

## Studia Podyplomowe "Aplikacje i Usługi Internetowe"


Materiały do przedmiotu "Tworzenie Witryn Internetowych - PHP"

### Niektóre operacje graficzne w PHP

---

#### Co jest potrzebne do operacji graficznych

Operacje graficzne w PHP, jak w przypadku wielu innych rozszerzeń tego środowiska, nie są realizowane całkowicie przez PHP - zapewnia on tylko funkcje dostępu do elementów zawartych w innych bibliotekach. W przypadku grafiki oficjalnie "wspomagana" biblioteką jest **gd**, i to od jej wersji zależą możliwości graficzne środowiska PHP.

 W przypadku wersji PHP pod Windows biblioteka *gd* powinna się znajdować wśród dostarczonych rozszerzeń - bibliotek DLL w `ext` jako `php_gd2.dll`, trzeba tylko ją odblokować w `php.ini`.  
W przypadku dystrybucji linuxowych dostępność zależy od dystrybucji, natomiast witryna PHP informuje, że od wersji 4.3 biblioteka *gd* powinna być umieszczona w każdej "ściągalnej" oficjalnej wersji instalacyjnej.

Aktualne możliwości stosowanej bibliotek *gd* można zobaczyć na przykład przy pomocy `phpinfo()`:

#### gd

<b>GD Support</b>	enabled
<b>GD Version</b>	bundled (2.0.28 compatible)
<b>FreeType Support</b>	enabled
<b>FreeType Linkage</b>	with freetype
<b>FreeType Version</b>	2.1.9
<b>GIF Read Support</b>	enabled
<b>GIF Create Support</b>	enabled
<b>JPG Support</b>	enabled
<b>PNG Support</b>	enabled
<b>WBMP Support</b>	enabled
<b>XPM Support</b>	enabled
<b>XBM Support</b>	enabled

Z informacji wynika na przykład jakiego typu pliki graficzne są obsługiwane przez bibliotekę (JPEG,GIF,PNG itd.) lub czy ma ona wsparcie dla czcionek *Free Type*.

---

#### Tworzenie obrazów

Obraz - grafika w PHP nie ma formalnie określonego typu, nie jest to ani JPEG, ani PNG, ani nic podobnego. Reprezentowana jest przez pewien *zasób*, na którym można wykonywać odpowiednie operacje. Zasób ten można uzyskać w dwojaki sposób:

- albo tworząc go "od zera", jako abstrakcyjny obraz
- albo uzyskując go z już istniejącego obrazu (np. pliku), o ile używana biblioteka *gd* daje możliwość odczytu obrazu tego typu.

W przypadku pierwszego sposobu najczęściej używa się funkcji `imagecreate($width, $height)` lub `imagecreatetruecolor($width, $height)`. Druga z tych funkcji jest

stosunkowo nowa, ale od momentu pojawienia się zdecydowanie zalecana, gdyż daje większe możliwości, nie ograniczając palety kolorów do 256 jak pierwsza funkcja. Parametry funkcji określają odpowiednio szerokość i wysokość tworzonego obrazu.

Przykład - tworzymy obraz o wymiarach 400x200:

```
$image=imagecreatetruecolor(400,100);
```

Jeżeli chcemy zacząć pracę od już istniejącego obrazu, do stworzenia zasobu użyjemy funkcji z rodziny `imagecreatefrom...`, gdzie końcówka nazwy odpowiada typowi obrazu, np. `imagecreatefromjpeg` daje nam zasób z obrazu typu JPEG, `imagecreatefrompng` daje nam zasób z obrazu typu PNG itd. Jako parametr tych funkcji podaje się ścieżkę do istniejącego pliku z grafiką, może to być również URL.

