Reactor Image Processing

Reactor has a variety of powerful image processing capabilities that can facilitate the rendering of dynamic graphics for monochrome, grayscale and color displays on SKAARHOJ controllers.

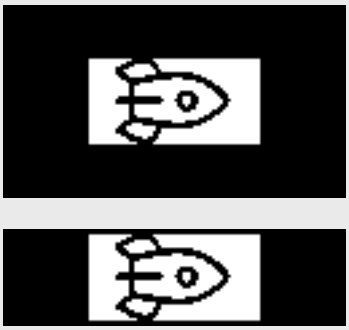
The basics

The DataSource decides what kind of graphics you will render. These are the options:

- Icon - a reference to an embedded icon file in Reactor (see list in appendix)
- Inline - a base64 inline image string
- IOref - image taken from an IO reference, such as a thumbnail from a device core.
- QRcode - generates a QR code
- Generator - generates a gray wedge or sample frames
- Composition - produces a layered composition of various types of elements, including those in this list.
- Widget - generates a complex, often dynamic graphic used in specific cases on a SKAARHOJ panel, such as a VU meter or a strength indicator for a T-bar display.

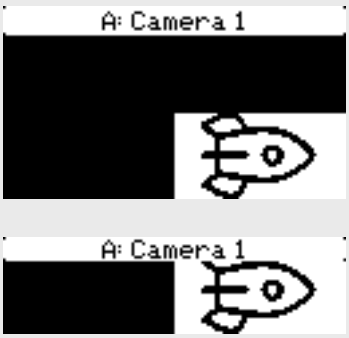
The output from any of the data sources is rendered onto the destination tile on the panel, but being subject to various filters, alignments and other conveniences. This table provides examples (the full list of settings and values is found inside Reactor in the JSON editor. The examples are rendered onto tiles of the size 128x72 and 128x36 pixels with color capability (these tiles are found on the Blue Pill).

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "ShrinkMode": "Ignore",
        "DataSource": "Icon",
        "IconFile": "icons/64x32bw/Fun-SpaceCraft.png"
    }
}
```



Inclusion of an embedded icon.

```
"Constants": {
    "TextSample": {
        "Values": [
            "1"
        ],
        "Labels": [
            "Camera 1"
        ]
    }
},
"FeedbackDefault": {
    "DisplayGraphics": {
        "ShrinkMode": "Ignore",
        "Title": "A: {Behavior:Const:TextSample:Label}",
        "SolidHeaderBar": "On",
        "DataSource": "Icon",
        "IconFile": "icons/64x32bw/Fun-SpaceCraft.png",
        "ImageVerticalAlign": "Bottom",
        "ImageHorizontalAlign": "Right"
    }
}
```



Adding a title with solid bar behind. Title includes a value from a constant (IO reference).
Notice how the title changes the available area for the image underneath. The icon is aligned to the bottom and right (other options include Center by default and Top/ Left of course)

```
"Constants": {
    "BgColor": {
        "Values": [
            "#336699"
        ]
    }
},
"FeedbackDefault": {
    "DisplayGraphics": {
        "ShrinkMode": "Ignore",
        "Title": "My title",
        "SolidHeaderBar": "On",
        "PixelColorCode": "LIGHTGRAY",
        "BackgroundColorCode": "{Behavior:Const:BgColor}",
        "DataSource": "Icon",
        "IconFile": "icons/64x32bw/Fun-SpaceCraft.png",
        "ImageFilters": "Invert",
        "ImageVerticalAlign": "Bottom",
        "ImageHorizontalAlign": "Right",
        "ImageBlendmode": "Screen"
    }
}
```

In this example the Pixel and Background colors have been set up. You can use one of the color labels from Reactor or an HTML color definition and even fetch it from an IO reference as we do in this example, picking it up from the constant "BgColor".
An invert filter is applied to the image and then the blend mode is set to Screen so the whites are applied to the background but not the blacks. Other blendmodes include Multiply and Alpha

```
...
        "ImageBlendmode": "Screen",
        "ForceImageMode": "Gray"
    }
}
```

The color mode is usually defined by the tile the graphic is generated for, but you can force it to Gray if you need to. (RGB was the default for this tile)

```
...
        "ImageBlendmode": "Screen",
        "ForceImageMode": "Mono"
    }
}
```

The color mode is usually defined by the tile the graphic is generated for, but you can force it to Mono if you need to.

```
...
        "ImageVerticalAlign": "Bottom",
        "ImageHorizontalAlign": "Right",
        "ImageFitting": "Fit",
        "ImageBlendmode": "Screen"
    }
}
```

You can force the image to fit into the available area using the "Fit" function.

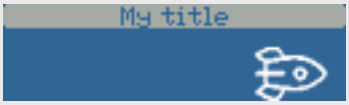
```
...
        "ImageVerticalAlign": "Bottom",
        "ImageHorizontalAlign": "Right",
        "ImageFitting": "Fill",
        "ImageBlendmode": "Screen"
    }
}
```

If you want to fill the entire image area with the image, use the "Fill" function.

