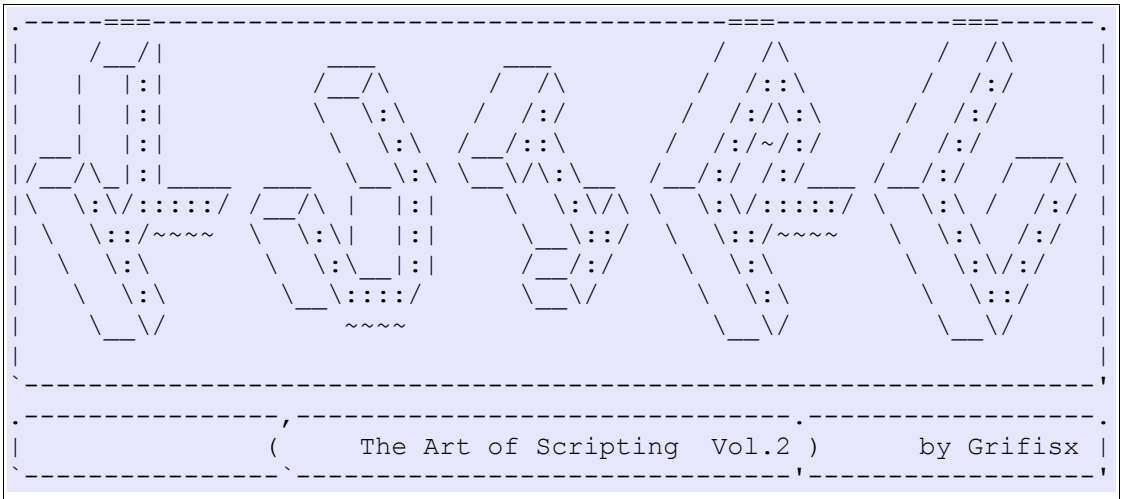

Oggetti, Classi, Array e dizionari

Start:

Aggiornamento 20.07.2010

In questo tutorial ci occuperemo dei concetti di *oggetto* e di *classe*, degli *array*, dei *dizionari* e della *scrittura e lettura dei file*. Anche questa volta cercheremo di dare spazio all'esame di piccoli script, perché,sicuramente,il modo migliore per imparare è buttare giù codice.

Introduciamo il concetto di oggetto.

Per capire cosa sia, senza perderci in giri di parole, vediamo il seguente codice:

```
%myobject = $new(object,0,oggetto);
if(%myobject)echo Oggetto creato correttamente!";
else echo "Creazione dell'oggetto fallita!";
```

Mettetelo nello "script tester" (quello con l'icona della bomba a mano nera) ed eseguitelo per vederne il risultato che apparirà nella finestra della *console*.

Adesso esaminiamo quello che abbiamo scritto:

%myobject = $new(object,0,nome_oggetto);

Per creare un oggetto bisogna usare la funzione *$new()* la quale richiede 3 parametri:

- La classe a cui appartiene l'oggetto e che ne identifica il tipo,la natura,(dopo vedremo cosa si intente per classe)
- L'oggetto padre, (che può essere 0 per gli oggetti top-level (cioè a dire oggetti che non hanno padre, questi concetti verranno spiegati in seguito).
- Il nome di fantasia che vorremo dare all'oggetto (può anche essere vuoto).

Gli oggetti così creati possono essere paragonati agli oggetti reali, e, proprio come gli oggetti reali, sono contenitori di informazioni e di "azioni" per farvi capire meglio vi faccio un esempio pratico:
un libro, nella realtà, ha un titolo, un autore, una casa di editrice,un anno di pubblicazione ed un contenuto.
Creiamo,tramite script, il nostro oggetto libro, imitando la realtà

```
// Creiamo l'oggetto libro
%Libro = $new(object,0,libro)
// Settiamo i campi
%Libro->%titolo = "La via Lattea"
```

```
%Libro->%autore =  "Piergiorgio Odifreddi"
%Libro->%editore = "Longanesi"
%Libro->%anno = "2008"
%Libro->%contenuto =","
```