Przykład - wczytujemy logo PG:

```
$image=imagecreatefromjpeg("http://www.pg.gda.pl/szuk/img/herb_sz.jpg");
```

## Generowanie obrazu danego typu

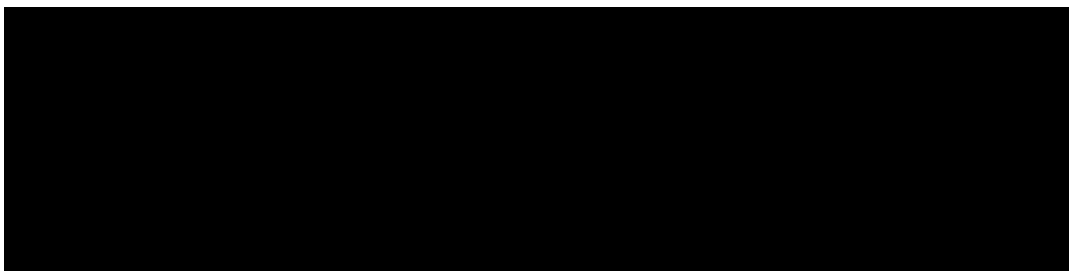
Korzystając z uzyskanego zasobu możemy wykonywać na nim różne operacje graficzne (o nich za chwilę), ale nic to nam nie da, gdy nie stworzymy wyniku końcowego naszych operacji, czyli gotowego obrazu danego typu. Do tego celu służą, podobnie jak w przypadku uzyskiwania zasobu z już istniejącego obrazu, funkcje z rodziny `image...`, np. `imagejpeg` lub `imagepng`. Pierwszym, obowiązkowym parametrem funkcji jest nasz zasób. Bez podanych żadnych dodatkowych parametrów każda z tych funkcji generuje obraz odpowiedniego dla siebie typu na standardowe wyjście. Jako drugi parametr można podać ścieżkę do pliku, wtedy wynik działania funkcji znajdzie się w tym pliku zamiast na standardowym wyjściu. Pozostałe parametry opcjonalne zależą już indywidualnie od danej funkcji.

W przypadku generowania wyniku na standardowe wyjście, najczęściej jest to przekaz przez internet do przeglądarki użytkownika. Trzeba wtedy pamiętać o ustawieniu nagłówka odpowiedzi, aby przeglądarka zrozumiała, że to, co odbiera, jest obrazem PNG, a nie tekstem.

Przykład - wysyłamy obraz z pierwszego przykładu jako PNG:

```
$image=imagecreatetruecolor(400,100);  
header("Content-type: image/png");  
imagepng($image);
```

W wyniku w przeglądarce uzyskamy:



Należy przy tym pamiętać o tym, że w do klienta można przesłać odpowiedź *tylko jednego typu*, stąd nie możemy jednocześnie wysłać np. kodu strony HTML i obrazków w niej zawartych (chyba, że skorzystamy z odpowiedzi skompresowanej, ale to już jest bardziej zaawansowane działanie).

## Użyteczne informacje o obrazie

Gdy mamy już zasób - obraz, na którym można pracować, warto skorzystać z pewnych użytecznych funkcji informacyjnych:

- `imagesx($image)`, `imagesy($image)` - pozwalają na uzyskanie szerokości i wysokości obrazu. Warto używać tych funkcji, gdy chcemy np. aby nasze operacje graficzne były skalowane wraz z obrazem. Jeżeli chcemy, by coś np. było w środku obrazu, to zamiast zakładać, że środek to jest punkt (200,100), możemy ustalić go jako `(imagesx($image)/2, imagesy($image)/2)`.
- `imageistruecolor($image)` - daje informację, czy nasz obraz ma paletę tzw. *truecolor*, czy tylko 256 kolorów.
- `imagecolorat($image, $x, $y)` - daje w wyniku kolor piksela na pozycji ( $x, y$ ). W przypadku palety 256 kolorów, jest to indeks koloru w paletce. W przypadku *truecolor* jest to liczba całkowita, gdzie na odpowiednich bitach znajdują się składowe RGB.

Przykład:

```
$rgb = imagecolorat($image, 1, 5);
$red = ($rgb >> 16) & 0xFF;
$green = ($rgb >> 8) & 0xFF;
$blue = $rgb & 0xFF;
```

## Kolory

Przyjęte jest, że aby rysować w danym kolorze na obrazie, trzeba ten kolor najpierw dla obrazu "zaalokować". Podstawową funkcją do tego celu jest `imagecolorallocate($image, $red, $green, $blue)`. Funkcja daje nam w wyniku liczbę całkowitą identyfikującą kolor, stworzoną w oparciu o składowe RGB, czyli liczby od 0 do 255, np.:

```
$maroon = imagecolorallocate($image, 128, 0, 0);
```

Możliwe jest również "usunięcie" koloru z obrazu poprzez adekwatną do poprzedniej funkcję `imagecolordeallocate($image, $color)`, gdzie jako `$color` podajemy identyfikator koloru utworzonego przy pomocy `imagecolorallocate`.