```
...
        "ImageVerticalAlign": "Bottom",
        "ImageHorizontalAlign": "Right",
        "ImageFitting": "Stretch",
        "ImageBlendmode": "Screen"
    }
}
```

The "Stretch" function will force the image into the entire area and is likely to scale it out of proportions.

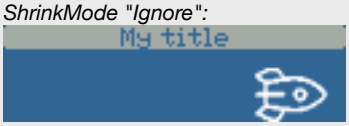
```
...
        "ImageVerticalAlign": "Bottom",
        "ImageHorizontalAlign": "Right",
        "ImageFitting": "0x20",
        "ImageBlendmode": "Screen"
    }
}
```



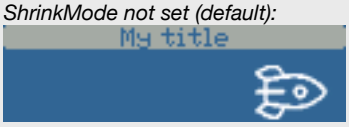
The ImageFitting field can also take a value on the form (W)x(H) to scale to a specific size. If one of the dimensions are set to zero (like W in this case) it will be scaled proportionally to the other dimension (here: 20 pixels height)

```
...
        "ImageVerticalAlign": "Bottom",
        "ImageHorizontalAlign": "Right",
        "ImageFitting": "0x20",
        "ImageBlendmode": "Screen",
        "ShrinkMode": ""
    }
}
```

ShrinkMode "Ignore":



ShrinkMode not set (default):

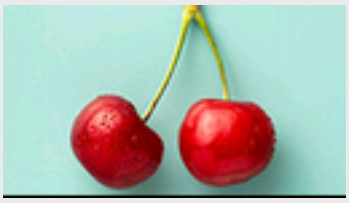


Adjacent tiles on SKAARHOJ controller displays may have shrink-bits set for their bottom or right edges. This helps to leave a single pixel strip empty of content so tiles are visually distinct from each other. In some cases you may want to disable this border to fille the entire area of a tile. ShrinkMode "Ignore" disables this for image rendering.

Data Sources

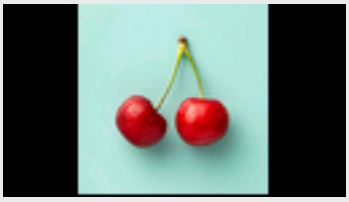
The table below demonstrates the various data sources that exists for images.

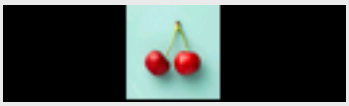
```
"FeedbackDefault": {
    "DisplayGraphics": {
        "DataSource": "Icon",
        "IconFile": "icons/pictures/cherries.jpg",
        "ImageFitting": "Fill"
    }
}
```



Icon - Fill
Example of icon type, image fitting set to fill. Notice the black line in the bottom - this is the 1 pixel shrink-border to the neighbouring tile.

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "DataSource": "Icon",
        "IconFile": "icons/pictures/cherries.jpg",
        "ImageFitting": "Fit"
    }
}
```





Icon - Fit
Example of icon type, image fitting set to fit.

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "DataSource": "Inline",
        "InlineImage":
"png::iVBORw0KGgoAAAANSUhEUgAAAEAAAAAgCAIAAAAt/
+nTAAAAGXRFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAA2hpVFh0WE1MOm
NvbS5hZG9iZS54bXAAAAAAADw/eHBhY2tldCBiZWdpbj0i77u/
IiBpZD0iVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkIj8+IDx4OnhtcG1ldGEgeG1sbnM6e
D0iYWRvYmU6bnM6bWV0YS8iIHg6eG1wdGs9IkFkb2JlIFhNUCBDb3JlIDUuMy1jMDExID
Y2LjE0NTY2MSwgMjAxMi8wMi8wNi0xNDo1NjoyNyAgICAgICAgIj4gPHJkZjpSREYgeG1
sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMj
Ij4gPHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wTU09Imh0dHA6L
y9ucy5hZG9iZS5jb20veGFwLzEuMC9tbS8iIHhtbG5zOnN0UmVmPSJodHRwOi8vbnMuYW
RvYmUuY29tL3hhcC8xLjAvc1R5cGUvUmVzb3VyY2VSZWYjIiB4bWxuczp4bXA9Imh0dHA
6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8iIHhtcE1NOk9yaWdpbmFsRG9jdW1lbnRJRD0i
eG1wLmRpZDo4RUJGNzcxQjEyMjA2ODExODA4M0IwMjBFNkI1OUYwNSIgeG1wTU06RG9jd
W1lbnRJRD0ieG1wLmRpZDpFMkMzNUI4NTk4QjcxMUU3ODE0MUFDQTJFM0ZCNzlDRiIgeG
1wTU06SW5zdGFuY2VJRD0ieG1wLmlpZDpFMkMzNUI4NDk4QjcxMUU3ODE0MUFDQTJFM0Z
CNzlDRiIgeG1wOkNyZWF0b3JUb29sPSJBZG9iZSBQaG90b3Nob3AgQ1M2IChNYWNpbnRv
c2gpIj4gPHhtcE1NOkRlcml2ZWRGcm9tIHN0UmVmOmluc3RhbmNlSUQ9InhtcC5paWQ6O
TRCRjc3MUIxMjIwNjgxMTgwODNCMDIwRTZCNTlGMDUiIHN0UmVmOmRvY3VtZW50SUQ9In
htcC5kaWQ6OEVCRjc3MUIxMjIwNjgxMTgwODNCMDIwRTZCNTlGMDUiLz4gPC9yZGY6RGV
zY3JpcHRpb24+IDwvcmRmOlJERj4gPC94OnhtcG1ldGE+IDw/
eHBhY2tldCBlbmQ9InIiPz4EUSuEAAABOklEQVR42sxY2w6FIAwTsv//
Zc6JJsZMHV03wD34YHCspe5Caa1tHiulPL73+smy6gr9LXoD2GgTgvU72ceC/
3P+OUgw9PP9qhNAJdR2sxcsEZLgPwAoDwRDotL6YV0DelM/
bXEkEK90lFfnhpMIDEfesGHE6wkHw5f4bDmRFMZ8Vk4YiX+hkplXrnX7gClGXBhkTohIQ
bxWQzxr1/
nRgx0heA6SuH233UAaJ29XUtMrgPqE8KCgzpaQ4jWSr5BvxeXioISsOLtmTlINJ5kSyj0
TJDIE4YI0inDPleRJALox0Q2F0GoeUSUItxLfld4bnFfTJjJiwWMSy50HJHLKHKrciczR
Stw3AwkeOhNLkMKhwSUXMhvMR++F7qNG7mQ86V5IXVqpv2IV99Gh/gv0uwuZ0Zatul7/
CTAABJXYUo2pweYAAAAASUVORK5CYII="
    }
}
```


Inline
The 64x32 pixel space craft icon from before. The base64 string is generated by a simple command on the command line (Mac/Linux):