Mettiamo il codice nello script-tester ed eseguiamo, dopodiché posizioniamoci sulla riga di input, di qualsiasi finestra, e digitiamo:
/echo %Libro->%autore
come vedrete ci ha dato l'echo del campo autore, mentre se facciamo un echo dell'oggetto in se tipo:
/echo %Libro
darebbe come output 'object'.
Come potete vedere è molto comodo usare gli oggetti, soprattutto se si devono creare script molto complessi.
Prendiamo un respiro...
Adesso, prima di passare a fare del codice più complesso dobbiamo chiarire alcuni piccoli concetti, per l'esattezza quello di *array* e quello di *dizionario* sono concetti importanti, perché ci permettono di creare delle "collezioni" di valori, di stringhe o di oggetti, insomma di dati.
Un *array* è una collezione di dati variabili indicizzati per numero, il primo indice dell'array è 0 mentre l'ultimo è uguale alla grandezza dell'array meno uno (poiché si parte da zero).
Per ottenere il numero di elementi che è contenuto in un array possiamo usare la funzione ***$length(%ArrayEsempio)*** o l'espressione ***%ArrayEsempio[]#.***
Non è necessario dichiarare la grandezza dell'array come in altri linguaggi di programmazione, a mano a mano che si aggiungerà un numero, la grandezza del nostro array varierà automaticamente e se il primo elemento che assegneremo, lo assegneremo ad un indice maggiore di 0, tutte le posizioni precedenti saranno vuote.
Proviamo ad esempio:

```
%array[0]=Grifisx
%array[1]=Noldor
%array[2]=Pragma
#Stampo il contenuto di tutto l'array
echo %array[]
#Stampo la grandezza dell'array
echo $length(%array[])
#Stampo solo il primo elemento
echo %array[0]
```

oppure proviamo questo codice:

```
%array[0]=Grifisx
%array[1]=Non mostrare questo
%array[2]=Noldor
%array[5]=Segreto shhhh..
%array[8]=Pragma
for(%i=0;%i < $length(%array[]);%i+=2)echo Entry %i: \"%array[%i]\";
```

Come vedete è abbastanza semplice crearsi delle collezioni indicizzate per numero, cosi come lo è anche muoversi all'interno di esse, qui si è voluto usare un ciclo *for* ma ovviamente avremmo anche potuto usare un *foreach(%item,%Array[])echo %item* =D, oppure un ciclo *while*.

Un array potremmo anche inizializzarlo in questo modo **%array[]=$array(Grifisx,Noldor,Pragma,Madero);**
cioè utilizzando la funzione *$array(<el1>,<el2>,<el3>,<el4>,..)*.
Adesso è il momento di esaminare il fratello maggiore dell'array: il dizionario.
I dizionari non sono altro che array associativi di stringhe, per capire bene, riprendo l'esempio del manuale ufficiale:

```
%songs{Jimi Hendrix} = Voodo child
%songs{Shawn Lane} = Gray piano's flying
%songs{Mina} = Brava
%songs{Greg Howe} = "Full Throttle"
# Mostra tutto in una stringa
echo %songs{}
# Mostra tutti gli elementi del dizionario
foreach(%var,%songs{})echo %var
```

Ovviamente anche qui, come negli array $length(*%songs{}*) restituirà il numero degli elementi del dizionario.
Mentre *%songs{}* restituirà una lista degli elementi separata da virgole.
Ovviamente potremmo unire dizionari e array insieme, per avere un dizionario di array ad esempio.
Insomma possiamo gestirci le nostre collezioni come più ci piace.
Altro respiro.....

Ritornando ai nostri libri, pensate alla possibilità, ad esempio di crearvi una vostra "libreria", potendo fare un array di oggetti libro, o, meglio ancora, un dizionario: pensate ad una cosa del genere:

```
// Creiamo l'oggetto libro
%libro = $new(object,0,libro)

%libro->%titolo = "La via Lattea"
%libro->%autore =  "Piergiorgio Odifreddi"
%libro->%editore = "Longanesi"
%libro->%anno = "2008"
%libro->%contenuto ="........."

// Creiamo un dizionario utilizzando come chiave il titolo del libro
%Libreria{lavialattea}=%libro

// settiamo una stringa di ricerca, che, in questo esempio è fissa ma potrebbe essere un $0 immesso dall'utente che lancia un alias
%strtofind=lattea

/* L'indice ci servirà per darci il numero totale dei risultati, poiché di titoli contenenti la parola "lattea" potremmo averne molti. Considerate sempre che questo è solo un esempio che potreste raffinare all'infinito */
%idxSearch=1

// Cerchiamo se la parola è contenuta in una key del dizionario
foreach(%titolo,$keys(%Libreria))
{
      if($str.contains(%titolo,%strtofind))
      {
```

```
            // trovata! Stampiamo i risultati ed incrementiamo l'indice della ricerca
            echo Risultati: %idxSearch
            echo Titolo: %Libreria{%titolo}->%titolo
            echo Autore: %Libreria{%titolo}->%autore
            echo Editore: %Libreria{%titolo}->%editore
            echo Anno: %Libreria{%titolo}->%anno
            %idxSearch++
      }

}
```