## Przeźroczystość

W ramach obrazu istnieje również możliwość ustawienia koloru jako "przeźroczysty". Jest to wysoce użyteczne np. w grafikach planowanych jako wyświetlane na innym tle.

Przeźroczyste tło można uzyskać na kilka sposobów. Jeden z nich to:

```
/* alokujemy biały kolor */
$white=imagecolorallocate($image,255,255,255);
/* wypełniamy cały nasz obraz tym kolorem */
imagefill($image,0,0,$white);
/* ustalamy, że nasz kolor będzie przeźroczysty */
imagecolortransparent($image,$white);
```

W obrazach typu *truecolor* możliwa jest bardziej zaawansowana zabawa z przeźroczystością poprzez ustalanie stopnia przeźroczystości danego koloru, czyli zabawę z tzw. *kanalem alfa*. Przydatna może być tutaj funkcja `imagecolorallocatealpha`, będąca rozszerzeniem `imagecolorallocate` o dodatkowy parametr określający wartość kanału alfa pomiędzy 0 i 127, gdzie 0 oznacza "pełen kolor", 127 - "pełną przeźroczystość".

Przykład:

```
$image=imagecreatetruecolor(400,100);
$white=imagecolorallocate($image,255,255,255);
imagefill($image,0,0,$white);
imagecolortransparent($image,$white);
$start=0;
$end=imagesx($image);
$range=$end-$start;
for ($i=$start;$i<$end;$i++) {
    /* wyznaczamy alfa według trochę zmodyfikowanej funkcji Gaussa */
    $alpha=127-floor(127*exp(-0.1*pow($i-$range/2-$start,2)/$range));
    $c=imagecolorallocatealpha($image,255,0,0,$alpha);
    imagefilledrectangle($image,$i,0,($i+1),imagesy($image),$c);
}
header("Content-type: image/png");
imagepng($image);
```

Efekt (obraz został specjalnie umieszczony w ramce, żeby było widać, gdzie się zaczyna):



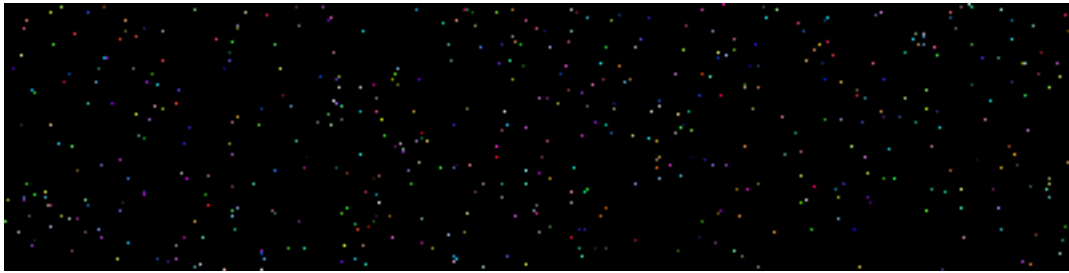
## Punkty i linie

Punkt, czyli pojedynczy piksel możemy w PHP rysuje się w PHP przy pomocy funkcji `imagepixel($image, $x, $y, $color)`, gdzie `$x` i `$y` ustalają pozycję punktu, a `$color` jest zaalokowanym kolorem.

