blocks that define how the individual display panels look, feel, and react to user input. They are grouped into two primary classifications: visual primitives, which are primitives that render data to the screen, and event primitives, which are primitives that handle user input.

4.5.1 Visual Primitives

Element	Container
Parent	Panel, Container, Button, Active, Inactive, On, Transition, Off
Children	(display primitives)
Attributes	X, Y, Width, Height, HorizontalAlign, VerticalAlign, VirtualWidth, VirtualHeight, Rotate
Description	This redefines the coordinate frame for subsequent primitives by allowing the user to define a box of size *Width* by *Height* at position *X*, *Y*, and aligned by *HorizontalAlign* and *VerticalAlign*, within the current coordinate frame. The new coordinate frame can also be rotated by *Rotate* degrees from the current coordinate frame, and the new

	coordinate frame uses *VirtualWidth* and *VirtualHeight* to define the width and height of subsequent elements within the new frame.

Element	Line
Parent	Panel, Container, Button, Active, Inactive, On, Transition, Off
Children	Vertex
Attributes	LineWidth, Color
Description	This attaches the enclosed "Vertex" primitives to form a single, continuous line with the specified *LineWidth* and *Color*.

Element	Polygon
Parent	Panel, Container, Button, Active, Inactive, On, Transition, Off
Children	Vertex
Attributes	FillColor, LineColor, LineWidth
Description	This attaches the enclosed "Vertex" primitives to form a polygon. The polygon is filled with *FillColor* and outlined with a line of color *LineColor* and a width of *LineWidth*. If *FillColor* is not set, then the polygon is not filled. Likewise, if *LineColor* and *LineWidth* are not set, then the polygon is not outlined.

Element	Vertex
Parent	Panel, Container, Button, Active, Inactive, On, Transition, Off
Children	(none)
Attributes	X, Y
Description	This defines the *X* and *Y* coordinates of a vertex within a "Line" or "Polygon" primitive.

Element	Rectangle
Parent	Panel, Container, Button, Active, Inactive, On, Transition, Off
Children	(none)
Attributes	X, Y, Width, Height, HorizontalAlign, VerticalAlign, Rotate, FillColor, LineColor, LineWidth
Description	

Element	Circle
Parent	Panel, Container, Button, Active, Inactive, On, Transition, Off
Children	(none)
Attributes	X, Y, HorizontalAlign, VerticalAlign, Radius, Segments, FillColor, LineColor, LineWidth
Description	

Element	String
Parent	Panel, Container, Button, Active, Inactive, On, Transition, Off
Children	(none)
Attributes	X, Y, Rotate, Size, HorizontalAlign, VerticalAlign, Color,

	BackgroundColor, ShadowOffset, Font, Face, Format, ForceMono
Description	

Element	Image
Parent	Panel, Container, Button, Active, Inactive, On, Transition, Off
Children	(none)
Attributes	X, Y, Width, Height, HorizontalAlign, VerticalAlign, Rotate
Description	

Element	PixelStream
Parent	Panel, Container, Button, Active, Inactive, On, Transition, Off
Children	(none)
Attributes	X, Y, Width, Height, HorizontalAlign, VerticalAlign, Rotate, SharedMemoryKey
Description	

Element	ADI
Parent	Panel, Container, Button, Active, Inactive, On, Transition, Off
Children	(none)
Attributes	X, Y, Width, Height, HorizontalAlign, VerticalAlign, OuterRadius, BallRadius, ChevronWidth, ChevronHeight, BallFile, CoverFile, Roll, Pitch, Yaw, RollError, PitchError, YawError
Description	

4.5.2 Event Primitives

Element	Button
Parent	Panel, Container
Children	Active, Inactive, On, Transition, Off, OnPress, OnRelease, (display primitives)
Attributes	X, Y, Width, Height, HorizontalAlign, VerticalAlign, Rotate, Type, Key, KeyASCII, BezelKey, Variable, On, Off, SwitchVariable, SwitchOn, SwitchOff, IndicatorVariable, IndicatorOn, ActiveVariable, ActiveOn
Description	

Element	Active
Parent	Button
Children	(display primitives)
Attributes	(none)
Description	

Element	Inactive
Parent	Button
Children	(display primitives)
Attributes	(none)

Description	

Element	On
Parent	Button
Children	(display primitives)
Attributes	(none)
Description	

Element	Transition
Parent	Button
Children	(display primitives)
Attributes	(none)
Description	

Element	Off
Parent	Button
Children	(display primitives)
Attributes	(none)
Description	

Element	MouseEvent
Parent	Panel, Container
Children	OnPress, OnRelease, Set
Attributes	X, Y, Width, Height, HorizontalAlign, VerticalAlign
Description	

Element	KeyboardEvent
Parent	Panel, Container
Children	OnPress, OnRelease, Set
Attributes	Key, KeyASCII
Description	

Element	BezelEvent
Parent	Panel, Container
Children	OnPress, OnRelease, Set
Attributes	Key
Description	

Element	OnPress
Parent	Button, MouseEvent, KeyboardEvent, BezelEvent
Children	Set
Attributes	(none)
Description	

Element	OnRelease

Parent	Button, MouseEvent, KeyboardEvent, BezelEvent
Children	Set
Attributes	(none)
Description	

5.0 Technical Details

5.1 Color Format Specification

When specifying color formats for any dcapp display elements, the following format must be used:

```
red_level green_level blue_level
```

where each level is expressed as a number between 0 (full off) and 1 (full on). In other words, black would be specified as “0 0 0”, white would be “1 1 1”, blue would be “0 0 1”, grey might be “0.5 0.5 0.5”, etc.

5.2 Alignment Specification

5.3 Graphic File Formats

dcapp can currently handle graphic files in two formats: TARGA (.tga) and bitmap (.bmp). TARGA files should be saved uncompressed with a “bottom left” origin. Bitmap files should be saved in 24-bit format, although files saved in other valid bitmap formats may work.

5.4 Display Logic File

5.5 Element Values

5.5.1 Constants

5.5.2 Variables

5.5.3 Environment Variables