
Edje EDC Reference

Ben Rockwood, The Enlightenment Project <benr@cuddletech.com>

Revision v0.1

Revision History
August 15th 2003
Initial yank from Edje Manual

br

Table of Contents

..... 1

Figure 1. EDC General Form

```
images {  
    // Images  
}  
  
data {  
    // Data  
}  
  
collections {  
    group {  
        // Group1 Params  
  
        data {  
            //Data  
        }  
  
        parts {  
            part {  
                // Part Params  
  
                description {  
                    // State Params  
                }  
            }  
        }  
  
        programs {  
            program {  
                // Program Params  
            }  
        }  
    }  
  
    group {  
        // Group2 Params  
        parts {  
            .....  
        }  
        programs {
```


Section	Keyword	Parameters	Description
			ceptable FX_TYPES are: NONE, PLAIN, OUT-LINE, SOFT_OUTLINE, SHADOW, SOFT_SHADOW, OUT-LINE_SHADOW, OUT-LINE_SOFT_SHADOW. Default type is NONE. ex: effect, SOFT_OUTLINE;
part	mouse_events,	0;	Boolean value specifying whether the part accepts mouse events or not. No signals are generated from parts that do not accept events.
part	repeat_events,	0;	Boolean value specifying whether a part repeats an event to the part below it. When repeat is set to 0 (off, the default) and two parts that accept events are on top of each other the top most object will receive the event and not any parts below it, turning repeat to 1 (on) will continue to send the event down to the next part below it.
part	clip_to,	"part";	Clip to the size of the specified part. Any amount of the current part that extends beyond the size of the clipped to part will be clipped off. Clipped text parts always truncate the text string to "...".
part	color_class,	"class";	Name of color class to apply to the current part. Color classes are defined in application code.
part	text_events,	"class";	Name of text class to apply to the current part. Text classes are defined in application code.
description	state,	"name" INDEX;	Descriptive name for the individual state, the default state must always be named "default". The INDEX value is a double between 0.0 and 1.0 which indicates levels of completion, that defaults to 0.0. Multiple states can have the same name yet with a different index value. ex: state, "de-

Section	Keyword	Parameters	Description
			fault" 0.0;
description	visible,	0;	Boolean value specifying whether part is visible or not. Non-visible parts do not generate events.
description	align,	HOR_VAL VER_VAL;	Specify alignment of the part within it's container as specified by rel1/rel2. Values are specified as doubles from 0.0 (align left/top) to 1.0 (align right/bottom). ex: align, 0.5 0.5; (Aligns part in center of container) ex: align, 0.0 1.0; (Aligns part to bottom left of container)
description	min,	HOR_SIZE VER_SIZE;	Integer values specifying minimum horizontal (arg1) and vertical (arg2) size of part in pixels. ex: min, 100 100;
description	max,	HOR_SIZE VER_SIZE;	Integer values specifying maximum horizontal (arg1) and vertical (arg2) size of part in pixels. ex: max, 100 100;
description	step,	HOR_VAL VERT_VAL;	Integer stepping values in integer pixels for horizontal (arg1) and vertical (arg2) scaling. When stepping is enabled the width or/and height of the image will always be divisible by it's stepping value when scaled. Default stepping values are 0 0 (ie: stepping disabled). ex: step, 20 1 (Image width must always be multiple of 20, ie: 0, 20, 40, 60, etc. Height can be any value)
description	aspect,	MIN MAX;	Double min (arg1) and max (arg2) aspect ratio values. This controls the aspect ratio (ratio of width to height) of a scaled part, typically images. The default ratio is 0.0. If both values are the same the ratio is fixed. ex: aspect, 1.0 5.0; (Minimum aspect of 1:1, maximum of 5:1 - Width:Height);
description	border,	LEFT RIGHT TOP BOTTOM;	Border scaling values for an image part as specified in integer pixel widths, for each four sides of an image.