Come vedete basta un po' di fantasia ed avrete tantissime soddisfazioni utilizzando gli oggetti =)

Adesso passiamo ad esaminare il concetto di *classe*.
Una classe, come anche si legge sul manuale ufficiale, è una collezione di metodi che definiscono le caratteristiche di un oggetto, però così detto non è molto chiaro (anzi, a distanza di anni dalla prima stesura di questo tutorial(2005-2010) direi che non si capisce proprio .....), quindi proviamo a capirlo, anche questa volta, con un parallelismo con la realtà:
prendiamo una biblioteca, è un edificio (oggetto) che contiene tanti libri(oggetti),ma può anche non contenerne nessuno,ciascuno con delle proprie caratteristiche, e, oltre i libri, ha anche del personale che vi lavora, c'è chi si occupa di prendere il libro che voi richiedere, c'è chi aggiunge nuovi libri al posto giusto, c'è chi apre le porte dell'edificio all'apertura e chi le chiude alla chiusura.
In pratica, oltre a contenere degli oggetti ha anche del personale che svolge funzioni che sono necessarie.
Portiamo tutto in un codice =P

```
class (libreria,object)
{
      // Funzione che apre la libreria, viene chiamata automaticamente alla creazione dell'oggetto
      constructor()
      {
            echo Benvenuti! APERTO
      }
      // Funzione che chiude la libreria, viene chiamata automaticamente alla eliminazione dell'oggetto
      destructor()
      {
            // Chiudiamo la libreria
            echo Alla prossima! CHIUSO
      }
      addbook()
      {
            //    Aggiungiamo un libro
            /*    Per aggiungere un linbro devo sapere il titolo e l'autore
                  che sono le variabili che passeremo alla funzione tramite $0 e $1
            */
            %titolo=$0
            %autore=$1
            if(%titolo=="" || %autore=="")
            {
```

```
                  echo la funzione addbook richiede due parametri,
titolo del libro ed autore
                  return
            }
            // Creiamo l'oggetto libro
            %libro = $new(object,$$,libro)
            %libro->%titolo = %titolo
            %libro->%autore =  %autore
            //  Creiamo  un  dizionario  utilizzando  come  chiave  il
titolo del libro
            @%libreria{%titolo}=%libro

      }
      searchbook()
      {
            //  Controlliamo  la  presenza  di  un  libro,  l'utente
passera' come parametro $0 la stringa da cercare
            %strtofind=$0

            // Indice che ci restituirà quanti risultati la ricerca
ci ha dato
            %idxSearch=1

            //  Cerchiamo  se  la  parola  è  contenuta  in  una  key  del
dizionario
            foreach(%titolo,$keys(@%libreria))
            {
                  if($str.contains(%titolo,%strtofind))
                  {
                  // trovata! Stampiamo i risultati ed incrementiamo
l'indice della ricerca
                  echo Risultati: %idxSearch
                  echo Titolo: @%libreria{%titolo}->%titolo
                  echo Autore: @%libreria{%titolo}->%autore
                  %idxSearch++
                  }

            }
      }
}
```

Mettiamo nello script tester (anche se vi consiglio di abituarvi ad usare il class-editor che è molto comodo per la scrittura delle classi) ed eseguiamo.
Adesso creiamo un nuovo oggetto "libreria" dalla riga di input oppure dallo script-tester:
%Libreria=$new(libreria)
adesso dalla riga di input vediamo di aggiungere un libro richiamando la funzione $addbook() che abbiamo creato:
/%Libreria->$addbook(Oggi e domani,Mario Rossi)
e poi proviamo se la ricerca funziona:
/%Libreria->$searchbook(dom)
...da me è andata alla grande =D!
Chiudiamo la libreria cancellando l'oggetto:
/delete %Libreria
Come avrete notato, la funzione destructor() è stata chiamata automaticamente, eseguendo il codice che vi si trovava dentro.
La nostra classe libreria, che vi ritroverete nel class-editor

