Quick guide to DUIML

DUIML is a XML based language to define UI. The DUIML file has a display side independent description of UI content. The actual UI graphics are illustrated using PNG images.

The DUIML is compiled into internal format before copied it can be used: DUIML and graphics images are compiled into single repository file using DUIML compiler, written in perl duimlc.pl.

use: perl duimlc.pl DUIMLFILE DIRFILE, a repository DIRFILE is generated.

Dui

A mandatory description definition start. The actual UI definition is between start and end tags.

Synopsis: <dui> “ui description” </dui>

Attributes:

“version”

Number to define version number, must be “1”.

Background

Background of the UI.

Synopsis: <background />

Attributes:

- “image”
  - String, a filename relative to directory DUIML file is located.
- “color”
  - Hexadecimal number that defines Background color in hexadecimal mode RRGGBB, e.g. “00ff33”.
    - RR is red component value between 0-FF.
    - GG is green component value between 0-FF.
    - BB is blue component value between 0-FF.
- “placement”
  - “tile”
    - Image is tiled to fill background area.
  - “strecth_all”
    - Image is stretched to fill background area.
  - “strech_aspect”
    - Image is stretched, but its aspect ratio is maintained.
  - “centrify”
    - Image is centrified.

Button

Focusable button that has two states: focused and non-focused.

Synopsis: <button />

Attributes:

“image”

String, a filename relative to directory DUIML file is located.

“image_focused”

String, a filename relative to directory DUIML file is located. Optional. If exists, the button has to be focused at first before command is executed. Button can be accessed using keyboard. If image does not exists, button cannot be focused and command is executed immediately when pointed. Button can be accessed only with pointer.

“command”

String that match to options menu command string. If string is matching, that command should disappear from menu.

Special commands:

“Menu” - opens options menu.

“placement”

“strecth_all”

Image is stretched to fill its rectangle area.

“strech_aspect”

Image is stretched, but its aspect ratio is maintained.

“centrify”

Image is centrified in its rectangle.

"id" String that defines an identifier of this rectangle so it can be accessed from code.

Rectangle

A Generic area on screen, default is transparent element., but may contain an image or be filled.

Synopsis:<rect />

Parameters:

"image"
String, a filename relative to directory DUI file is located.

"color"
Hexadecimal nymber that defines background color in hexadecimal mode AARRGGBB, e.g. "ff00ff33".
AA is alpha (transparency) value between 0-FF.
RR is red component value between 0-FF.
GG is green component value between 0-FF.
BB is blue component value between 0-FF.

"placement"
"strecth_all"
Image is stretched to fill its rectangle area.
"strech_aspect"
Image is stretched, but its aspect ratio is maintained.
"centrify"
Image is centrified in its rectangle.

"id"
String that defines an identifier of this button so it can be accessed from code.

Container (1/2)

The control area is base area which can then split using containers. Each element (button, rect or container) has its own size that is given from its container (You can think that there is always at least implicit “root” container). Element is size is equally divided area by it container. E.g. if there is 150 width screen and three elements, each of those are 50 pixels width.

However containers can set their own comparative size. The size parameter tells how many element sizes its rectangle takes. E.g. if there is 150 width screen and three elements, each of those are 50 pixels width, but one element has size 2, then that element width is 75 pixels and rest are 37 pixels. So container can provide more room for its elements.

Container can be used to set UI layout dynamically using “horizontal” and “reversed” parameters. The container is either landscape or portrait and elements can be drawn from left to right or right left. Some attribute value depends on screen aspect ratio, i.e. is it landscape or horizontal (square is considered as vertical) or soft buttons (i.e. those labels that refers to buttons next to screen) position.

All elements that are between start and end tags are inside of that container.

Synopsis: <container> “container content” </container>

Attributes:

“size”
Set both vertical and horizontal size.

“size_vertical”
Set vertical size.

“size_horizontal”
Set horizontal size.

Container (2/2)

“horizontal”
- “never”
  - Elements are always drawn vertically. (default)
- “always”
  - Elements are always drawn horizontal.
- “portrait”
  - Elements are drawn horizontal if screen proportion is portrait.
- “landscape”
  - Elements are drawn vertically if screen proportion is portrait.
- “button_side”
  - Elements are drawn horizontal if soft buttons buttons are on (left) side.
- “button_bottom”
  - Elements are drawn horizontal if soft buttons are on bottom side.