Section	Keyword	Parameters	Description
			This will stop Edje from scaling the outside edge of an image when scaling an image part. ex: border, 10 10 10 10; (Scale the edge of the image part 10 pixels on all sides)
description	color,	RED GREEN BLUE ALPHA;	Integer values ranging from 0 to 255 specifying the color of a rectangle or text part. ex: color, 0 0 0 255; (Part is colored black)
description	color2,	RED GREEN BLUE ALPHA;	Integer values ranging from 0 to 255 specifying the color of a text parts shadow. ex: color2, 0 0 255 255; (Shadow is blue)
description	color3,	RED GREEN BLUE ALPHA;	Integer values ranging from 0 to 255 specifying the color of a text parts outline. ex: color3, 255 0 0 255; (Outline is colored red)
rel1/rel2	relative,	HORZ_VAL VERT_VAL;	Doubles representing the horizontal (arg1) and vertical (arg2) positioning of top left corner (for rel1) or bottom right corner (for rel2) as relative to the part specified by the "to" keyword. If no "to" keyword is present, the values are relative to the corners of the interface. ex: relative, 0.0 1.0; (For rel1 with no "to": top left corner of part is positioned at the left (0.0), bottom (1.0) corner of the interface.)
rel1/rel2	offset,	HORZ_OFF VERT_OFF;	Integers specifying deviation in pixels from the position as defined by the relative keyword, both horizontally (arg1) and vertically (arg2) ex: offset, 5 10; (Position 5 px to the right and 10 px down from the position as stated by the relative keyword)
rel1/rel2	to,	"part_name";	Specify another part as the reference to be used for the positioning of the current part. ex: to, "some_part";
rel1/rel2	to_x,	"part_name";	Specify another part as the reference to be used for the

Section	Keyword	Parameters	Description
			positioning of the current part. Same as "to", but relativity applies only on the X axis.
rel1/rel2	to_y,	"part_name";	Specify another part as the reference to be used for the positioning of the current part. Same as "to", but relativity applies only on the Y axis.
image	normal,	"image_name";	Name of image to be used. In an animation, this is the first and last image displayed.
image	tween,	"image_name";	Name of an image to be used in an animation loop. Images are display in the order they are listed. There is no limit to the number of tweens that can be specified.
fill	smooth,	0;	Boolean value determining whether scaled images will be smoothed, 0 for no, 1 for yes.
fill { origin	relative,	HOR_VAL VERT_VAL;	Doubles representing the horizontal (arg1) and vertical (arg2) position from which a fill (tile) should start within it's container as defined by rel1/rel2. Tiling then occurs in all directions from that point of origin. This is similar in use to relativity by rel1 except that it is relative to the parts container rather than the whole interface. ex: relative, 0.5 0.5; (part starts tiling from the middle of it's container)
fill { origin	offset	HOR_VAL VERT_VAL;	Integers specifying a pixel offset horizontally (arg1) and vertically (arg2) from the relative position specified by origin{relative,}. This is similar in use to offset used in rel1.
fill { size	relative,	HOR_VAL VERT_VAL;	Doubles representing the horizontal (arg1) and vertical (arg2) position of the bottom right corner of a fill (tile). This is similar in use to relativity by rel2 except

Section	Keyword	Parameters	Description
			that it is relative to the parts container rather than the whole interface. ex: relative, 1.0 1.0; (Tile fills entire space)
fill { size	offset,	HOR_VAL VERT_VAL;	Integers specifying a pixel offset horizontally (arg1) and vertically (arg2) from the relative position specified by size{relative,}. This is similar in use to offset used in rel2.
text	text,	"some string";	Text string to be rendered.
text	font,	"font_name";	Font used for text, where "font_name" is the name of the font file minus its extension. Path to font is determined by your applications evas font path. ex: font, "Impact"; (Font used is Impact.ttf found in the evas font path)
text	size	12;	Font size in points.
text	fit,	HOR_VAL VERT_VAL;	Boolean values specifying whether to scale text to fill its container horizontally (arg1) and/or vertically (arg2). Default is 0 0;
text	min,	HOR_VAL VERT_VAL;	Boolean values specifying whether the current text string should define the minimum size of the part, such that all future changes to the text string can be no smaller both horizontally (arg1) and vertically (arg2).
text	align,	0.5 0.5;	Alignment of text within its containers as defined by rel1/rel2, horizontally (arg1) and vertically (arg2).
program	name,	"prog_name";	Symbolic name of program as a unique identifier.
program	signal,	SIGNAL;	Specifies signal(s) that a should cause program to run. The signal recieved must match the specified source to run. Signals may be globbed, but only one signal keyword per program may be used. ex: signal, "mouse.clicked,*"; (clicking any mouse button that matches source starts

Section	Keyword	Parameters	Description
			program)
program	source,	"signal-source";	Source of accepted signal. Sources may be globbed, but only one source keyword per program may be used. ex: source, "button-*"; (Signals from any part or program named "button-*" are accepted)
program	action,	ACTION (param1) (param2);	Action to be performed by the program. Valid actions are: STATE_SET, ACTION_STOP and SIGNAL_EMIT. Only one action can be specified per program.
program	transition,	TYPE LENGTH;	
program	target,	"action-target";	Program or part on which the specified action acts. Multiple target keywords may be specified, one per target. SIGNAL_EMITs do not have targets.
program	after,	"next-program";	Specifies a program run after the current program completes. The source and signal parameters of a program run as an "after" are ignored.