```
base64 icons/64x32bw/
Fun-SpaceCraft.png
```

Due to the large data size embedded in configuration text files, this is not recommended on a large scale basis.

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "Title": "Scan Me!",
        "BackgroundColorCode": "ROSE",
        "DataSource": "QRcode",
        "QRcodeContent": "http://www.skaarhoj.com",
        "QRcodePixelSize": 2,
        "ImageBlendmode": "Multiply"
    }
}
```

QRcode
Generates a QRcode from a text string and with a given pixel size. By default it's rendered black on white, but in this case we set a rose-colored background and use Multiply blend mode to impose it onto the background.
Also notice the header with default white color and no solid header bar. It's generally a bad idea to use image fitting (fit, fill, stretch etc) with pixel-exact graphics, so adjust the QRcodePixelSize instead to fit your display.

(TODO: IO reference) - thumbnail from PTZ

(TODO: IO reference) - thumbnail from Device Core

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "DataSource": "Generator",
        "GeneratorConfig": "Clip:Cam1:1"
    }
}
```

Generator - Clip
Sample clips at a given frame rate

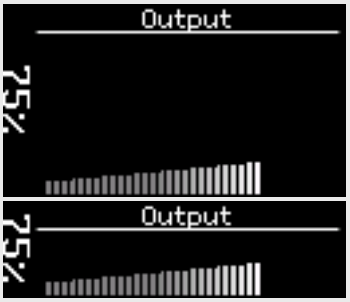
```
"FeedbackDefault": {
    "DisplayGraphics": {
        "DataSource": "Generator",
        "GeneratorConfig": "Wedge"
    }
}
```

Generator - Wedge
Gray wedge

Data Sources - Widgets

Widgets generates a complex, often dynamic graphic use in specific cases on a SKAARHOJ panel, such as a VU meter or a strength indicator for a T-bar display.

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "DataSource": "Widget",
        "Widget": {
            "Type": "Strength",
            "Subtype": "Tbar",
            "Title": "Output",
            "Value": "75%",
            "Data1": {
                "Raw": "750"
            }
        }
    }
}
```



Widget - Strength
Generates a strength meter in grayscale (Work-in-progress). Designed to be rotated 90 CCW

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "DataSource": "Widget",
        "Widget": {
            "Type": "VUMeter",
            "Subtype": "Fixed176x32",
            "Title": "Output",
            "RangeMapping":
"0,77,154,231,308,385,462,538,615,692,769,846,923,1000",
            "Data1": {
                "Raw": "90"
            },
            "Data2": {
                "Raw": "768"
            },
            "Data3": {
                "Raw": "450"
            },
            "Data4": {
                "Raw": "900"
            }
        }
    }
}
```

Widget - VU Meter
A VU meter for audio. Has a fixed size of 176x32 pixels because of the background image (Work-in-progress). Designed to be rotated 90 CCW
Data1 and Data2 is normalized values from 0-1000 for the L+R bars. Data3 and Data4 are the peak points.
The RangeMapping can be used to set which values from 0-1000 will comply with the 13 steps on the scale. The example is the default completely linear scale. Crafting the RangeMapping values carefully can make the VU meter perfectly match the values from any host system for accurate measuring.
(The Widget presented above is rendered at exactly 176x32 and not on any of the Blue Pill tiles.)
Widget has a BackgroundFile field that can specify an alternative background PNG.

Data Sources - Compositions

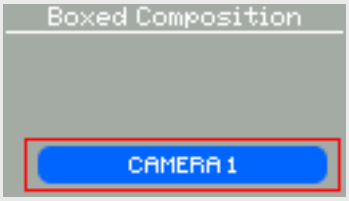
Compositions is a particularly complex data source since it represents a layered structure that can include other data sources as well as graphical primitives and text. The table below demonstrates examples:

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "Title": "My Title Line",
        "BackgroundColorCode": "LIGHTGRAY",
        "DataSource": "Composition",
        "Composition": {
            "Type": "Graphics",
            "Graphics": [
                {
                    "Type": "Text",
                    "Text": "Camera 1",
                    "TextSize": 20,
                    "TextFont": "NotoSans-Bold"
                },
                {
                    "Type": "Rectangle",
                    "ColorCode": "#0066ff",
                    "RoundedCorner": 10,
                    "X": 5,
                    "Y": 3,
                    "W": -10,
                    "H": -6
                }
            ]
        },
        "ImageBlendmode": "Alpha",
        "ShrinkMode": "Ignore"
    }
}
```

Basic Composition
Renders true type text onto a blue rectangle with rounded corners. The text is centered in the canvas both horizontally and vertically. The Rectangle is defined from upper left corner with X and Y starting position being 5,3. Since the W and H of the rectangle is negative, it will pick up the canvas width/height and subtract these numbers.

