Scripting Inkscape

Automatizziamo le operazioni piu' frequenti tramite semplici script...

Inkscape e' probabilmente il miglior software libero per la grafica vettoriale. E' multipiattaforma ed e' oggi giunto alla versione 0.48. Per chi utilizza Inkscape con assiduita', la possibilita' di scriptare per velocizzare le operazioni compiute piu' frequentemente usate diviene fondamentale.
Una delle cose che utilizzo piu' spesso nei miei disegni e' una sorta di spilla nella quale inserire i loghi (vedere l'immagine). Creare una semplice spilla consiste principalmente di tre passi:
1. inserire un cerchio pieno grigio scuro con un po' di sfocatura (per l'ombra);
2. inserire un cerchio pieno del colore di cui vogliamo la spilla;
3. inserire un cerchio con riempimento gradiente bianco alpha 0 – bianco alpha 255 (per il riflesso).
Vediamo come automatizzare questa procedura con un piccolo script in python.

Lo script...
Prima di tutto dovremo importare la classe inkex (Inkscape Extension), poi potremo iniziare a scrivere la nostra classe che chiameremo BadgeEffect che erediti dalla classe Effect contenuta in inkex.

```
#!/usr/bin/env python

import sys
sys.path.append('/usr/share/inkscape/extensions')

import inkex, simplestyle
from simplestyle import *
```

Nella classe che creeremo dovremo inserire un costruttore (__init__) che si occupi di inizializzare la classe e istanziare i parametri che vengono passati. Consideriamo come parametro soltanto il colore che vogliamo per la spilla.

```
class BadgeEffect(inkex.Effect):
    """
    Create a badge!
    """
    def __init__(self):
        """
        Constructor.
        Defines the "--color" option of the script.
        """
        inkex.Effect.__init__(self)

        self.OptionParser.add_option(
           '-c', '--color', action = 'store',
           type = 'string', dest = 'color',
           default = '#008080',
           help = 'What colour do you prefer?')
```

In questo pezzo di codice abbiamo inziato a definire la classe e abbiamo definito il costruttore. I due parametri vengono definiti tramite OptionParser.add_option che accetta numerosi parametri:

- '-c' e '--color' si utilizzano per definire il modo in cui passare i parametri di questo plugin nell'uso da riga di comando;

- action indica l'azione da compiere su quel valore (store indica di salvare il valore);

- type indica, come e' facile intuire, il tipo di valore;

- dest e' la destinazione dove verra' salvato il valore (che nel nostro caso troveremo dunque in self.options.color.attribute);

- default, anche qui e' facile intuirlo, contiene il valore predefinito del parametro;

- help e' la stringa di aiuto che verra' mostrata nel caso di errori con i parametri passati.

Arriviamo alla parte interessante, cioe' a cosa il plugin deve effettivamente fare! Definiamo dunque la funzione effect.

```
    def effect(self):
        """
        Creates a badge at the center of the page.
        """
        # Get script's "--color" option value.
        color = self.options.color

        # Set the badge radius to 50 and blurRadius
to 5
        radius = 50
        blurRadius = 5

        # Get document's properties
        svg = self.document.getroot()
        width  = inkex.unittouu(svg.get('width'))
        height = inkex.unittouu(svg.attrib['height'])
        defs =
self.document.getroot().xpath('//svg:defs',
namespaces=inkex.NSS)
        defs = defs[0]

        # Create a new layer
        layer = inkex.etree.SubElement(svg, 'g')
        layer.set(inkex.addNS('label', 'inkscape'),
'Badge')
        layer.set(inkex.addNS('groupmode',
'inkscape'), 'layer')
```

Abbiamo iniziato assegnando alla variabile *color* il parametro, inizializzando le variabili radius e blurRadius (che definiscono il raggio della spilla e il raggio di sfocatura) e prelevando il documento, la sua larghezza, la sua altezza e il ramo defs. Abbiamo inoltre creato ora un nuovo livello nel quale inserire la spilla. In realta' abbiamo creato un nuovo elemento SVG e l'abbiamo segnalato a Inkscape come livello con le due righe successive. Ora, prima di disegnare i vari cerchi dobbiamo creare il filtro che useremo per applicare la

sfocatura gaussiana (la sfocatura che normalmente impostiamo da *Riempimento e Contorni*).

```
        filt = inkex.etree.SubElement(defs,
inkex.addNS('filter', 'svg'))
        filtId = self.uniqueId('filter')
        self.filtId = 'filter:url(#%s);' % filtId
       for k, v in [('id', filtId), ('height',
str(10)), ('width', str(12)),('x', '-0.5'), ('y', '-
0.5')]:
                filt.set(k, v)
        fe = inkex.etree.SubElement(filt,
inkex.addNS('feGaussianBlur','svg'))
        fe.set('stdDeviation', str(blurRadius))
```