(ctrl+shift+D) è pronta per ricevere libri, volendo potete, usando le funzioni del modulo config (guardate la doc del KVIrc), scrivervi sul disco tutte le informazioni e farvi un vostro database personale con tutti i vostri libri.
Ad esempio il costruttore potreste modificarlo in questo modo:

```
echo Benvenuti! APERTO
// Il file c'è, leggiamo tutti i valori e riempiamo il nostro dizionario
%cfg = $config.open(libreria.db)
// Apriamo il file e posizioniamoci nella sezione Libri
config.setsection %cfg Libri
/* Abbiamo scelto di salvare i valori del database secondo lo schema
   TitoloLibro=Titolo,autore
*/
// Per ciascuna chiave (che poi corrisponde a TitoloLibro)
foreach(%valori,$config.keylist(%cfg))
{
      /* Creiamo un array con il suo valore (Titolo,Autore), splittando in base alla virgola */
      %array=$str.split(",",$config.read(%cfg,%valori))
      // Creiamo, sempre per ciascuna key, dato che siamo nel ciclo foreach
      // un nuovo oggetto, settiamone i valori e piazziamolo nel dizionario
      %libro = $new(object,$$,libro)
      %libro->%titolo = %array[0]
      %libro->%autore =  %array[1]
      // Il dizionario utilizza come chiave il titolo del libro
      @%libreria{%array[0]}=%libro
      }
// Chiudiamo il file
config.close %cfg
}
```

mentre la funzione addbook() in questo modo:

```
//    Aggiungiamo un libro
/*    Per aggiungere un libro devo sapere il titolo e l'autore
      che sono le variabili che passeremo alla funzione tramite $0 e $1
*/
%titolo=$0
%autore=$1
if(%titolo=="" || %autore=="")
{
      echo la funzione addbook richiede due parametri, titolo del libro ed autore
      return
}
// Creiamo l'oggetto libro
%libro = $new(object,$$,libro)
%libro->%titolo = %titolo
%libro->%autore =  %autore
// Creiamo un dizionario utilizzando come chiave il titolo del libro
@%libreria{%titolo}=%libro
```

```
// Mettiamo in un array i valori di %titolo e %autore
%value[]=$array(%titolo,%autore)
// Apriamo il file di configurazione che conterrà il database
%cfg = $config.open(libreria.db)
// Settiamo la sezione in cui andremo a scrivere i valori [Libri]
config.setsection %cfg Libri
// Scriviamo i valori
config.write %cfg %titolo %value
// Chiudiamo il file di configurazione
config.close %cfg
```

Così avrete i vostri dati salvati sul disco e la vostra libreria potrà essere sempre tenuta aggiornata =), a voi aggiungere altre funzioni come una possibile $removebook() oppure una $totalbook() etc.etc.

Vediamo qualche altro esempio di classi:

```
class (calc,object)
{
      constructor()
      {
            echo Usage\:
            echo Per calcolare la somma di 2 numeri:
            echo \/\%Calc\-\>\$somma2\(\x\,\y\)
            echo Per calcolare il quoziente di 2 numeri:
            echo \/\%Calc\-\>\$div\(\x\,\y\)
      }
      somma2()
      {
/*$0 e $1 sono i valori che noi passeremo alla funzione richiamandola
 ad esempio in "\%Calc->$somma2(8,6)" $0=8 e %1=6 */
            echo $($0+$1)
      }
      div()
      {
            echo $($0/$1)
      }

}
%Calc=$new(calc)
```

L'esecuzione di questo script avrà l'effetto di darvi la possibilità di sommare o dividere 2 numeri tra di loro utilizzando le funzioni che voi avrete creato $somma2() e $div().
Provate ad esempio a fare dalla riga di input
/%Calc->$div(10,5)
Come si può vedere per prima cosa è stata creata una classe *calc* che è di tipo *object* (potrebbe anche essere di tipo widget, ad esempio, per gli oggetti grafici, ma questo lo vedremo più avanti) e , all'interno della classe, sono state create 3 funzioni (o metodi come preferite) *$constructor(), $somma2()* e *$div();* dopo questo creiamo l' oggetto con il codice *"%Calc=$new(calc)"* e a questo punto possiamo usare le funzioni che ho realizzato all'interno della classe, come vedete basta richiamarle tramite il simbolo "->".
Ricordiamo tra l'altro che la funzione *$(<espressione_aritmetica>)* esegue il calcolo dell'espressione contenuta all'interno delle

