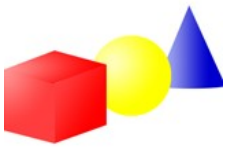
The OWLNext Journal



Number: 2 **Jul, 2009**

Hello OWLNext users!

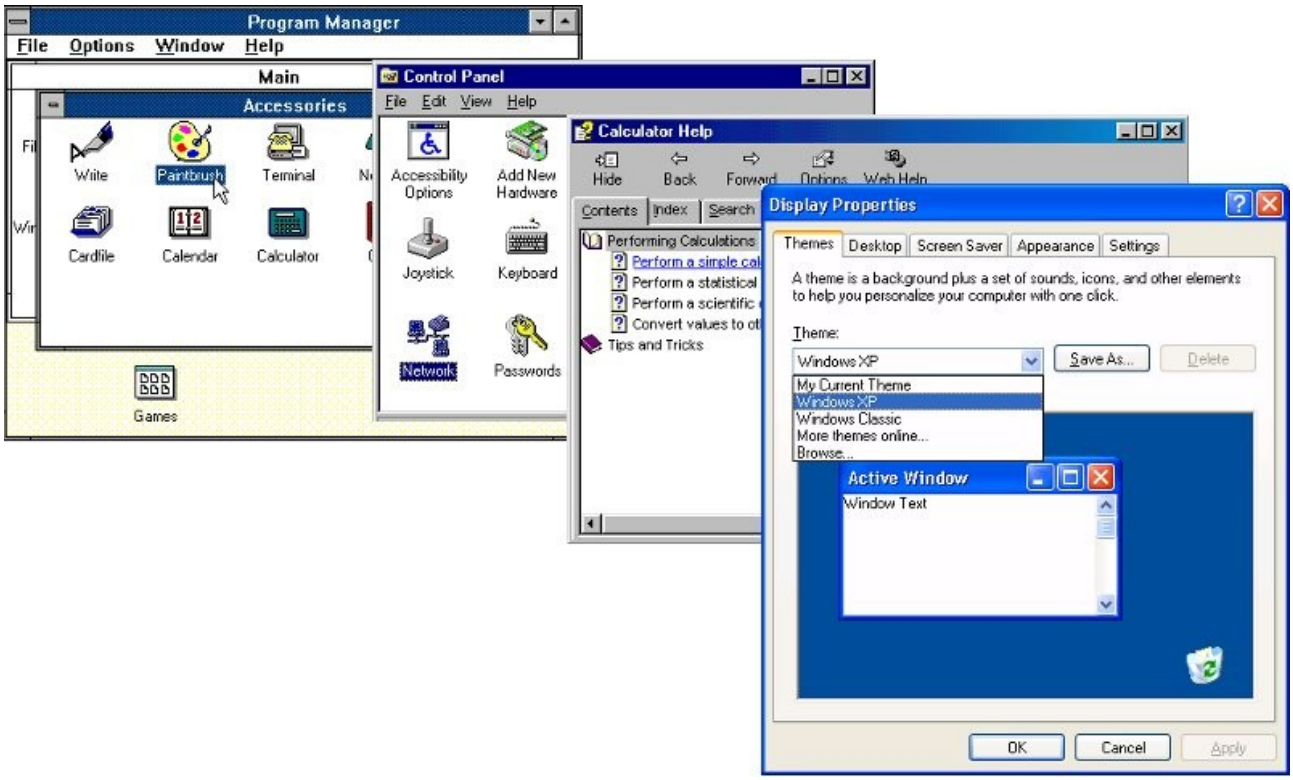
This second edition OWLNext journal we will keep showing cool techniques feasible s with this great framework. As we approach to the twenty anniversary of OWL framework, it demonstrates how a well designed, object oriented, standard compatible and easy to learn framework is a good investment. OWL not only adapted to the changing technologies, and keeps growing towards new horizons (Win64, Ubuntu) it also survived the parent company Borland!

So, lest take a good moment learning, growing and enjoing.

Sincerely yours
Sebastian Ledesma

Thinking outside the box: OWLPlayer.

As we studied in the previous edition, the de facto definition of window is a square area of the screen.



We all recognize that the rectangle is probably the most economic way of utilize the screen resource, but wait... why should our applications do follow the strict economic criteria while it's possible to optimize also considering the good looking. Remember: there is no a second_first good impression.