Przykład - losowe punkty o losowych kolorach:

```
$image=imagecreatetruecolor(400,100);
for ($i=0;$i<100;$i++) {
    $x=rand(0,imagesx($image));
    $y=rand(0,imagesy($image));
    $color=imagecolorallocate($image,
        rand(0,255),rand(0,255),rand(0,255));
    imagepixel($image,$x,$y,$color);
}
header("Content-type: image/png");
imagepng($image);
```

Efekt:

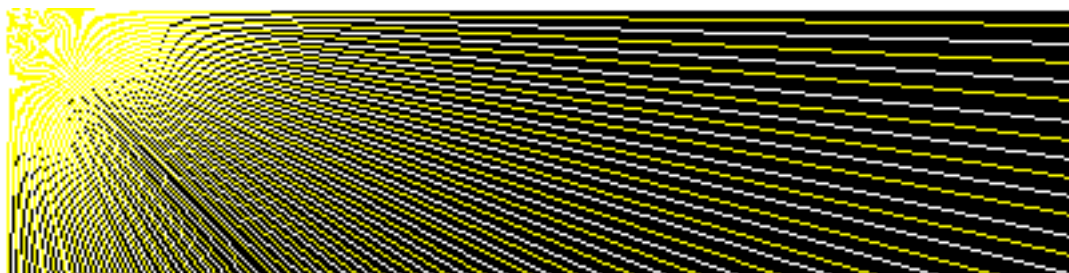


Linie w PHP rysuje się przy pomocy `imageline($image, $x1, $y1, $x2, $y2, $color)`, gdzie  $(x_1, y_1)$  oznacza punkt początkowy rysowania,  $(x_2, y_2)$  - punkt końcowy, a `$color` - kolor rysowania.

Przykład:

```
$image=imagecreatetruecolor(400,100);
for ($a=0;$a<=90;$a++) {
    $rad=$a*M_PI/180;
    if ($a<=45) {
        $x=imagesx($image);
        $y=$x*tan($rad);
    }
    else {
        $y=imagesy($image);
        $x=$y*tan(M_PI/2 - $rad);
    }
    $color=imagecolorallocate($image,255,255,($a%2)?0:255);
    imageline($image,0,0,$x,$y,$color);
}
header("Content-type: image/png");
imagepng($image);
```

Efekt:



---

## Figury geometryczne

PHP posiada funkcje ułatwiające rysowanie figur geometrycznych, jak:

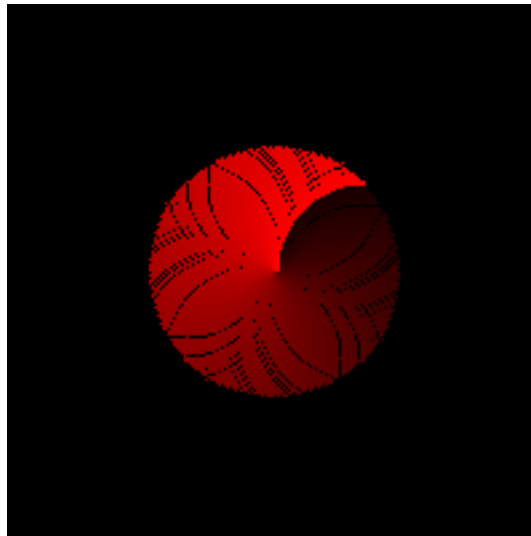
- `imagerectangle($image, $x1, $y1, $x2, $y2, $color)` - rysuje prostokąt o krawędzi w zadanym kolorze, gdzie  $(x_1, y_1)$  oznacza lewy górny róg figure,  $(x_2, y_2)$  - prawy dolny
- `imageellipse($image, $cx, $cy, $width, $height, $color)` - rysuje elipsę o krawędzi w zadanym kolorze, gdzie  $(cx, cy)$  oznacza środek elipsy, `$width` i `$height` jej szerokość i wysokość. W szczególności, ustalając `$width=$height` możemy narysować okrąg o średnicy równej tej wartości.
- `imagearc($image, $cx, $cy, $width, $height, $start, $end, $color)` - rysuje łuk elipsy o krawędzi w zadanym kolorze, dodatkowe parametry `$start` i `$end` oznaczają

początkowy i końcowy kąt łuku w stopniach. 0 stopni zlokalizowane jest na tzw. "godzinie 3", a łuk rysowany jest zgodnie z ruchem wskazówek zegara.