Notice that ImageBlendMode is set to Alpha since a composition is transparent by default, otherwise it would render black background all over the canvas area (like here:):

```
"Variables": {
    "ShowComposition": {
        "Name": "Show Composition",
        "Description": "If equal to on, the composition will render",
        "OptionList": [
            {
                "ValueID": "on",
                "Name": "Show"
            },
            {
                "ValueID": "off",
                "Name": "Hide"
            }
        ],
        "Default": [
            "on"
        ]
    }
},
"IOReference": {},
"FeedbackDefault": {
    "DisplayGraphics": {
        "Title": "Boxed Composition",
        "BackgroundColorCode": "LIGHTGRAY",
        "DataSource": "Composition",
        "Composition": {
            "ActiveIf": "Var:ShowComposition == on",
            "Box": {
                "Width": -10,
                "Height": 20,
                "VerticalAlign": "Bottom",
                "HorizontalAlign": "Right",
                "OffsetX": -2,
                "OffsetY": -2
            },
            "ShowBox": true,
            "Type": "Graphics",
            "Graphics": [
                {
                    "Type": "Text",
                    "ColorCode": "WHITE",
                    "Text": "Camera 1",
                    "TextFont": "Small"
                },
                {
                    "Type": "Rectangle",
                    "ColorCode": "#0066ff",
                    "RoundedCorner": 5,
                    "X": 5,
                    "Y": 3,
                    "W": -10,
                    "H": -6
                }
            ]
        },
        "ImageBlendmode": "Alpha",
        "ShrinkMode": "Ignore"
    }
}
```

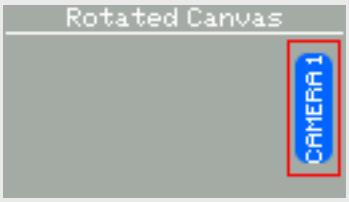


Composition canvas

A composition has a canvas which by default is equal to the target it is placed on (in this case that would be the gray area under the white line of the tile). But you can change it's width, height, offset and alignment like here where the width is tile-width minus 10, height is fixed, it's aligned to the bottom right and offset -2,-2 from the edges.
The ShowBox property is true, which renders the red border around the canvas of the composition as a temporary help to the designer.
The text font is changed to "Small" which is a truetype pixel font that renders sharp edges at sizes 8 (default), 16, 24 etc.

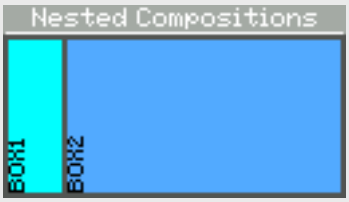
Compositions have an ActiveIf field which decides if it's rendered or not. An empty field equals true. In this case we are using a variable to determine whether to render it or not.

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "Title": "Rotated Canvas",
        "BackgroundColorCode": "LIGHTGRAY",
        "DataSource": "Composition",
        "Composition": {
            "Box": {
                "CanvasOrientation": "CCW",
                "Width": -10,
                "Height": 20,
                "VerticalAlign": "Bottom",
                "HorizontalAlign": "Right",
                "OffsetX": -2,
                "OffsetY": -2
            },
            "ShowBox": true,
            "Type": "Graphics",
            "Graphics": [
                {
                    "Type": "Text",
                    "ColorCode": "WHITE",
                    "Text": "Camera 1",
                    "TextFont": "Small"
                },
                {
                    "Type": "Rectangle",
                    "ColorCode": "#0066ff",
                    "RoundedCorner": 5,
                    "X": 5,
                    "Y": 3,
                    "W": -10,
                    "H": -6
                }
            ]
        },
        "ImageBlendmode": "Alpha",
        "ShrinkMode": "Ignore"
    }
}
```



Rotation of the canvas
The only change to the previous configuration (except the removal of the ActiveIf field) is that CanvasOrientation is set to CCW (Counter Clock Wise). This changes the reference for alignment, offset as well as width and height. For example the width is now (original height)-10. Compare it to the previous example to see the similarity.

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "Title": "Nested Compositions",
        "BackgroundColorCode": "LIGHTGRAY",
        "DataSource": "Composition",
        "Composition": {
            "Box": {
                "CanvasOrientation": "CCW"
            },
            "BackgroundColorCode": "DARKGRAY",
            "Type": "Layers",
            "Layers": [
                {
                    "Box": {
                        "Width": -4,
                        "Height": 20,
                        "VerticalAlign": "Top",
                        "OffsetY": 2
                    },
                    "BackgroundColorCode": "CYAN",
                    "Type": "Graphics",
                    "Graphics": [
                        {
                            "Type": "Text",
                            "Text": "Box1",
                            "TextHorizontalAlign": "Left",
                            "TextVerticalAlign": "Top",
                            "TextFont": "Small"
                        }
                    ]
                },
                {
                    "Box": {
                        "Width": -4,
                        "Height": -26,
                        "VerticalAlign": "Bottom",
                        "OffsetY": -2
                    },
                    "BackgroundColorCode": "ICE",
                    "Type": "Graphics",
                    "Graphics": [
                        {
                            "Type": "Text",
                            "Text": "Box2",
                            "TextHorizontalAlign": "Left",
                            "TextVerticalAlign": "Top",
                            "TextFont": "Small"
                        }
                    ]
                }
            ]
        },
        "ShrinkMode": "Ignore"
    }
}
```



Nested compositions
In this example the composition is of type "Layers" which holds another two compositions.
The main composition is rotated CCW and has a dark gray background color (so we don't need Alpha blend mode here).