Actually if we analyze the GDI API, most of the functions expect a TRect parameter. In other hand our design should be smart enough to be easily reusable in other applications or even within the same application.

So the basic purpose of this edition is to develop a way to create a non-rectangular Window.

OWLNext allows this with easy , since it provides a real coverage of Windows API (eat that Win32++), while supports multiple compilers (and real soon multiple platforms).

To reduce the work, I've started this demo with a previous example

wmPlayer. So it allowed me to concentrate in the particular work.

I wanted to show a bitmap in the window, but also wanted that the silhouette of my window was according to my bitmap. Also I wanted that I can change the bitmap without reprograming. How to do it? Simple:

I just created a 'loadSkin' function. It loads a bitmap file into a Tdib. It allows to define the directory where the bitmap file is (in my example it's just in the same directory of the application).
Also added a foldback mechanism: if the bitmap is not available, just use one from internal resource (to economize I used the very same).

Then explore the bitmap, pixel by pixel, if the color is the predefined as 'maskColor' create a 1 pixel region.

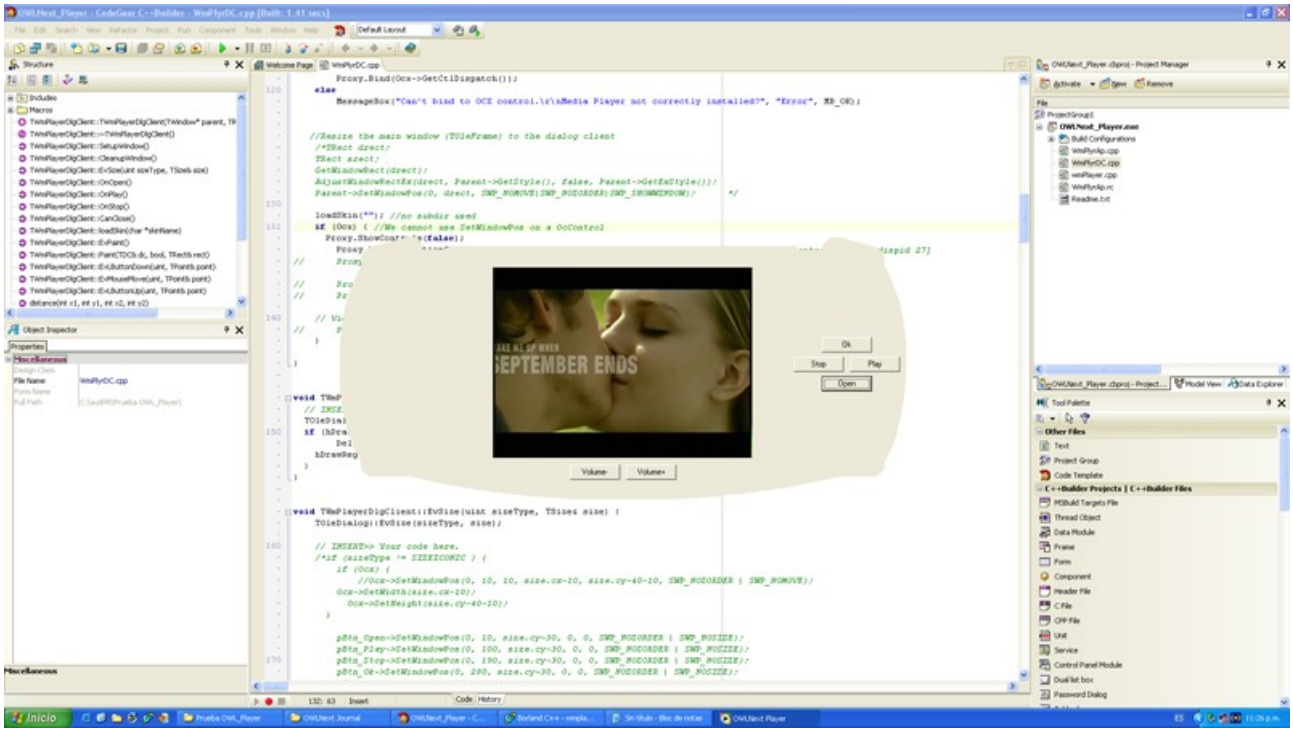
```
hRegion=CreateRectRgn( x, y, x+1, y+1);
```