parentesi.
Come già detto il codice dentro *$constructor()* viene eseguito alla creazione della classe in modo automatico, questo è il posto migliore dove, ad esempio, inizializzare variabili.
Altro esempio:

```
class(rubrica,object)
{
      addName()
      {
            $$->%name=$0-
      }
      addSurname()
      {
            $$->%surname=$0-
      }
      addTel()
      {
            $$->%tel=$0-
      }

}
%Rubrica=$new(rubrica)
```

Spieghiamo, per completezza, cosa indicano "*$$->*" e "*$0-*":
il *$0-* è il parametro che la funzione riceve (come gli alias ricordate? Funziona allo stesso modo, cioè quando chiameremo la funzione $addName() dovremo passargli il nome da aggiungere).
$$, invece,* sta per "Variabile appartenente a questa classe", cioè se dobbiamo creare una variabile di classe (che ha vita fino alla distruzione dell'oggetto), dobbiamo crearla con *$$-> oppure con $this->, allo stesso modo, se dovessimo chiamare una funzione della stessa classe all'interno di un'altra funzione dovremmo chiamarla con *$$->$funzione()*.
Al posto di "$$->", come abbiamo visto nella classe precedentemtne creata, la libreria, si può anche utilizzare il simbolo "@", così la sintassi diventerà @$funzione oppure @%variabile.

Ora facciamo un cenno su come si possa salvare e leggere da un file di testo, poiché se abbiamo delle collezioni, e vogliamo conservarcele, la prima cosa da fare e metterle su file.
Vediamo:
$file.read("file_da_leggere")
file.write("file_da_scrivere")
la prima funzione ci permette di leggere da un file di testo mentre il secondo (è un comando non una funzione, lo vedere dal fatto che non inizia per $) ci permette di scriverci dentro
Quindi mettiamo il caso che avessimo un file di testo (tipo "userdatabase") con dentro una lista di nomi separati da virgole (tipo "Grifisx,Noldor,Pragma,Madero") e che volessimo mettere questi nick in un array, come dovremmo fare? Niente di più semplice:
%users[] = $str.split(",",$file.read($file.localdir(usersdatabase)))
la funzione *$str.split* serve per dividere una stringa in base ad un separatore che gli diamo (in questo caso la virgola ",");
mentre la funzione *$file.localdir()* restituisce la directory delle configurazioni locali del KVIrc (provate subito uno "*/echo $file.localdir()*"):

quindi, in questo modo, avremo creato l'array %users[], che al suo interno ha i nick che abbiamo letto dal file.
Se volessimo scrivere sul file avremmo invece:
file.write $file.localdir(usersdatabase) %users[]
Vediamo di testare il tutto nello script tester:

```
%users[]=$array(Grifisx,Noldor,Pragma,Madero);
file.write $file.localdir(usersdatabase) %users[];
%usersNew[]=$str.split(",",
$file.read($file.localdir(usersdatabase)));
%idx=0;
while(%idx!=%usersNew[]#)
{
      echo Utente: %usersNew[%idx];
      %idx++;
}
file.remove $file.localdir(usersdatabase)
```

Come avrete potuto notare, eseguendo lo script, il file è stato creato, letto e successivamente rimosso con il comando *file.remove* (per vedere gli altri comandi disponibili per la gestione dei file, controllate nel manuale alla voce *Comandi* e poi lettera *f)*.