“reversed”
- “never”
  - Elements are always drawn left to right. (default)
- “always”
  - Elements are always drawn right to left.
- “portrait”
  - Elements are drawn right to left if screen proportion is portrait.
- “landscape”
  - Elements are drawn right to left if screen proportion is landscape.
- “button_side”
  - Elements are drawn right to left if soft buttons are on (left) side.
- “button_bottom”
  - Elements are drawn right to left if soft buttons are on bottom side.

View (1/2)

View is a “dialog”, a sub screen that is viewed on-demand. It defines a frame that can contain elements, typically containers. The view has its size that is defined in percents. That area defines a rectangle of screen.

Synopsis: <view> “view definition” </view>

Attributes:

“size”

The percentage (0-100) that defines size of the view. (default is 100)

“ratio”

Number pair, two numbers separated with comma explicitly defines width and height ratio of view. E.g. “4, 3” defines that height is ¾ of width. (default is screen ratio)

“shadow”

Number pair, two numbers separated with comma defines width and height of shadow in percents. Positive values sets right and bottom shadows and negative left and top shadows.

“shadow_color”

Hexadecimal number that defines background color in hexadecimal mode AARRGGBB, e.g. “ff00ff33”.

AA is alpha (transparency) value between 0-FF.
RR is red component value between 0-FF.
GG is green component value between 0-FF.
BB is blue component value between 0-FF.

View (2/2)

“position-x”
- “center”
  - View is positioned at center. (default)
- “left”
  - View is positioned at left.
- “right”
  - View is positioned at right.

“position-y”
- “center”
  - View is positioned at center. (default)
- “top”
  - View is positioned at top.
- “bottom”
  - View is positioned at bottom.

”id”
- String, an identifier of view that is used to refer view from code.

Example: soft buttons

Software buttons may vary between devices; usually they are located bottom of display , but can be on right side as well. This example show how to button positions can be handled in a single definition. The two soft buttons container starts:

```
<container horizontal="button_bottom" reversed="button_side">
```

If buttons are on bottom side, a container is horizontal and if buttons are on side they are drawn in reversed order (from bottom to top) . E.g. then Exit button is drawn to bottom right in E61 display, but top and right in E70 display if flip is open, even both devices have a landscape display.

Button rectangles have different paddings depending if display is horizontal or vertical.
The left padding rectangle has zero size of view is horizontal.

```
      <container: size_horizontal="0">
            <rect />
      </container>

      <rect image = "menu.png"   placement="strech_all"/>
```