Przykład:

```
$image=imagecreatetruecolor(200,200);
/* punkt styczności łuków */
$cx=imagesx($image)/2;
$cy=imagesy($image)/2;
/* średnica */
$d=imagesy($image)/3;
/* jasność początkowa */
$bts=40;
for ($a=0;$a<=360;$a+=2) {
    $rad=$a*M_PI/180;
    /* środek aktualnego łuku */
    $ncx=$cx+$d/2*cos($rad);
    $ncy=$cy+$d/2*sin($rad);
    /* aktualny kolor */
    $color=imagecolorallocate($image, ($a+$bts)*255/(360+$bts),0,0);
    imagearc($image,$ncx,$ncy,$d,$d,180+$a,270+$a,$color);
}
header("Content-type: image/png");
imagepng($image);
```

Efekt:



Istnieje możliwość stworzenia dowolnego zamkniętego wielokątu przy użyciu `imagepolygon($image,$points,$num_points,$color)`. W tym przypadku parametr `$points` zawiera tablicę koordynat kolejnych wierzchołków - wierzchołek 1 to (`$points[0]`, `$points[1]`), wierzchołek 2 to (`$points[2]`, `$points[3]`) itd. Parametr `$num_points` zawiera ilość wierzchołków wielokątu.



W dokumentacji PHP jest ostrzeżenie, że funkcja może zmieniać zawartość tablicy `$points`.

W PHP dostępne są również wersje "wypełnione" figur geometrycznych, zawierające słowo "filled" np. `imagefilledrectangle`. Funkcje te zawierają w zasadzie te same parametry, co wersje "puste", tyle że przy ich wykonaniu zadany kolor nie jest rysowany tylko krawędź, ale całość figury.

## Tekst i czcionki

Najprostszym sposobem napisania tekstu na obrazie jest użycie funkcji `imagestring($image, $font, $x, $y, $string, $color)`. Parametr `$font` określa tutaj numer wbudowanej czcionki, od 1 do 5, a praktycznie jej wielkość (1 - najmniejsza). Parametry `$x` i `$y` określają górny lewy róg, od którego zacznie się pisanie. Istnieje również funkcja `imagestringup`, która pisze podany łańcuch znaków od góry do dołu.

Przydatne są również funkcje pomocnicze do uzyskiwania wielkości wbudowanych czcionek, `imagefontwidth($font)` oraz `imagefontheight($font)`

Przykład:

```
$image=imagecreatetruecolor(20,100);
$color=imagecolorallocate($image,255,255,255);
$y=0;
for ($font=5;$font>=1;$font--) {
    imagestring($image,$font,2,$y,"A",$color);
    $y+=imagefontheight($font)+2;
}
header("Content-type: image/png");
imagepng($image);
```

Efekt:



Ciekawsze efekty można uzyskać stosując np. czcionki *True Type*.

Do tego celu można skorzystać z funkcji `imagettftext($image, $size, $angle, $x, $y, $color, $fontfile, $text)`, która wypisuje podany `$text` czcionką *True Type* o rozmiarze `$size`, zawartą w pliku o ścieżce podanej w `$fontfile`, zaczynając od punktu `($x, $y)`, pod kątem `$angle` i w kolorze `$color`. Punkt `($x, $y)` wyznacza nam początek *linii bazowej* pisania, czyli odpowiednik linii w zeszyte szkolnym. W praktyce nad linią bazową są wielkie litery oraz małe jak "a,b,c,d,e,h,i,k,...", natomiast "ogonki" liter "f,g,j,..." przeważnie kończą się poniżej linii bazowej.

Powyższa funkcja daje w wyniku tablicę zawierającą koordynaty na obrazie prostokąta, w którym znajduje się napisany tekst, kolejno lewy dolny narożnik, prawy dolny, prawy górny i lewy górny. Podobny efekt możemy uzyskać bez wypisywania tekstu, przy pomocy funkcji `imagettfbbox($size, $angle, $fontfile, $text)`, ale należy przy tym uważać, gdyż ze względu na brak lokalizacji w ramach obrazu, w wyniku działania funkcji linia bazowa będzie w punkcie (0,0), stąd kordynata pionowa górnego wierzchołka będzie ujemna - dzięki temu jednak możemy się dowiedzieć o ile tekst wychodzi poza linię bazową w jedną lub w drugą stronę.

