Mouse Over Effect on a Button

Introduction

There are plenty of ways to implement some kind of hovering effect. In this tutorial I will describe how do it using the animation framework of orx. In order to do so, you will have to write a few lines in your ini files. I will not provide a running example, but rather some bits and pieces that you can use for your existing project.

Prerequisites

It is supposed that your button consists of two images. One representing the active and on the inactive state.

The configuration

Ok let's suppose you have some kind of game menu, in which you got a “Play” button, that will start the actual game. So we define two animations for this button: one that shows the active image and one showing the inactive image. You can just copy paste those lines, replacing “PlayButton” with the name you like and the paths inside “PlayButtonActive1” and “PlayButtonInactive1”, so that they will point to the correct images. Repeat this step for all the buttons you have.

```
; --- Common to all buttons

; Buttons are part of the UI group, we're going to use that group to filter
objects when picking them under the mouse's position
[Button]
Group         = UI

; Animation set common to all buttons, composed of two states: 'Inactive'
and 'Active'
[ButtonAnimSet]
KeyDuration   = 1
StartAnim     = Inactive ; The default anim is 'Inactive'
Inactive      = # 0 ; We're not renaming the animation but we're going to
use as many frames as are defined in config with the 0 as second item
Active        = # 0 ; Same as above
Inactive->    = Inactive # .Active ; Inactive will loop onto itself and will
get interrupted right away when transitioning to 'Active'
Active->      = Active # +.Inactive ; 'Active' can loop onto itself but will
transition immediately back to 'Inactive' when no target is set

; --- Play button specifics

; The PlayButton itself
[PlayButton@Button]
```

```
Graphic       = PlayButtonInactive1
AnimationSet  = PlayButtonAnimSet
Position      = (0,0,0)

; Set the prefix for the PlayButton animset while retaining the animation
names and transitions from the common animset
[PlayButtonAnimSet@ButtonAnimSet]
Prefix        = PlayButton

; First frame of the 'Inactive' anim using the 'PlayButton' prefix
[PlayButtonInactive1]
Texture       = ui/btn_badge_play.png

; First frame of the 'Active' anim using the 'PlayButton' prefix
[PlayButtonActive1]
Texture       = ui/btn_badge_play_hover.png
```

The code

Now that you have all the buttons configured, we can go to the coding part: We want to set the target animation to Active when the mouse hovers the button and to Inactive when the mouse leaves the button.

So you probably have some update function that handles all your game logic stuff. You may either use this one, are register a new one, that is called less frequently(because the hovering effect of the mouse is not really important/time critical). Nevertheless this is how the update function would look like:

```
//object that stores the currently highlighted button:
orxOBJECT* highlighted_button = orxNULL;

void orxFASTCALL Update(const orxCLOCK_INFO *_pstClockInfo, void
*_pstContext)
{
  orxOBJECT *object = orxNULL;

  // Let's fetch the mouse's position
  orxVECTOR vPos;
  if(orxRender_GetWorldPosition(orxMouse_GetPosition(&vPos), orxNULL, &vPos)
!= orxNULL)
  {
    // Let's see what's currently under the mouse
    object = orxObject_Pick(&vPos, orxString_GetID("UI"));
  }

  // Not hovering the same button as before?
  if(object != highlighted_button)
  {
    // Was hovering a button before?
    if(highlighted_button != orxNULL)
```

```
    {
      // Go back to inactive state by removing the current target anim
      orxObject_SetTargetAnim(highlighted_button, orxNULL);
    }

    // Are we currently hovering a button?
    if(object != orxNULL)
    {
      // Go to active anim (the anim's name is `Active`, the prefix we
defined in config if only used to find the data but doesn't modify the name
itself, this makes the animation set reusable between buttons)
      orxObject_SetTargetAnim(object, "Active");
    }

    // Keep track of what we're hovering
    highlighted_button = object;
  }

  // Check if the user clicked on a button
  if(object && (orxInput_HasBeenActivated("Select")))
  {
    // He clicked...so let's start the game if it was the PlayButton
    if(orxString_Compare(orxObject_GetName(object),"PlayButton") == 0)
    {
      //INSERT SOME CODE THAT STARTS YOUR GAME
    }
    else if(orxString_Compare(orxObject_GetName(object),"OtherButton") == 0)
    {
      //DO OTHER STUFF FOR OTHER BUTTONS
    }
  }
}
```

It assumes, that you have multiple buttons and you want the possibility to add new buttons easily. Once you added the configuration lines for a new button simple add a new check (orxString_Compare(orxObject_GetName(object),“NewButton”)==0) at the bottom of the update function(where the animation is set to AnimationActive).

Hope that helps!!

Notes

There are ways to organize your data in order to reduce the number of lines of config necessary to add new buttons. For example, if you were to follow a precise naming convention for your inactive/active textures, you could reduce config part to:

```
; --- Common to all buttons

; Buttons are part of the UI group, we're going to use that group to filter
objects when picking them under the mouse's position
```

```
[Button]
Group         = UI

; Animation set common to all buttons, composed of two states: 'Inactive'
and 'Active'
[ButtonAnimSet]
KeyDuration   = 1
StartAnim     = Inactive ; The default anim is 'Inactive'
Inactive      = # png ; We're not renaming the animation but we're going to
use look for PNG files for every frame of this animation
Active        = # png ; Same as above
Inactive->    = Inactive # .Active ; Inactive will loop onto itself and will
get interrupted right away when transitioning to 'Active'
Active->      = Active # +.Inactive ; 'Active' can loop onto itself but will
transition immediately back to 'Inactive' when no target is set

; --- Play button specifics

; The PlayButton itself
[PlayButton@Button]
Graphic       = PlayButtonInactive1
AnimationSet  = PlayButtonAnimSet
Position      = (0,0,0)

; Set the prefix for the PlayButton animset while retaining the animation
names and transitions from the common animset
[PlayButtonAnimSet@ButtonAnimSet]
Prefix        = PlayButton

; No need to specify config entries for the animations frames as we're now
using precisely named files for each frame of each animation
; For example, the first frame for 'Inactive' will be stored in a file named
'PlayButtonInactive1.png', etc...
```

Now by following the naming convention (prefix+animation_name+frame_index+.png), we can add new buttons by simply redefining the Prefix property.

Another way to reduce this is by having all the frames inside a single texture and keeping the position of each frame identical for each button. By doing so, we'd simply need to provide the texture at the AnimationSet level and not use the Prefix property.