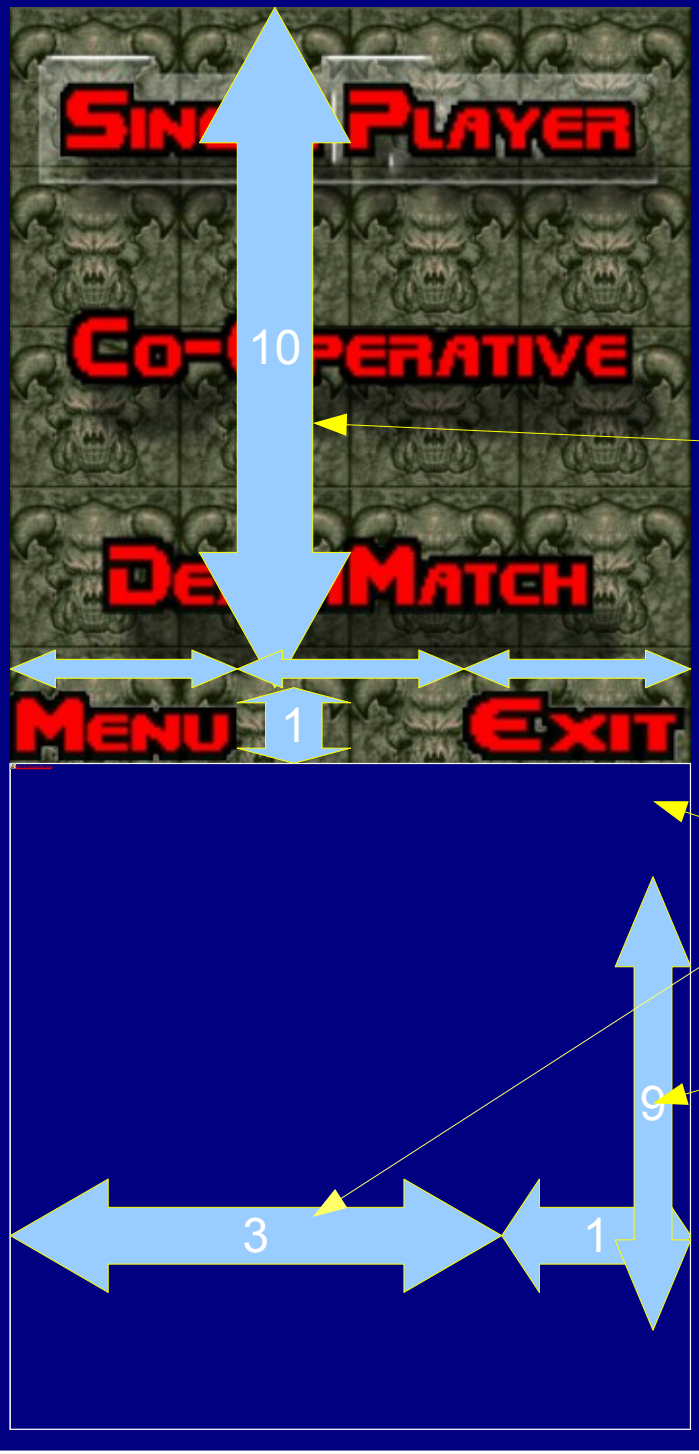
There are a relative padding rectangle between the labels, that takes 9/11 (rectangle * 9 + two buttons) of screen when screen is vertical, otherwise it takes only a default 1/3 (rectangle + two buttons).

```
      <container size_vertical="9">
            <rect />
      </container>

      <rect image = "exit.png"  placement="strech_all" />
```

The right padding rectangle has zero size of view is horizontal.

```
      <container size_horizontal="0">
            <rect />
      </container>

</container>
```

Example: Doom UI 1




```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- C2Doom UI -->
<dui version = "1">
```

Demon images as tiled background.

```
 <background image="demon1.png" color="ffffff" placement="tile"/>
```

Defintion that these containers are next to each others when buttons are at side.

```
 <container horizontal = "button_side">
```

The size of container depends on screen position.

```
  <container size_vertical="10" size_horizontal="3>"
   <button  image="singleplayer1-1.png" image_focused="singleplayer1-1b.png"
      command= "Play" placement="strech_all"/>
   <button image="cooperative1-1.png" image_focused="cooperative1-1b.png"
      command= "Coperative" placement="strech_all" />
   <button image = "deathmatch1-1.png" image_focused="deathmatch1-1b.png"
      command= "Deathmatch" placement="strech_all" />
   <button image = "singleplayer1-1.png" image_focused="singleplayer1-1b.png"
      command= "Start Multiplayer" placement="strech_all" />
  </container>
```

Buttons are as in soft buttons example.

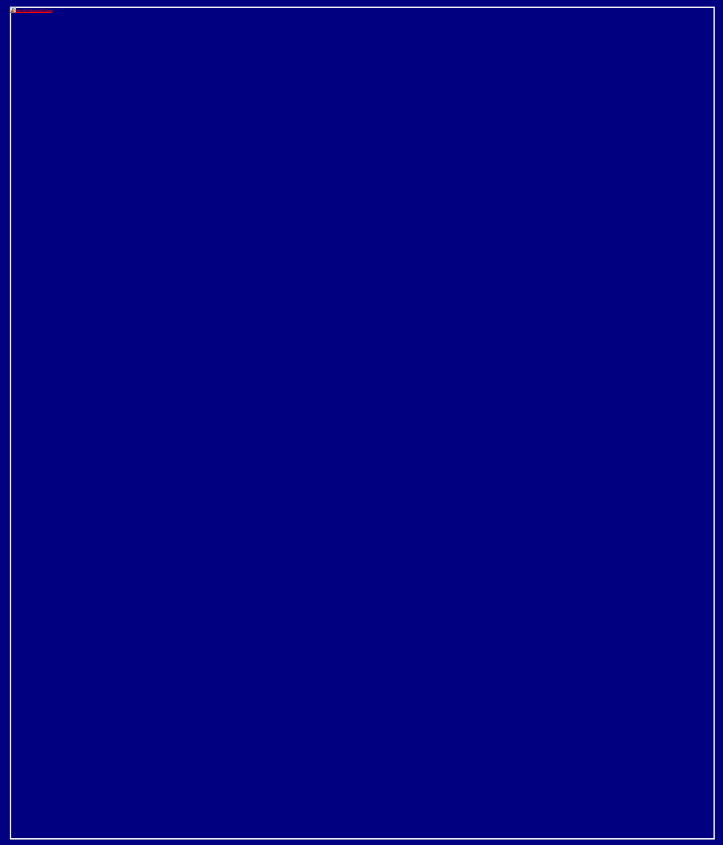
```
  <container horizontal="button_bottom" reversed="button_side">
   <container: size_horizontal="0">
    <rect />
   </container>
```

