Surfaces MyrScript

A- Prérequis

1- Dans tous nos exemples, les variables suivantes sont préinitialisées:

- gr : le contexte graphique courant (Graph) permettant de visualiser le résultat, qui a été rempli avec une couleur violet foncé

- cats : Une surface dans laquelle a été importée l'image "cat.jpg" par `cats=Surface.New("cat.jpg")`
L'image "cat" fait 280 pixels par 186 :

- myrs : Une surface dans laquelle a été importée l'image "myriad.png" par `myrs=Surface.New("myriad.png")`
L'image "myriad" fait 47 pixels par 45 et possède un fond transparent:

2- L'exemple travaille sur une surface appelée "s"

3- Le résultat de l'exemple est affiché par `gr.DrawSurface(0,0,s.Width,s.Height)`

4- Les surfaces sont libérées par
```
cats.Dispose()
myrs.Dispose()
s.Dispose()
```

B- Les méthodes de surface

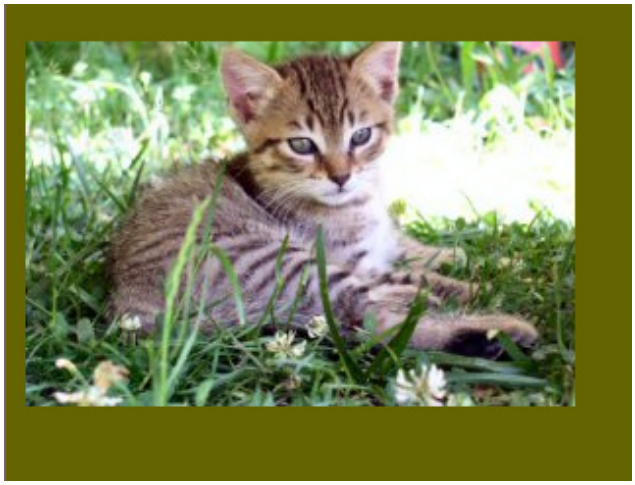
- GetPixel/SetPixel
Lecture et écriture de pixels rouge/vert/bleu/opacité

```
s=Surface.New(cats.Width,cats.Height) -- Create empty surface
for y=0,s.Height-1 do
for x=0,s.Width-1 do
r,g,b,op=cats.GetPixel(x,y)
s.SetPixel(x+20*sin(y*5),y,r,g,b,op) -- Wave
```

```
end
end
```

- AddMargins
Ajoute des marges autour de l'image

```
s=Surface.New(cats) -- Create by copying
s.AddMargins(10,20,30,40, 100,100,0, 100) -- left,top,right,bottom, r,g,b, opacity
```



- Crop
Recadre l'image

```
s=Surface.New(cats) -- Create by copying
s.Crop(100,10,190,100) -- left,top,right,bottom
```

- Scale
Applique à la taille x et y les valeurs de zoom

```
s=Surface.New(cats) -- Create by copying
s.Scale(60,90) -- in %
```

- Resize
Etire l'image pour lui faire atteindre les dimensions largeur/hauteur fournies

```
s=Surface.New(cats) -- Create by copying
s.Resize(168,168)
```

- Rotate
Fait pivoter l'image. Les marges qui apparaissent peuvent être colorees

```
s=Surface.New(cats) -- Create by copying
s.Rotate(20, 0,0,0, 0) -- in degrees, transparent margins
```

- Perspective
Met l'image en perspective

```
s=Surface.New(cats) -- Create by copying
```

```
s.Perspective(30) -- in degrees
```

- Highlight
Augmente ou diminue la luminosité

```
s=Surface.New(cats) -- Create by copying
s.Highlight(150) -- in percent
```

- Glow
Augmente la luminosité de manière non-linéaire

```
s=Surface.New(cats) -- Create by copying
s.Glow(250) -- in percent
```

- AdjustBCS
Règle la luminosité, le contraste et la saturation

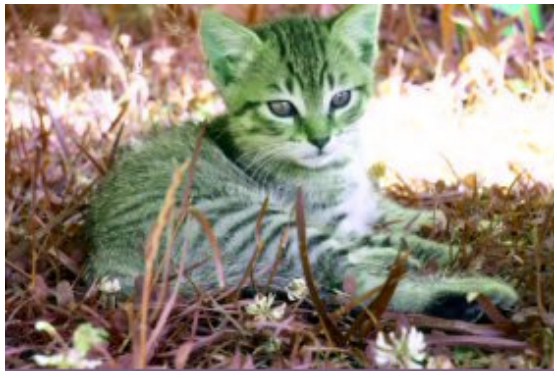
```
s=Surface.New(cats) -- Create by copying
s.AdjustBCS(-10,40,30) -- in percent
```

- AdjustHSL
Règle la teinte, Saturation et luminosité

```
s=Surface.New(cats) -- Create by copying
s.AdjustHSL(20,30,-10) -- in percent
```

- ChannelMixer
Chaque composante (R,V,B) recoit une base + une combinaison des composantes d'origine

```
s=Surface.New(cats) -- Create by copying
s.ChannelMixer(0, 0,100,0, 0, 100,0,0, 0, 0,0,100) -- invert red & green
```



- Blur
Flou

```
s=Surface.New(myrs) -- Create by copying
s.Blur(6)
```

- DropShadow
Ombre portee : delta x et y, couleur RVB, niveau de flou et opacité

```
s=Surface.New(myrs) -- Create by copying
s.DropShadow(7,7, 0,0,0, 6, 50)
```

- DropInnerShadow
Ombre portee a l'intérieur (reflet): delta x et y, couleur RVB, niveau de flou et opacité

```
s=Surface.New(myrs) -- Create by copying
s.DropInnerShadow(7,7, 255,255,255, 3, 80)
```

- Rectangle
Remplit une zone rectangulaire avec couleur et opacité

```
s=Surface.New(myrs) -- Create by copying
s.Rectangle(20,20,60,60, 0,255,0, 80, true)
```

- Embed
Inclut une surface dans une autre

```
s=Surface.New(cats) -- Create by copying
s.Embed(myrs, 190,110, false)
```

- OpacitySet
Force l'opacité

```
s=Surface.New(cats) -- Create by copying
s.OpacitySet(40) -- in percent
```

- OpacityFade
Modifie l'opacité

```
s=Surface.New(cats) -- Create by copying
s2=Surface.New(myrs)
s2.OpacityFade(50) -- in percent
s.Embed(s2,190,110, false)
s2.Dispose()
```

- OpacityBlur
Rend floue l'opacité

```
s=Surface.New(myrs)
s.OpacityBlur(10)
```

- OpacityInvert
Inverse l'opacité

```
s=Surface.New(myrs)
s.OpacityBlur(10)
s.OpacityInvert()
```

- OpacityCompute
Calculs entre les opacités de deux surfaces (centrées)

```
s=Surface.New(cats)
s.OpacityCompute(myrs,0) --0 copy, 1 overlay, 2 subtract, 3 multiply
```

- OpacityFromColor
L'opacité est calculée en fonction d'une couleur

```
s=Surface.New(cats)
s.Highlight(150)
s.OpacityFromColor(255,255,255) --white becomes transparent
```

- OpacityFromLuminance
L'opacité est calculée en fonction de la luminance

```
s=Surface.New(cats) -- Create by copying
s2=Surface.New(myrs)
s2.OpacityFromLuminance()
s.Embed(s2,190,110, false)
s2.Dispose()
```