The first nested composition is 20 pixels high and width-4, aligned to the top and offset 2 pixels down. Since the main canvas is rotated CCW, all of this makes sense if you "turn your head" counter clock wise too to see it from that reference. The background of this box is cyan and we render a basic text "Box1" in the top left corner

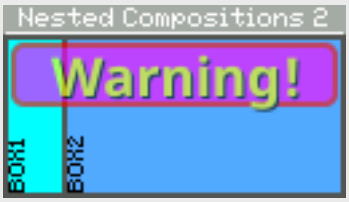
The second nested composition is bottom aligned, has a height equal to the main composition minus 26 pixels (which accommodates 2 pixels borders three places and the 20 pixel height of the first nested composition). In this composition we also render a basic text "Box2" in the upper left corner.

Nested compositions and boxes helps us to divide the tiles into flexible sections where we can place any fixed or dynamic content from Reactor.

```
"FeedbackDefault": {
    "DisplayGraphics": {
        "Title": "Nested Compositions 2",
        "BackgroundColorCode": "LIGHTGRAY",
        "DataSource": "Composition",
        "IOReference": {},
        "Composition": {
            "Box": {
                "CanvasOrientation": "CCW"
            },
            "BackgroundColorCode": "DARKGRAY",
            "Type": "Layers",
            "Layers": [
                {
                    "Box": {
                        "CanvasOrientation": "CW",
                        "Height": 30,
                        "VerticalAlign": "Top"
                    },
                    "Type": "Graphics",
                    "Graphics": [
                        {
                            "Type": "Text",
                            "ColorCode": "YELLOW",
                            "Text": "Warning!",
                            "TextSize": 20,
                            "TextFont": "NotoSans-Bold"
                        },
                        {
                            "Type": "Text",
                            "ColorCode": "BLACK",
                            "OffsetX": 1,
                            "OffsetY": 1,
                            "Text": "Warning!",
                            "TextSize": 20,
                            "TextFont": "NotoSans-Bold"
                        },
                        {
                            "Type": "Rectangle",
                            "ColorCode": "PINK",
                            "LineWidth": 2,
                            "StrokeColorCode": "RED",
                            "RoundedCorner": 4,
                            "X": 4,
                            "Y": 4,
                            "W": -8,
                            "H": -8
                        }
                    ],
                    "Transparency": 40
                },


.... (Insert the two boxes from prev example here) ...


            ]
        },
        "ShrinkMode": "Ignore"
    }
}
```



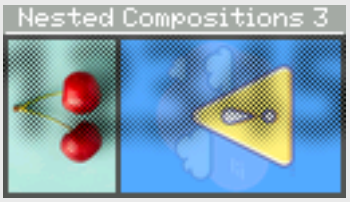
Overlapping composition layers

In the basic example the two nested compositions didn't overlap, but if they did, the top one would occlude the bottom one.
In this example a new composition is added that spans across the two boxes from before. The box of that composition is rotated CW which brings it "back" to the original orientation. The height is set to 30 and it's top-aligned. Implicitly the width matches the parent composition.

The composition itself is of type "Graphics" and has two text items and a rectangle at the bottom.
The text items are simply rendering the same text, but with a black shadow under, offset 1,1 pixel.
The rectangle is with pink fill, red stroke of 2 pixels and with rounded corners. Width and height is picked up from the canvas and 8 pixels are subtracted.

Finally, notice the Transparency for the composition is set to 40% (0-100, 100=completely transparent)

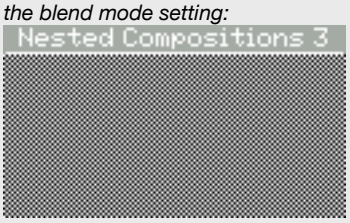
```
"FeedbackDefault": {
    "DisplayGraphics": {
        "Title": "Nested Compositions 3",
        "BackgroundColorCode": "LIGHTGRAY",
        "DataSource": "Composition",
        "Composition": {
            "Box": {
                "CanvasOrientation": "CCW"
            },
            "BackgroundColorCode": "DARKGRAY",
            "Type": "Layers",
            "Layers": [
                {
                    "Mask": {
                        "Box": {
                            "CanvasOrientation": "CW"
                        },
                        "BackgroundColorCode": "#000000",
                        "Type": "Graphics",
                        "Graphics": [
                            {
                                "Type": "Text",
                                "ColorCode": "WHITE",
                                "Text": "12345",
                                "TextVerticalAlign": "Top",
                                "TextSize": 50,
                                "TextFont": "NotoSans-Bold"
                            }
                        ],
                        "ImageFilters": "GaussianBlur=3"
                    },
                    "ShowMask": false,
                    "DisableMask": false,
                    "Type": "Image",
                    "Image": {
                        "DataSource": "Icon",
                        "IconFile": "icons/64x32bw/Pattern-
CheckerFine-256x256.png",
                    },
                    "Blendmode": "Multiply"
                },
                {
                    "Box": {
                        "Width": -4,
                        "Height": 40,
                        "VerticalAlign": "Top",
                        "OffsetY": 2
                    },
                    "BackgroundColorCode": "CYAN",
                    "Type": "Image",
                    "Image": {
                        "DataSource": "Icon",
                        "IconFile": "icons/pictures/cherries.jpg",
                        "Fitting": "Fill"
                    }
                },
                {
                    "Box": {
                        "Width": -4,
                        "Height": -46,
                        "VerticalAlign": "Bottom",
                        "OffsetY": -2
                    },
                    "BackgroundColorCode": "ICE",
                    "Type": "Image",
                    "Image": {
                        "DataSource": "Icon",
                        "IconFile": "icons/Scenarium/2997979.png",
                        "Fitting": "Fit"
                    }
                }
            ]
        },
        "ShrinkMode": "Ignore"
    }
}
```



