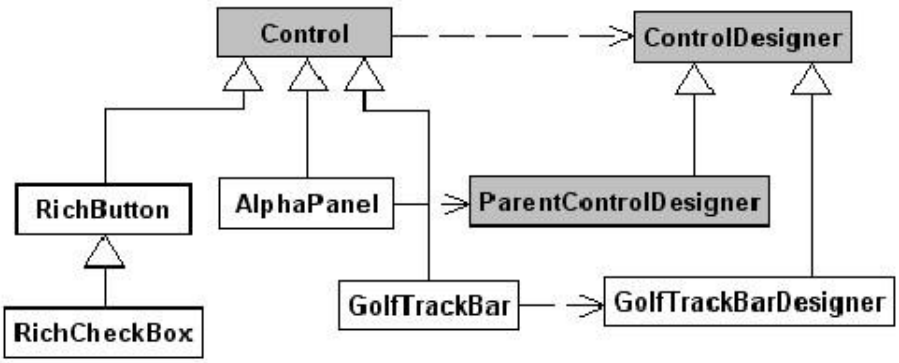
Annex N - SwingImproverControls

Due to the nature of the project, a detail that was no less, was the user interface.
Since the same, regardless of its character as a prototype, would be tested by a broad audience
it was necessary to have not only a simple interface is needed but also the
it was friendly and nice. That's why we turn to some extent controls
custom although not facilitate the use of already provided by the framework. NET,
greatly improved appearance.

In a first degree TrackBar was replaced (to the position of the video) by a personalized
as the framework provided by not meeting expectations or equipment or the customer,
even in the early stage of the project. Later in the same and already view
circulation, is attended most colorful panels and buttons. The buttons did not meet some of
function of a single button, and other function checkbox.



All controls implemented by the team must inherit from the Control class
framework, since it is one that can work with them like controls
also take on many normal and basic functionalities which are necessary for the use of
thereof. Another option is to use Custom Controls that provides the framework and facilitates its
use of a designer through the Visual Studio, but due to previous experiences the
Team members decided to use the first option.

The advantages thereof are:

- More flexibility.
- Stability.
- Greater control over their behavior.
- Fewer constraints.

However it has disadvantages:

- Programming without design tool.
- Less inherited features.
- Increased complexity of programming.

Note that since all of these controls were designed for this project particular, many values were used as constants to facilitate rapid change and global impact. However, it also took into account the possibility that the same are reused tomorrow and this feature was maintained at the lowest possible level for your change is not as burdensome.

GolfTrackbar

Key Features:

- Transparency
- Marker image
- Custom Designer.

Transparency

The way to solve the transparency for this control was quite simple. He enabled a transparent background using the method inherited from the Control class:

SetStyle (ControlStyles.SupportsTransparentBackColor, true)

Where control is used, it is put as the background color transparent, so does the desired effect.

Marker Image

The marker image is achieved through a tiff because it is the type of image that allows the use an alpha channel (see Annex M) and achieves the desired blending in contrast with the background.

Custom Designer

Due to the size of a trackbar can be modified only to the sides, is limited the changeover of the designer creating a design class that inherits from itself ControlDesigner. Overwriting SelectionRules property, it achieves the desired effect. This class is assigned to objects GolfTrackBar the following class attribute:

```
[Designer (typeof (GolfTrackBar.GolfTrackBarDesigner))]
public class GolfTrackBar: Control {
```

AlphaPanel

Key Features:

- Semi-transparent background.
- Curved edges and hazy
- Container
- Overlap with other AlphaPanels

Semitransparent Fund

While the first impression is that this is a minor issue, note that it was the most complexity of this control, and for which research was needed most.

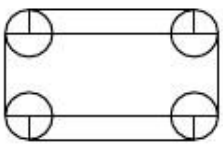
The first thing to note is that the transparency of this control should be independent background color. That is, if it is set to white background color, transparency is being variable. Why clarifies this point? For a possible solution would have been assign a color with alpha component to establish transparency. However this not only creates a direct relationship between color and transparency, but does not solve the the whole problem because the controls. NET background only support full transparency or solid colors.

Through the open source tool "Reflector" was investigated as do the controls framework to solve at least full transparency. Basically what it does is invoke the method of control father painted (or container) by being painted on the object *Graphics* control before painting itself. How was that resolved identical, but using a semi-transparent color with alpha component for paint above.

Curved edges and hazy

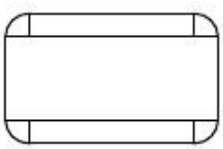
Instead of painting the full panel for a nicer interface, is needed curved corners.

First drawn by parties to achieve the effect, painted the following regions:



Unfortunately, while using solid colors, the image acquired the desired shape, the
use semi-transparent colors, the effect achieved was as follows:

Considering this difficulty, we proceeded to define better the way they would be painted
going to the following regions:



With these regions was finally achieved that the figure had the shape and the coating desired.

