PHP:
IMAGE PROCESSING

Libreria GD

- La manipolazione di immagini in PHP è possibile tramite la **libreria GD**:
  http://php.net/manual/en/book.image.php
- A partire da PHP 4.3 è installata di default
- Comprende **funzioni** per:
  - aprire immagini esistenti (JPEG, PNG, GIF...) o per crearne di nuove
  - disegnare forme e scrivere stringhe
  - gestire il colore e l'opacità
  - applicare filtri ed operatori per l'image processing
  - restituire al browser il flusso di dati dell'immagine nel modo corretto

Come funziona in pratica

- La pipeline comprende **4 passi principali**:

1) Creare una struttura in memoria per contenere l'immagine (tipo *resource*)
2) Aggiungere contenuto all'immagine con le primitive della libreria
3) Restituire l'immagine al browser assieme agli *header HTTP* corretti per il formato scelto
4) Eliminare la struttura precedentemente creata dalla memoria

1) Creare la struttura per l'immagine

Per creare una nuova immagine si usano le **funzioni**:

```
resource imageCreate ( int $x_size, int $y_size )
resource imageCreateTrueColor ( int $x_size, int $y_size )
```

Con imageCreate() viene creata una immagine **palette-based**, mentre con imageCreateTrueColor() ne viene creata una in modalità **true-color**. L'oggetto restituito è la struttura in cui sarà contenuta l'immagine.

Questo è l'**identificativo** che dovrà essere passato alle varie funzioni della libreria per poter manipolare l'immagine stessa.

E’ possibile anche caricare immagini **da file esterni**

```
resource imageCreateFromPNG ( string $filename )
resource imageCreateFromJPEG ( string $filename )
resource imageCreateFromWBMP ( string $filename )
resource imageCreateFromGD ( string $filename )
```

oppure **da una stringa** già presente in memoria

```
resource imageCreateFromSTRING ( string $data )
```

(quest’ultima modalità è molto utile, ad esempio, quando si devono estrarre immagini da campi blob di un database)

2) Aggiungere contenuto all'immagine

Una volta creata la struttura per l'immagine, è possibile utilizzare le **funzioni messe a disposizione dalla libreria GD** per manipolare l'immagine stessa.

Per utilizzare una funzione, è necessario passare come parametro l'identificativo dell'immagine creata con le funzioni imageCreate() e imageCreateTrueColor().

NB: attenzione alla **dimensione delle immagini**: PHP ha un limite *(di solito 8/16 MB)* per la memoria a disposizione!
(le immagini contenute in memoria non sono compresse, anche se provengono ad esempio da un file JPEG compresso!)

3) Restituire l'immagine al browser

Per restituire l'immagine *(come flusso di byte)* al browser, o a chi l'ha richiesta, si usano ad esempio le **funzioni**:

```
bool imageJPEG ( resource $image [, string $filename [, int $quality]])
bool imagePNG ( resource $image [, string $filename [, int $quality]])
```

A seconda del formato scelto, si devono inviare gli **headers HTTP** corretti (*prima dell'immagine stessa!*):

```
... creazione dell'immagine $image
... manipolazione/aggiunta di contenuto
header( "Content-type: image/jpeg" );
imageJPEG ( $image );
```

4) Eliminare la struttura dalla memoria

E’ importante **ricordarsi di eliminare le strutture delle immagini dalla memoria** appena possibile, altrimenti il server potrebbe andare in crash!

La funzione da usare è:

```
bool imageDestroy ( resource $image )
```

Ovviamente, questa funzione non deve essere chiamata prima di aver restituito al browser l’immagine stessa!

Interazione con OpenLaszlo

- Il flusso di dati restituito da uno script PHP che fa uso della libreria GD **genera in tutto e per tutto una immagine standard**!

- Per OpenLaszlo non fa alcuna differenza se l'immagine da caricare tramite il tag

```
<view resource="...url dell'immagine da caricare..." />
```

sia una immagine statica oppure generata da uno script PHP!

- Quindi avviene tutto come prima: basta indicare semplicemente **nell'attributo resource/source l'url di uno script PHP** che genererà l'immagine:

```
<view width="100" height="100" stretches="both"
    resource="http://miosito.com/image.php?str=ciao&col=red"
/>
```

- E' possibile **passare parametri allo script PHP** tramite la relativa *querystring* dell'url
- Un **errore frequente** è quello di provare a caricare l'immagine attraverso il tag <dataset>:
  <dataset> = "testo XML" mentre <view> = "immagini"!

OpenLaszlo GUI

```
<view
  resource="...url a script PHP..."
/>
```

Parametri passati tramite **querystring** (modalità GET) nell'attributo *resource/source*

Immagine restituita con i corretti **headers HTTP**

Server PHP

```
<?php
  ...creazione immagine $image...
  header( "Content-type: image/jpeg" );
  imagejpeg( $image );
?>
```

Esempio

image.php:

```
<?php
    $image = imageCreate( 100, 100 );
    imageColorAllocate( $image, $_GET['r'], $_GET['g'], $_GET['b'] );
    header('Content-type: image/png');
    imagePNG( $image );
    imageDestroy( $image );
?>
```

index.lzx:

```
<view width="100" height="100" stretches="both"
    resource="http://miosito.com/image.php?r=255&amp;amp;g=0&amp;amp;b=0"
/>
```

Funzioni: colore

- La libreria GD adotta il **modello additivo** del colore: un sistema basato sulla *luce emessa.*
- Tutti i formati delle immagini manipolabili con GD usano la **modalità RGB** per il colore:
  - nero = nessuna emissione (es: 0x00 00 00)
  - bianco = emissione totale (ex: 0xFF FF FF)
- E' possibile manipolare immagini in **due modalità**:
  - **True Color**: 3 canali (R,G,B) con 8 bit/canale (0-255)
  - **Indexed Color**: colori memorizzati in una palette (*look-up table*)

