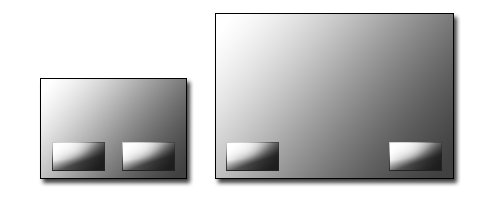
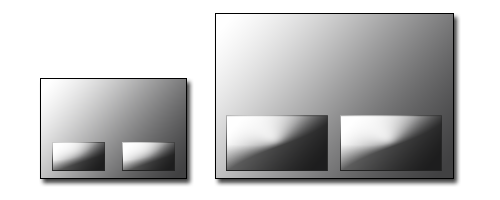
**User Interfaces (UI)**  
  
There was a time when 80 characters and 20 lines were the peak of perfection, people were doing this, so far, you can be surprised. And now? And what now? Now diversity, diversity is a choice, and the choice of this problem. But do not let the bad news, let's talk about the good. In this article (forgive me, journalists), we'll talk about the types of interfaces, in any case will not affect the usability aspects, because it is a different story. Just drop the exotic, and that we have left? And is not much...  
  
  
**The main types of building interfaces**  
  
Each time thinking about the interface, people think pictures, here such here this, but here 's this beautiful. And when it comes to implementation, many things are unacceptable, other unattainable third road either to develop or to maintain. Therefore, each artist and designer, in whose department includes the development of the interface should present how they are implemented, at least in general terms. I have to say, there are no bad type of UI. Each type is better suited for particular types of problems, but mixing them is not worth it, this complicates the support, tools and development well. And each type has its pluses and minuses. There are roughly two types of UI:  
  
1. Absolute.  
2. Relative.  
  
Briefly describe these types, and then talk more about each, compare them, and consider methods of dealing with minuses.  
  
  
  
**A brief overview of types of UI**  
The main emphasis in absolute style, done on a constant window size in pixels. If the window has a size of 200 by 200 pixels, then any monitor it will have the same size. It immediately follows that on monitors with higher resolution, the window will appear smaller ( ignore the diagonal of the monitor ). And on monitors with a lower resolution, the window will occupy more space. And as a consequence of such interfaces, basically, have to make more flexible, such as increasing the workspace, move, or grouping toolbars. Below is a schematic version of the UI layout on different monitors.



In all of the relative type of UI elements that have a size that is expressed in relation to the dimensions of the parents. Accordingly, the main window, expressed relative to the size of the monitor. It follows that on different monitors, the location UI will look about the same. But there are artifacts, because reflection permission UI on the monitor resolution is not the same. Below is a schematic version of the UI layout on different monitors.



In this brief review can be completed, proceed to the detailed description of the construction elements UI. Let's start with the absolute.  
  
  
  
**Absolute UI**  
  
We assume no rush artist paralyzed, and from his pen was born masterpiece. So nice to meet you, I button.



Well, a good button, we will gladly insert it in our form and admire. Everything is going fine until one day, we need a button twice longer experimenting with stretching, we understand that this option is not for us because we have the following:



Hmm... this option certainly interesting, but I think it's better that the artist did it for us, maintaining the proportions texture, we think.



Quite another thing, again we are happy and satisfied with their work all...  
  
  
Now imagine that these elements do not even 5 or even 10, and sizes not two options, and designing forms autsortitsya. Terrible picture, but there are two ways, either we strictly limit the designers set of elements and their sizes, either initially create such an option, which includes stretching. In this article, we try to create items that may stretch without spoiling decorative pieces (mostly border element), ie interface element should look the same in different sizes. So now the artist put the task to draw an element, so it can be divided into parts, and to identify those that will be exposed to stretching. Depending on the direction of stretching, or obtain three pieces 9. For simplicity, we take only the horizontal stretching, respectively will suffice to 3 parts, so the original button.



Divide into 3 parts:



Now it is possible to stretch the middle part and receive any key length.



These simple steps may get stretched window titles, buttons, edit fields, and virtually any elements of UI. One of the advantages of such interfaces, it is predictable outcome. We can be sure that any monitor and at different sizes and proportions, the item will look like its painted artist, pixel by pixel. A major drawback is the inhomogeneous space. Depending on the resolution, different users will vary the amount of space, and thus an urgent need to build UI. To mitigate this shortcoming, a versatile layout system and freedom UI. For example, you can allow the user to move the items to add or remove them from the panel. Assemble the panels themselves. Move windows, resize them, and similar operations. Just face the problem of scaling the entire UI, if it is not laid at once, perhaps when the size of individual elements, it would be wrong to look layout forms.  
  
The simplest example of this is a Windows UI and similar graphical operating systems. Scope interface - this application, mainly working in windowed mode, where you can change the size and proportions of the windows in an arbitrary manner. Up to portrait mode.  
  
  
  
**Relative UI**  
  
In this type of UI elements all have a relative size which is set in a range from 0 to 1. And you can guarantee that if the window takes up half the screen, on any display, this window will occupy half of the screen. Elements not usually made ​​from pieces, and each cell consists of a single piece of texture. Rapid development of one element is high, but for each new window, you must re- arrange the form or collect texture artists. If you change the layout, this again means a complete mess, and for texture windows involving artist. For example, if the Add button, and you need to change the width, in most cases, the texture of the parent window, you need to redo a new dimension, a simple stretching distorts the decorative elements such as a header or frame. Windows rarely change their size, and if acceptable arrangement of elements, it predetermined or available outside of the application. Thus, in most cases, is not supported portrait mode display and for non-standard ( not 3:4 ) ratios may be observed distortion. But in most cases supported widescreen option where width is ignored, and the windows are arranged in proportion to height. Scope of UI, this full-screen games, both on the PC and consoles. And also interfaces complex shapes that can not be stretched, for example in the form of a flower, and an animal, etc.. If permissible switching to windowed mode, it is usually forbidden to change the proportions of the windows, or do its dimensions. As well there is a problem of image quality textures. Texture size must be taken with reserve and avoid thin contrasting lines, and their sequencing. Most irritable, this font, unless special equipment, it is almost always the font will be blurred, so make it bigger, and almost always aligned to pixel boundaries. Here's a simple example of the consequences of the miss pixels textures per pixel monitor.



**Comparing the two types of UI**

|  |  |  |
| --- | --- | --- |
|  | absolute UI | relative UI |
| texture Quality | **+**  (elements look like an artist painted them) | **-**  (possible distortion depending on the resolution) |
| Rapid development of one element | **-**  (for each element created skin) | **+**  (painted by the artist element is used almost immediately) |
| Speed ​​the replacement of one type of item to another | **+**  (one skin is replaced by another) | **-**  (in each window element needs to be replaced) |
| The rate of modification of all the elements of the same type | **+**  (settings are changed in the skin) | **-**  (in each window a modification to the element) |
| Speed ​​creating forms | **+**  (using the forms editor) | **-**  (artist draws again form) |
| Rate of modification of forms | **+**  (changes in the editor) | **-**  (can complete overhauls from scratch) |
| Scaling | **-**  (necessary to create the theme manager) | **+**  (originally created for different scales) |

**Conclusion**  
  
On this, perhaps, we're done, lastly say that there is no good and bad types of UI, there are more preferred, depending on the conditions, and each has both pluses and minuses. And of course there are always exceptions. If the project type of solitaire, it makes no sense to use complex system UI. If the project is constantly changing, the interface changes can turn into a nightmare, some blaming it on the shoulders of users, but that's another story.

my.name lead MyGUI