RetroArch - Overlay image configuration

Hans-Kristian Arntzen Daniel De Matteis

January 25, 2013

Purpose

RetroArch supports overlay images for use with hardware accelerated drivers. The purpose of this is to allow some kind of input interface that is mouse/touch oriented.

The overlay image is displayed with transparency over the regular game image, and the user is able to trigger input by pressing on certain parts of the overlay.

Since the overlay is an image, the user should be able to fully configure the look and functionality of this overlay. This allows skinners and themers to go wild.

Configuration

The overlay is described in a config file (*.cfg). The config file uses the same config file syntax as RetroArch itself.

The overlay system supports use of multiple overlays that can be switched out on the fly. Input is not only restricted to gamepad input, but can also work with any input that is bindable in RetroArch, e.g. save states, rewind, load state, etc.

The config file describes:

- Which overlay images to use (.png, .tga, etc).
- Which coordinates on each overlay correspond to which input event.
- The hitbox for each input event, i.e. "size" of the button.
- Where on the screen the overlay should be displayed.

Overlay images

First we configure how many overlays we use, and where they can be found.

```
overlays = 2
overlay0_overlay = overlay_img0.png
overlay1_overlay = overlay_img1.png
```

The paths are relative to where the overlay config is loaded from. If the path is absolute, absolute paths will be used. On Unix-like systems ' /' is recognized as '$HOME'.

Screen placement

By default, the overlay image will be stretched out to fill the whole game image. However, for some overlays, this is not practical.

It is possible to set the placement using for example:

```
overlay0_rect = "0.2,0.3,0.5,0.4"
```

We assume that the game screen has normalized coordinates in X and Y that span from '[0, 0]' in the top-left corner to $[1, 1]$ in the lower-right corner.

This will render the overlay to $x = 0.2, y = 0.3$ with size $width = 0.5, height = 0.4$. The default (stretch to full screen) could be described as such:

```
overlay0_rect = "0.0,0.0,1.0,1.0"
```

Full-screen overlays

By default, overlays will be stretched out to fill game viewport. However, in some cases the aspect ratio of the game causes there to remain large black borders around the game image. It is possible to stretch the overlay to full screen (instead of viewport) by specifying this option:

```
overlay0_full_screen = true
```

Coordinate descriptors

We must also describe where on the overlay image buttons can be found for each overlay. E.g.:

```
overlay0_descs = 3 # Three buttons for this overlay in total
overlay0_desc0 = "a,32,64,radial,10,20"
overlay0_desc1 = "start,100,50,rect,80,10"
overlay0_desc2 = "overlay_next,200,180,radial,40,40"
```

The format is:

```
"button,position_x,position_y,hitbox_type,range_x,range_y"
```

'button' corresponds to the input event being generated. The names are the same as in the general input config, e.g. input_player1_start would translate to start here. overlay_next is a special bind designed to swap to the next overlay, or wrap around to the first one.

'position_x' and 'position_y' are the x and y coordinates in pixels of the source image for the center of the button.

'hitbox_type' describes which type of shape the button has. 'radial' (circle, ellipsis) and 'rect' (rectangular) shapes are supported.

'range_x' and 'range_y' describe the size of the button. The semantics differ slightly for radial and rect hitbox types. For 'radial' shape, 'range_x' and 'range_y' describe the radius in x and y directions in pixels of the source image. For 'rect' shape, the 'range_x' and 'range_y' values represent the distance from center to the edge of the rect in terms of pixels of the source image. E.g. a 'range_x' of 20 would mean the width of the rectangular is 40 pixels in total.

Triggering multiple buttons with one desc

It's possible to trigger multiple buttons (e.g. diagonals) with one overlay desc.

```
overlay0_desc0 = "left|up,32,64,radial,10,20"
```

will trigger both left and up at same time.