- Alcune funzioni:

int imageColorAllocate(resource $image, int $r, int $g, int $b)
Allocca un colore da usare nell'immagine. Ogni colore, per essere usato, deve prima essere allocato. Il primo, di default, è lo sfondo.

void imageColorSet(resource $image, int $index, int $r, int $g, int $b)
Reimposta i valori RGB di un colore allocato.

bool imageColorDeallocate(resource $image, int $index)
Rimuove un colore dalla palette.

int imageColorAt(resource $image, int $x, int $y)
Restituisce il colore al pixel (x,y).

int imageColorClosest(resource $image, int $r, int $g, int $b)
Restituisce il colore nella palette più somigliante a quello specificato.

int imageColorTransparent(resource $image, int $c)
Imposta il colore trasparente per l'immagine al colore $c.

Funzioni: disegnare

- Sono disponibili delle **primitive per disegnare** punti, linee, rettangoli, poligoni, ellissi ecc.
- Il **sistema di riferimento** per tutte le funzioni è l'angolo in alto a sinistra (0,0) dell'immagine.
- I colori devono essere precedentemente allocati per poter essere utilizzati da queste primitive.
- Lo stile di disegno (*tratto*, *spessore*, *tile*, *brush*) è personalizzabile con apposite funzioni.

Alcune funzioni:

bool imageSetPixel(resource $image, int $x, int $y, int $c)
Disegna un punto di colore $c nel pixel di coordinate ($x,$y).

bool imageLine(resource $im, int $x1,int $y1, int $x2,int $y2, int $c)
bool imageRectangle(resource $im, int $x1,int $y1, int $x2,int $y2, int $c)
bool imageEllipse(resource $im, int $x,int $y, int $w,int $h, int $c)
bool imageArc(resource $im, int $x,int $y, int $w,int $h, int $s,int $e, int $c)
Disegnano rispettivamente una linea, un rettangolo, una ellisse ed un arco con il colore $c. Esistono anche le versioni '*filled*'.

bool imagePolygon(resource $image, array $points, int $n, int $c)
Disegna un poligono specificato dai $n vertici contenuti nell'array $points (specificati come sequenza di 2*$n valori float).

bool imageFill(resource $image, int $x, int $y, int $c)
Riempie una regione con il colore $c a partire dal punto ($x,$y).

Funzioni: stringhe

- E’ possibile scrivere delle **stringhe** nell’immagine.
- Sono disponibili dei **font** *built-in*, ma è anche possibile caricarne di *user-defined.*
- Funzioni:

bool imageString(resource $im, int $font, int $x, int $y, string $s, int $c)
Scrive la stringa $s alle coordinate ($x,$y) con il colore $c e il font $font.

bool imageLoadFont(string $file)
Carica un font *user-defined* definito come bitmap in formato binario.

bool imageFontWidth(int $font) / bool imageFontHeight(int $font)
Restituisce larghezza/altezza di un font.

Funzioni: copia

```
bool imageCopy(
    resource $dst_image, resource $src_image,
    int $dst_x, int $dst_y,
    int $src_x, int $src_y, int $src_w, int $src_h
)
```

Copia una porzione di una immagine in un'altra.

```
bool imageCopyResized(
    resource $dst_image, resource $src_image,
    int $dst_x, int $dst_y, int $src_x, int $src_y,
    int $dst_w, int $dst_h, int $src_w, int $src_h
)
```

Copia e ridimensiona una porzione di una immagine in un'altra. Se le dimensioni nelle due immagini non coincidono, verrà effettuato un ridimensionamento.

La versione imageCopyResampled() utilizza l'interpolazione per ridimensionare.

Funzioni: utilità

array gd_info()
Restituisce una serie di informazioni sulla libreria installata nel sistema.

int imageSx(resource $image) / int imageSy(resource $image)
Restituiscono le dimensioni di un oggetto immagine.

array getImageSize(string $filename)
Restituisce una serie di informazioni di una immagine (contenuta in un file).

bool imageAntialias(resource $image, bool $enabled)
Abilita la funzione di antialiasing per le primitive di disegno.

resource imageRotate(resource $image, float $angle, int $bgcolor)
Ruota l'immagine di un angolo $angle (in gradi).

Funzioni: image processing

- La libreria GD mette a disposizione alcuni **filtri standard** tramite la funzione:

```
bool imageFilter(
      resource $image,
      int $filtertype [, int arg1  [, int arg2 .....]]
)
```

dove *$filtertype* può essere ad esempio:

- IMG_FILTER_GRAYSCALE
- IMG_FILTER_EDGEDETECT
- IMG_FILTER_SMOOTH

- Altrimenti, tramite la funzione

```
bool imageConvolution (
    resource $image,
    array $mask, float $div, float $offset
)
```

è possibile effettuare la **convoluzione con una qualsiasi maschera 3x3** definita dall'utente.

- La maschera si specifica in questo modo:

```
$mask = array( array(2, 0, 0), array(0, -1, 0), array(0, 0, -1) );
```

- Esempi: mean filter, gaussian blur, Prewitt/Sobel operator, Laplace operator ecc...

5° esercitazione

1. Creare dinamicamente una immagine che:
    - visualizzi una stringa passata dall'utente;
    - sia possibile specificare, oltre alla stringa, anche alcuni parametri della stringa stessa, come il colore, la posizione ecc..
2. Creare dinamicamente una miniatura *(thumbnail)* di una immagine qualsiasi, in cui:
    - l'immagine su cui lavorare è specificata come parametro;
    - la dimensione massima è indicata sempre come parametro.
3. Aggiungere alla stringa dell'immagine al punto 1 un'effetto ombreggiamento.