Combine all the regions into a single region:

```
CombineRgn(hRegionComplete, hRegionComplete, hRegion, RGN_XOR);
```

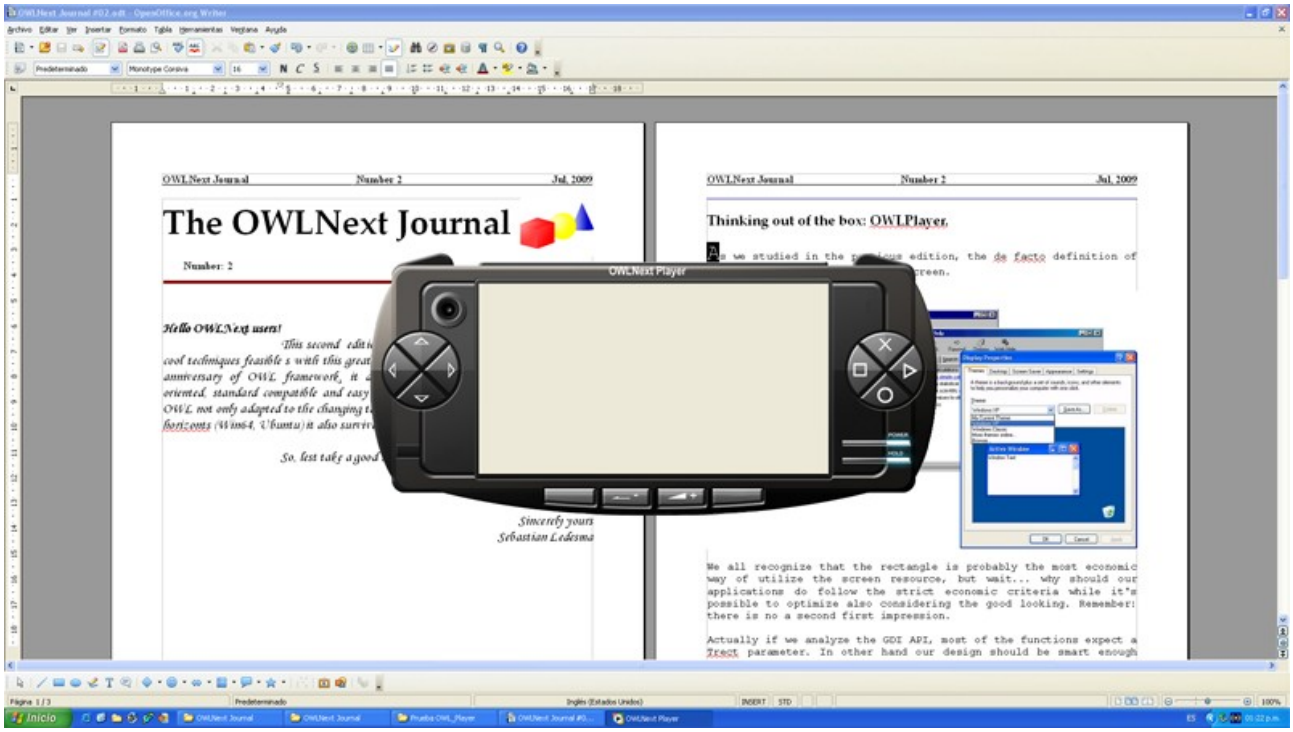
and then apply the region to the main window:

```
SetWindowRgn(GetApplication()->GetMainWindow()->GetHandle(),
hRegionComplete, true);
```



The next part is just define a implement the **EvPaint** (the default

handler for WM_PAINT message). Using **SetDIBitsToDevice** we paint our bitmap into the window. Since the area which has the color of the 'maskColor' is outside the defined Region we have the wanted effect.



To avoid use standard buttons, I've created 'virtual buttons', which are simple x/y coordinates with a radio size.

Notice how I check for them and notify the dialog using the standard OWL mechanism:

```
int cmd=checkVirtualButton(point);
if(cmd)
    PostMessage(WM_COMMAND,cmd,0);
```

All this parameters are readed using a TProfile, so they can be changed for every particular skin (bitmap).

It's all! You can download the demo and source code trough any of the affiliated OWLNext sites.

This example is another demonstration of the Power of OWLNext. We can do much cool and interesting things.

You´ll see. I´ll show you.■