Ora disegniamo il primo cerchio, quello nero che useremo come ombra. Possiamo vedere che con la variabile che chiamiamo *shadow_style* impostiamo il contorno (stroke), il riempimento (fill), e il filtro da usare (quello dichiarato in precedenza). *circ_attribs* lo usiamo per definire la posizione e il raggio del cerchio e ad esso abbiamo passato anche il valore che dovra' essere attribuito alla chiave *style* tramite la variabile *shadow_style* precedentemente inizializzata.

```
        shadow_style = { 'stroke': 'none',
               'fill': '#000000',
               'filter' : self.filtId.split(':')[1]}
        circ_attribs = {'style':
        simplestyle.formatStyle(shadow_style),
               inkex.addNS('label','inkscape'):
               'shadowstyle',
               'r':'50',
               'cx':str(width / 2),
               'cy':str(height/2)}
        shadow = inkex.etree.Element(
               inkex.addNS('circle','svg'),
               circ_attribs)
        layer.append(shadow)
```

Procediamo a creare il secondo cerchio con una procedura quasi analoga.

```
        style = { 'stroke': 'none', 'fill': color}
        circ_attribs = {'style':
               simplestyle.formatStyle(style),
               'r':'50',
               'cx':str(width / 2),
               'cy':str(height/2)}
        maincircle = inkex.etree.Element(
               inkex.addNS('circle','svg'),
               circ_attribs)
        layer.append(maincircle)
```

Ora, prima di disegnare il terzo e ultimo cerchio creiamo il gradiente con cui riempirlo

```
        lg = inkex.etree.SubElement(defs,
               inkex.addNS('linearGradient', 'svg'))
        lgId = self.uniqueId('linearGradient')
        self.lgId = 'linearGradient:url(#%s);' % lgId
        fork,vin[('id',lgId),('height',
               str(10)),('width', str(12)),
               ('x', '-0.5'), ('y', '-0.5')]:
        lg.set(k, v)
```

I seguenti sono i due passaggi che compongono il gradiente.

```
        lstop1 = inkex.etree.SubElement(lg,
               inkex.addNS('stop','svg'))
        lstop1.set('offset',str(0))
        lstop1.set('style','stop-color:#ffffff;
               stop-opacity:1;')
        lstop2 = inkex.etree.SubElement(lg,
               inkex.addNS('stop','svg'))
        lstop2.set('offset',str(1))
```

Finalmente creiamo l'ultimo cerchio.

```
        reflex_style = { 'stroke': 'none',
               'fill-opacity':'0.5'fill':
               self.lgId.split(':')[1], 'filter' :
               self.filtId.split(':')[1]}
        circ_attribs = {'style':
               simplestyle.formatStyle(
                      reflex_style),
               'r':str(radius),
               'cx':str(width / 2),
               'cy':str(height/2)}
        reflex =
               inkex.etree.Element(inkex.addNS(
               'circle','svg'), circ_attribs)
        layer.append(reflex)
```

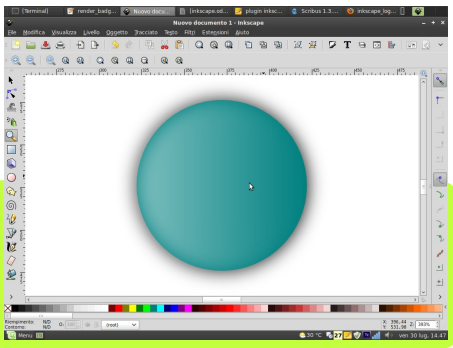
Conclusa la funzione relativa all'effetto si rende necessario istanziare la classe creata e lanciare l'effetto:

```
effect = BadgeEffect()
effect.affect()
```

Il nostro file render_badge.py e' completo. Ora dobbiamo creare un file con estensione .inx che descriva il plugin, l'interfaccia e il posizionamento nei menu. Ecco quindi render_badge.inx:

```
<inkscape-extension>
  <_name>Badge</_name>
  <id>org.ekips.filter.hello_badge</id>
  <dependency type="executable"
location="extensions">render_badge.py</dependency>
  <dependency type="executable"
location="extensions">inkex.py</dependency>
  <param name="color" type="string" _gui-text="What
colour do you prefer for your
badge?">#008080</param>
  <effect>
    <object-type>all</object-type>
    <effects-menu>
       <submenu _name="Render"/>
    </effects-menu>
  </effect>
  <script>
    <command reldir="extensions"
interpreter="python">render_badge.py</command>
  </script>
</inkscape-extension>
```

Copiate entrambe i file nella cartella ~/.inkscape/extensions e lanciate Inkscape. Nel menu Estensioni -> Render troverete Badge. Nell'immagine il risultato della sua applicazione ;)



http://wiki.inkscape.org/wiki/index.php/ScriptingHOWTO
http://wiki.inkscape.org/wiki/index.php/PythonEffectTutorial
http://wiki.inkscape.org/wiki/index.php/Extension_repository
http://wiki.inkscape.org/wiki/index.php/Extension_subsystem
http://wiki.inkscape.org/wiki/index.php/QuickDrawingAPI