Masks and image filters
In this example the underlying two boxes are still in place, but the height is adjusted a bit and they are filled with images instead of text labels. In the first case, the cherry picture is instructed to fill the entire composition (centered by default) while the second example is fitting the icon in. The icon is an embedded PNG file with transparency so it blends in with the ice colored background.

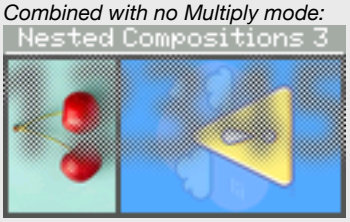
The top layer is a checker box image (alternating black and white pixels) which is multiplied onto the background (so white colors become transparent) through a mask. The mask image is the real tricky part: It's generated by a Text type graphics on a canvas rotated back to normal, and then blurred with an image filter.

To understand the components better, this is what you get with DisableMask set to true (just the checkerboard image) and remove the blend mode setting:

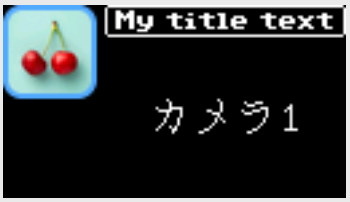


If ShowMask is set to true instead you see this:

Combined with no Multiply mode:



```
"FeedbackDefault": {
    "DisplayGraphics": {
        "DataSource": "Composition",
        "Composition": {
            "Box": {},
            "Type": "Layers",
            "Layers": [
                {
                    "Box": {
                        "Width": 35,
                        "Height": 35,
                        "VerticalAlign": "Top",
                        "HorizontalAlign": "Left"
                    },
                    "Type": "Layers",
                    "Layers": [
                        {
                            "Box": {},
                            "Type": "Graphics",
                            "Graphics": [
                                {
                                    "Type": "Rectangle",
                                    "LineWidth": 2,
                                    "StrokeColorCode": "ICE",
                                    "RoundedCorner": 6,
                                    "X": 1,
                                    "Y": 1,
                                    "W": -2,
                                    "H": -2
                                }
                            ]
                        },
                        {
                            "Box": {},
                            "Mask": {
                                "Box": {},
                                "BackgroundColorCode": "#000000",
                                "Type": "Graphics",
                                "Graphics": [
                                    {
                                        "Type": "Rectangle",
                                        "ColorCode": "WHITE",
                                        "RoundedCorner": 6,
                                        "X": 1,
                                        "Y": 1,
                                        "W": -2,
                                        "H": -2
                                    }
                                ]
                            },
                            "Type": "Image",
                            "Image": {
                                "DataSource": "Icon",
                                "IconFile": "icons/pictures/cherries.jpg",
                                "Fitting": "Fill"
                            }
                        }
                    ]
                },
                {
                    "Box": {
                        "Width": -38,
                        "Height": 12,
                        "VerticalAlign": "Top",
                        "HorizontalAlign": "Right"
                    },
                    "Type": "Layers",
                    "Layers": [
                        {
                            "Box": {},
                            "Type": "MonoText",
                            "MonoText": {
                                "Text": "My title text",
                                "FontFace": 1,
                                "OffsetY": 2,
                                "ColorCode": "WHITE"
                            }
                        },
                        {
                            "Box": {},
                            "Type": "MonoRect",
                            "MonoRect": {
                                "RoundedCorner": 1,
                                "ColorCode": "WHITE"
                            }
                        }
                    ]
                },
                {
                    "Box": {
                        "Width": -38,
                        "Height": -12,
                        "VerticalAlign": "Bottom",
                        "HorizontalAlign": "Right"
                    },
                    "Type": "Graphics",
                    "Graphics": [
                        {
                            "Type": "Text",
                            "ColorCode": "WHITE",
                            "Text": "カメラ1",
                            "TextFont": "Unifont"
                        }
                    ]
                }
            ]
        },
        "ImageBlendmode": "Alpha"
    }
}
```



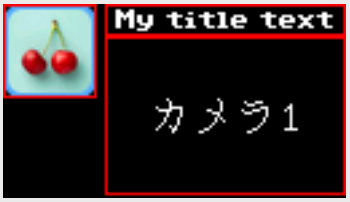
This example demonstrates a "typical" complex composition with nested layers.

The first layer defines a box of 35x35 pixels (top/left aligned) and inside this layer (composition) we will place the image and a border around it as nested layers. The ice blue rectangle with 6 pixel rounded corners and a line width of 2 has X,Y,W,H set so the double line width is accommodated. On the next layer the image itself is placed with a fill scaling but masked by another rounded corner rectangle, this time filled with white instead of stroked with ice blue.

Another box for the title is defined and inside of that a MonoText and MonoRect object is placed. These are legacy rendering of text and rounded corner rectangles for monochrome displays and are useful if you desire a style coming from UniSketch's tradition.