This buttons cannot be focused, so they are accessed using CBA keys or pointer

```
   <button image = "menu.png"   placement="strech_all" command="Menu"/>
   <container size_vertical="9">
    <rect />
   </container>
   <button image = "exit.png"  placement="strech_all" command="Exit"/>
    <container size_horizontal="0">
     <rect />
    </container>
   </container>
  </container>
```

Continue...

Example: Doom UI 2



View size and other properties.

```
 <view id="server_wait_dialog" size="75" shadow="10,15"
 shadow_color="7f000c0c" ratio="4, 3">
```

Frames are drawn using rects and containers.

```
   <container horizontal="always">
    <rect color="afc6c6c6" />
     <container size="18">
       <rect color="afc7c7c7" />
       <container size = "18">
```

The “canvas” of the dialog. Its drawn dynamically from application.

```
        <rect color="1fe34205" id="progress" />
       </container>
       <rect color="afc8c8c8"/>
     </container>
     <rect color = "afc9c9c9" />
   </container>
 </view>
</dui>
```

doomui.xml (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<dui version = "1">
<!--mandatory first line, 1  is the version number-->
    <background image="demon1.png" color="ffffff" placement="tile" />
        <container horizontal="button_side">
            <container size_vertical="10" size_horizontal="3">
            <button image="singleplayer1-1.png" image_focused="singleplayer1-1b.png" command="Play"
placement="strech_all" />
            <button image="cooperative1-1.png" image_focused="cooperative1-1b.png" command="Coperative"
placement="strech_all" />
            <button image="deathmatch1-1.png" image_focused="deathmatch1-1b.png" command="Deathmatch"
placement="strech_all" />
            <button image="singleplayer1-1.png" image_focused="singleplayer1-1b.png" command="Start
Multiplayer" placement="strech_all" />
        </container >
            <container horizontal="button_bottom" reversed="button_side">
                <container size_horizontal="0">
                <rect  />
            </container >
            <rect image="menu.png" placement="strech_all" />
                <container size_vertical="9">
                <rect  />
            </container >
            <rect image="exit.png" placement="strech_all" />
                <container size_horizontal="0">
                <rect  />
            </container >
        </container >
    </container >
```

doomui.xml (2/2)

```
 <!--cancel button-->
        <view id="cancel_button">
            <container horizontal="button_side">
                <container size_horizontal="3" size_vertical="8">
                <rect  />
            </container >
                <container horizontal="button_bottom" reversed="button_side">
                    <container size_vertical="8" size_horizontal="3">
                    <rect  />
                </container >
                <rect image="exit.png" placement="strech_all" />
                    <container size_horizontal="0">
                    <rect  />
                </container >
            </container >
        </container >
    </view >
<!--end of main view dialogs are inroduced then-->
        <view id="server_wait_dialog" size="75" shadow="10,15" shadow_color="7f000c0c" ratio="4, 3">
            <container horizontal="always">
            <rect color="afc6c6c6" />
                <container size="18">
                <rect color="afc7c7c7" />
                    <container size="18">
                    <rect color="1fe34205" id="progress" />
                </container >
                <rect color="afc8c8c8" />
            </container >
            <rect color="afc9c9c9" />
        </container >
    </view >
</dui>
```