Finally had to blend the edges so they do not notice the pixel at the corners of
panel. This was achieved by the property *SmoothingMode* Object *Graphics:*

```
pevent.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
```

Unfortunately, while this property solves the problem of hazy edges,
joins again the problem of the different regions used to paint the panel. This
occurs because the blending is created around each of the figures are drawn and generates
a border around the regions.

Finally, we analyzed a solution to this problem and created a Path that describes
the contour of the panel and painted by the method *FillPath* of *Graphics* which vanishes only
contour.

Container

Since, as its name implies, are AlphaPanel panels, needed the same
behave as a container. Fortunately. NET solves this problem by
control following style:

```
SetStyle (ControlStyles.ContainerControl, true);
```

However, this was not enough to work with these controls at designer
as if they were containers. As for the GolfTrackBar is assigned an
particular designer, but in this case you can use one provided by the framework, the
ParentControlDesigner.

Overlap with other AlphaPanels

As AlphaPanel have a degree as a flap, is used on top of each other for get a behavior similar to TabControl. However the mechanism transparency used, control control draw on all his father rectangle, which produced the following effect:

For this reason the region is marked by the Property Control inherited Region Path giving it the same mentioned above. Unfortunately this eliminated the gone on the right and bottom. So we added one pixel to the region in these sides. Doing this is achieved the desired effect.

RichButton

Key Features:

- Rounded
- Gradient
- Light
- Emboss
- Image
- Text
- Behavior
- Flickering

Rounded Edges

Rounded edges for the same solution was used for the AlphaPanel where what became the father was drawing before starting to draw the control itself. Then the Rounding is achieved by a *path*.

Gradient

In addition to the background color is inherited from the class *Control* the framework, this control GradientColor has a property. The button will color a vertical gradient between these two colors and is achieved through the LinearGradientBrush class, which provides the framework.

Light

For the light seen in the upper right corner used a tiff that is set by a static property. We used a tiff image such as allows us an alpha channel (See Annex M). Anyway, through a property can choose to not display no light.

Emboss

This was perhaps the most visual impact caused him to team work. Although an effect of 3 dimensions can be achieved easily using preset colors lines as Windows does not achieve freedom and softness that was desired. This is why it is using the color values you already have, obscuring the bottom and right, and turn clarifying the top and left edge. In turn, the closer the edge, the greater the index which darkens and lightens. Moreover the maximum index is defined by a This property causes the button to appear more or less three-dimensional. All this is done by a code block *unsafe* to achieve high performance.

Regardless, the corners were not as simple as the rest because we have several solutions that achieve different effects, and 3 different types of corners:

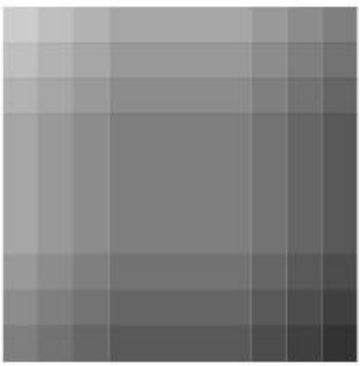
Clear - Clear

Clear - Dark

Dark - Dark

The solution adopted involved adding the index affects the row and column as
The following diagram shows:

0,6	0,5	0,4		0.3	0.3	0.2	0.1	0
0,5	0,4	0,3		0.2	0.2	0.1	0	-0.1
0,4	0,3	0,2		0.1	0.1	0	-0.1	-0.2
0,3	0,2	0,1		0	0	-0.1	-0.2	-0.3
0,3	0,2	0,1		0	0	-0.1	-0.2	-0.3
0,2	0,1	0		-0.1	-0.1	-0.2	-0.3	-0.4
0,1	0		-0.1	-0.2	-0.2	-0.3	-0.4	-0.5
0		-0.2		-0.3	-0.3	-0.4	-0.5	-0.6
	-0.1							



Image

This button allows the allocation of an image displayed on the front of the button. For these images also used the tiff format to achieve an effect of *antialias* at the edges of the image.

Text

The only special features of the text of these buttons is that they are with *antialias* and having an opacity variable.

Behavior

For the basic behavior of this button has already imitated the buttons that come with the framework. Without But to achieve the effect that a button is pressed, what was done was to modify the light to simulate a shadow (invert colors), and also reverses the emboss.

Flickering

Unfortunately, this type of image-consuming process to achieve the 3-dimensional effect. Is trabajase this that if on the drawing is usually obtained a flickering to be inconvenient in view of user. That is why an amendment to a backbuffer is generated with the image you want displayed and the backbuffer is the one drawn.