The lower box is aligned to bottom right and relative to the parent composition size and contains Japanese text ("Camera 1") rendered with the embedded Unifont (all character sets).

If you set ShowBoxOnTop to true for the three main layers/ compositions you will see the bounding boxes shown with red lines, useful for debugging:



Appendix

Icons

(Todo: List of embedded icon paths, their dimensions and color state to be)

Reactor / UniSketch color codes

- DEFAULT
- OFF
- WHITE
- WARM
- RED
- ROSE
- PINK
- PURPLE
- AMBER
- YELLOW
- DARKBLUE
- BLUE
- ICE
- CYAN
- SPRING
- GREEN
- MINT
- LIGHTGRAY
- DARKGRAY
- BLACK

HTML colors on the form #ff6600 or #f60 are often allowed too.

Image Filters

A comma separated, ordered list of image filters:

- Grayscale
- FlipHorizontal
- FlipVertical
- Invert
- Sharpen=[0:10] , example "Sharpen=5"
- GaussianBlur=[0:10]
- Threshold=[0:100, 50 is default]
- Saturation=[-100:500]
- Contrast=[-100:100, default 0]
- Brightness=[-100:100, default 0]
- Gamma=[0.0:2.0, default 1]
- Colorize=[Hue 0:360];[Saturation 0:100];[Percentage 0:100]
- Hue=[-180:180]

Embedded Fonts

Reactor has various build-in true type fonts you can use. They are mostly pixel fonts selected for their ability to render well in small sizes on monochrome displays. Below they are compared in their default sizes:

The table below provides more information and evaluation of the fonts:

	Extended latin & Cyrillic	Japanese, Chinese, Korean etc.	Special Characters	Pixel Font	Notes	Attribution
Small	*None*	*None*	*Some omissions, like : ; &*	*Yes*	*Caps*	*Small Pixel by Dark MaxX (100% free)*
PixelArt	*None*	*None*	*Many omissions*	*Yes*	*Small Caps*	*Pixel-Art by Parasite (Demo)*
m3x6	*None*	*None*	*OK*	*Yes*	*Smallest*	*Daniel Linssen, Creative Commons*
m5x7	*No Cyrillic*	*None*	*OK*	*Yes*		*Daniel Linssen, Creative Commons*
PixelType	*None*	*None*	*OK*	*Yes*	*Nicely narrow*	*Pixeltype by TheJman0205 (100% free)*
Pixolletta	*No Cyrillic*	*None*	*OK*	*Yes*		*Pixolletta 8px by Neuland_Ink (100% free)*
PixelArial	*No Cyrillic, issues with ÆØæø*	*None*	*OK*	*Yes*		*Pixel Arial by Max (100% free)*
PixelArial-Bold	*No Cyrillic, issues with ÆØæø*	*None*	*OK*	*Yes*		*Pixel Arial by Max (100% free)*
DogicaPixel	*No Cyrillic*	*None*	*OK*	*Yes*		*Dogica by Roberto Mocci (OFL)*
DogicaPixel-Bold	*No Cyrillic*	*None*	*OK*	*Yes*		*Dogica by Roberto Mocci (OFL)*
PixelOperatorSC	*No Cyrillic*	*None*	*OK*	*Yes*		*Pixel Operator by Jayvee Enaguas (OFL)*
PixelOperator	*No Cyrillic*	*None*	*OK*	*Yes*		*Pixel Operator by Jayvee Enaguas (OFL)*

	Extended latin & Cyrillic	Japanese, Chinese, Korean etc.	Special Characters	Pixel Font	Notes	Attribution
PixelOperator-Bold	*No Cyrillic*	*None*	*OK*	*Yes*		*Pixel Operator by Jayvee Enaguas (OFL)*
Maga-Sans	*None*	*None*	*OK*	*Yes*	*Small Caps*	*Maga Sans by TypeAntoniolu gb (100% free)*
SuperStar	*OK*	*None*	*OK*	*Yes*	*Caps*	*Superstar by memesbruh03 (100% free)*
Pixellari	*No Cyrillic, issues with Ø rendering*	*None*	*OK*	*Yes*		*Pixellari by Zacchary Dempsey-Plante (100% free)*
Unifont	*OK*	*OK*	*OK*	*Yes*	*Very large font file, may affect performance*	*GPL*
437Win	*No Cyrillic, issues with Ø*	*None*	*OK*	*Yes*		
NotoSans-Bold	*OK*	*None*	*OK*	*No*		*Google, OFL*
NotoSans-BoldItalic	*OK*	*None*	*OK*	*No*		*Google, OFL*
NotoSans-Italic	*OK*	*None*	*OK*	*No*		*Google, OFL*
NotoSans-Regular	*OK*	*None*	*OK*	*No*		*Google, OFL*

When a font is characterized as a pixel font it will render clean at the size shown in the image above (for example 8, 16, 10 etc). The sizes shown are the default sizes that will be used unless you specify otherwise.