Adesso, con le nuove nozioni che abbiamo imparato, andiamo a fare qualche cosa di più complesso ed allo stesso tempo più concreto: un piccolo gestore di appunti =)

```
class(appunti,object)
{
      constructor()
      {
/* Nel costruttore faccio apparire a video le istruzioni d'uso,
 ricordando che il costruttore è la prima cosa che viene eseguita
 appena la classe viene creata */
            echo $k(5,8) \-\-\-Appunti\-\-\-\
            echo $k(5,8) Comandi manuali:
            echo $k(5,8) \/\%Appunti\-\>\$viewApp          $b
Visualizza gli appunti presenti
            echo $k(5,8) \/\%Appunti\-\>\$addApp\(appunto\)$b
Aggiunge un nuovo appunto
            echo $k(5,8) \/\%Appunti\-\>\$delApp\(appunto\)$b
Cancella un appunto
      }
// Funzione per aggiungere un appunto
      addApp()
      {
// Creo un array e lo riempio con il contenuto del file
            @%appuntos[] = $str.split(",",
$file.read($file.localdir(kvircAppuntidb)))
            @%ap = 0
/* Ora creerò un dizionario, perché è più semplice lavorare con
 stringhe e sono più facilmente individuabili, ma allo stesso tempo
 perché cosi avremo una applicazione pratica dell'uso dei dizionari,
 e dopo lo riempio con il contenuto dell'array, questo mi permetterà
 di avere una cosa del tipo:
 %Array{ciao}=ciao , in modo che, quando vorrò poi cancellarlo, non
 dovrò preoccuparmi del suo indice numerico dato che il mio indice è
 uguale all'elemento, in pratica lo rintraccerò sempre.*/
```

```
            while(@%ap < @%appuntos[]#)
            {
                  @%ElencoAppunti{@%appuntos[@%ap]} = @%appuntos[@
%ap]
                  @%ap++;
            }
            @%ProvaAP =  $0-
            @%ElencoAppunti{@%ProvaAP} = @%ProvaAP
// Visualizziamolo a schermo tutti gli appunti
            foreach(@%var,@%ElencoAppunti{}) echo $k(5,8)@%var
// Conserviamo sul file
            file.write $file.localdir(kvircAppuntidb) @
%ElencoAppunti{}
            echo $k(5,8) Fine Lista : @%ElencoAppunti{}# \) Appunti
presenti
      }

// Funzione per cancellare un appunto
      dellApp()
      {
// Come la funzione precedente
            @%appuntos[] = $str.split(",",
$file.read($file.localdir(kvircAppuntidb)))
            @%ap = 0
            while(@%ap < @%appuntos[]#)
            {
                  @%ElencoAppunti{@%appuntos[@%ap]} = @%appuntos[@
%ap]
                  @%ap++;
            }
            @%ProvaAP =  $0
/* Ecco che posso cancellarlo facilmente senza preoccuparmi dell'
indice dato che utilizzo il suo stesso nome come indice */
            @%ElencoAppunti{@%ProvaAP} = ""
// svuotato =D
            foreach(@%var,@%ElencoAppunti{}) echo $k(5,8)@%var
            file.write $file.localdir(kvircAppuntidb) @
%ElencoAppunti{}
            echo $k(5,8) Fine Lista : @%ElencoAppunti{}# \) Appunti
presenti
      }
// Funzione per visualizzare gli appunti
      viewApp()
      {
// Tutto come sopra
            @%appuntos[] = $str.split(",",
$file.read($file.localdir(kvircAppuntidb)))
            @%ap = 0
            while(@%ap < @%appuntos[]#)
            {
                  @%ElencoAppunti{@%appuntos[@%ap]} = @%appuntos[@
%ap]
                  @%ap++;
            }
            foreach(@%var,@%ElencoAppunti{})echo $k(5,8) @%var
            echo $k(5,8) Fine Lista : @%ElencoAppunti{}# \) Appunti
presenti
      }
```

```
}
// Creo il mio oggetto %Appunti
%Appunti=$new(appunti)
```

Non credo che ci sia molto da spiegare, poiché alla fine, se avete capito bene i concetti di *array, dizionario* e di lettura e scrittura dei file (e ovviamente di *classe*), lo script si spiegherà da solo.
Si potrebbe fare, senza ombra di dubbio, uno script migliore (o utilizzare costrutti certamente più funzionali di quelli utilizzati), ma ci si è limitati ad usare tutti (e solo) gli argomenti trattati, cercando di riunirli in un unico script.
Lo farete sicuramente migliore quando avrete maturato tutti gli altri concetti che il linguaggio del KVIrc nasconde =).

/ECHO STOP.

- - - - - - -- - - - - - - -- - - - - - - - - -- - - - - -- - - - - -
" Tu vedi cose e ne spieghi il perché, io invece immagino cose che non sono mai esistite e mi chiedo perché no." (George Bernad Shaw)
- - - - - - - - -- - - - - - - -- -- - - - - - - - - - - - - - - - -
Grifisx (Tonino Imbesi)