	Demo
Small	SMALL, 8 THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 <>,.-_+!"#%'()=?$^'@
PixelArt	PIXELART, 8 THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 +MORE.US 1085.39 =MONEY.EH? 10652.54 <>,.-_ '!?^'@
m3x6	m3x6, 16 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 < > , . ; : - _ " ! " # % & / ( ) = ? $ ^ ' @
m5x7	m5x7, 16 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅæøåäüöÄÜÖ 1 2 < > , . ; : - _ " ! " # % & / ( ) = ? $ ^ ' @
PixelType	PixelType, 16 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 <>,.;:-_*!"#%&/()=?$^'@

	Demo
Pixolletta	Pixolletta, 10 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅæøåäüöÄÜÖ □□□□ 1 □□□□□□ 2 □□□2 □□□ 2 □□ 2 □□□□□ 2 □□□□□ 2 □□□□□□□□ 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @
PixelArial	PixelArial, 8 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 0±ÅÄÄäüöÄÜÖ □□□ 1 □□□□□ 2 □□2 □□ 2 □□ 2 □□□□ 2 □□□□ 2 □□□□□□ 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @
PixelArial-Bold	**PixelArial-Bold, 8 The Quick Brown Fox Jumps Over The Lazy Dog** **the quick brown fox jumps over the lazy dog** **THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG** **SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54** **0±ÅÄÄäüöÄÜÖ □□□ 1 □□□□□ 2** **□□2 □□ 2 □□ 2 □□□□ 2 □□□□ 2 □□□□□□ 2** **< > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @**
DogicaPixel	DogicaPixel, 8 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅæøåäüöÄüö □□□ 1 □□□□ 2 □□2 □□ 2 □□ 2 □□□□ 2 □□□□ 2 □□□□□ 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @
DogicaPixel-Bold	**DogicaPixel-Bold, 8 The Quick Brown Fox Jumps Over The Lazy Dog** **the quick brown fox jumps over the lazy dog** **THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG** **SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54** **ÆØÅæøåäüöÄüö □□□ 1 □□□□ 2** **□□2 □□ 2 □□ 2 □□□□ 2 □□□□ 2 □□□□□ 2** **< > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @**
PixelOperatorSC	PixelOperatorSC, 16 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅæøåäüöÄÜÖ □□□□ 1 □□□□□□ 2 □□□2 □□□ 2 □□ 2 □□□□□ 2 □□□□□ 2 □□□□□□□□ 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @
PixelOperator	PixelOperator, 16 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅæøåäüöÄÜÖ □□□□ 1 □□□□□□ 2 □□□2 □□□ 2 □□ 2 □□□□□ 2 □□□□□ 2 □□□□□□□□ 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @
PixelOperator-Bold	**PixelOperator-Bold, 16 The Quick Brown Fox Jumps Over The Lazy Dog** **the quick brown fox jumps over the lazy dog** **THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG** **SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54** **ÆØÅæøåäüöÄÜÖ □□□□ 1 □□□□□□ 2** **□□□2 □□□ 2 □□ 2 □□□□□ 2 □□□□□ 2 □□□□□□□□ 2** **< > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @**
Maga-Sans	MAGA-SANS, 8 THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅæ□□□□□□□□ □□□□ 1 □□□□□□ 2 □□□2 □□□ 2 □□ 2 □□□□□ 2 □□□□□ 2 □□□□□□□□ 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @

	Demo
SuperStar	SUPERSTAR, 16 THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅÆØÅÄÜÖÄÜÖ BXIД 1 KAMEPA 2 □□□2 □□□ 2 □□ 2 □□□□□ 2 □□□□□ 2 □□□□□□□□□ 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @
Pixellari	Pixellari, 16 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅæøåäüöÄÜÖ □□□□ 1 □□□□□ 2 □□□2 □□□ 2 □□ 2 □□□□□ 2 □□□□□ 2 □□□□□□□□ 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @
Unifont	Unifont, 16 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅæøåäüöÄÜÖ Bxiд 1 Камера 2 カメラ2 카메라 2 相机 2 กล้อง 2 कैमरा 2 الكاميرا 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @
437Win	437Win, 16 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652 ƒ Åæ åäüöÄÜÖ 1 2 2 2 2 2 2 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @
NotoSans-Bold	NotoSans-Bold, 12 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅæøåäüöÄÜÖ Bxiд 1 Камера 2 □□□2 □□□ 2 □□ 2 □□□□□ 2 कैमरा 2 □□□□□□□□ 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @
NotoSans-BoldItalic	*NotoSans-BoldItalic, 12 The Quick Brown Fox Jumps Over The Lazy Dog* *the quick brown fox jumps over the lazy dog* *THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG* *SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54* *ÆØÅæøåäüöÄÜÖ Bxiд 1 Камера 2* *□□□2 □□□ 2 □□ 2 □□□□□ 2 □□□□□ 2 □□□□□□□□ 2* *< > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @*
NotoSans-Italic	*NotoSans-Italic, 12 The Quick Brown Fox Jumps Over The Lazy Dog* *the quick brown fox jumps over the lazy dog* *THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG* *SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54* *ÆØÅæøåäüöÄÜÖ Bxiд 1 Камера 2* *□□□2 □□□ 2 □□ 2 □□□□□ 2 □□□□□ 2 □□□□□□□□ 2* *< > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @*
NotoSans-Regular	NotoSans-Regular, 12 The Quick Brown Fox Jumps Over The Lazy Dog the quick brown fox jumps over the lazy dog THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG SEND.ME 9567.15 + MORE.US 1085.39 = MONEY.EH? 10652.54 ÆØÅæøåäüöÄÜÖ Bxiд 1 Камера 2 □□□2 □□□ 2 □□ 2 □□□□□ 2 कैमरा 2 □□□□□□□□ 2 < > , . ; : - _ * ! " # % & / ( ) = ? $ ^ ' @