Przykład - wypisujemy na środku obrazu tekst czcionką Zurich Calligraphic Italic:

```
$text="To jest bardziej sympatyczny tekst";
$size=18;
$fontfile="ZurichCalligraphicItalic.ttf";
$bbox=imagettfbbox($size,0,$fontfile,$text);
$image=imagecreatetruecolor(400,50);
$color=imagecolorallocate($image,220,220,0);
$x=(imagesx($image)-($bbox[2]-$bbox[0]))/2;
$y=imagesy($image)/2+$bbox[1];
imagettftext($image,$size,0,$x,$y,$color,$fontfile,$text);
header("Content-type: image/png");
imagepng($image);
```

Efekt:



## Wklejanie innych obrazów i kopiowanie fragmentów

W PHP nie ma obowiązku pracy tylko z jednym zasobem - obrazem w danym momencie. Możemy np. wczytać inny obraz z pliku i umieścić go, w całości lub w części na naszym obrazie. Do takiego umieszczania można użyć funkcji `imagecopy`. Ma ona parametry:

- `$dst_im` - docelowy zasób - obraz
- `$src_im` - źródłowy zasób - obraz
- `$dst_x`, `$dst_y` - punkt docelowy kopiowania, w tym miejscu znajdzie się lewy górny róg umieszczonego obrazu
- `$src_x`, `$src_y`, `$src_w`, `$src_h` - parametry obszaru prostokątnego (lewy górny róg, szerokość i wysokość) do skopiowania w ramach obrazu źródłowego

Przykład - umieszczamy obraz z URL na środku naszego obrazu:

```
$image=imagecreatetruecolor(300,200);
$color=imagecolorallocate($image,10,25,190);
imagefill($image,0,0,$color);
$image2=imagecreatefromjpeg("http://www.eti.pg.gda.pl/wydzial/
zdjecia2/zdjecia%202.jpg?rozmiar=200");
$dst_x=(imagesx($image)-imagesx($image2))/2;
$dst_y=(imagesy($image)-imagesy($image2))/2;
imagecopy($image,$image2,$dst_x,
$dst_y,0,0,imagesx($image2),imagesy($image2));
header("Content-type: image/png");
imagepng($image);
```

Efekt:



Inny przykład - tworzymy mozaikę z obrazka:

```
$image=imagecreatefromjpeg("http://www.eti.pg.gda.pl/wydzial/
zdjecia2/zdjecia%202.jpg?rozmiar=200");
/* obraz docelowy */
$mosaic=imagecreatetruecolor(imagesx($image),imagesy($image));
/* ilość rzędów mozaiki */
$rows=3;
/* ilość kolumn mozaiki */
$cols=4;
/* tablica numerów "kafelków" mozaiki - liczby od 0 do 11 */
$tiles=range(0,$rows*$cols-1);
/* mieszamy tablicę tak, aby kafelki były losowe */
shuffle($tiles);

/* rozmiary kafelka */
$tilew=imagesx($image)/$cols;
$tileh=imagesy($image)/$rows;
for ($i=0;$i<sizeof($tiles);$i++) {
    /* bierzemy kolejne numery kafelków z tablicy */
    /* i umieszczamy odpowiadające im części $image */
    /* w następujących po sobie fragmentach $mosaic */
    $src_x=($tiles[$i] % $cols) * $tilew;
    $src_y=floor($tiles[$i] / $cols) * $tileh;
    $dst_x=($i % $cols) * $tilew;
    $dst_y=floor($i / $cols) * $tileh;
    imagecopy(
        $mosaic,$image,$dst_x,$dst_y,$src_x,$src_y,$tilew,$tileh);
}
header("Content-type: image/png");
imagepng($mosaic);
```

Efekt:



Z obrazem, lub jego częścią nie musimy pracować tylko w skali 1:1. Istnieją funkcje pozwalające do obrazu docelowego skopiować obraz źródłowy z jednoczesną zmianą jego rozmiarów. Funkcje te to `imagecopyresized` oraz `imagecopyresampled`. W celu zachowania większego podobieństwa wyniku skalowania do zawartości obrazu źródłowego lepiej stosować drugą z wymienionych funkcji, gdyż wykonuje ona skalowanie wraz z tzw. "próbkowaniem", co pozwala na zachowanie proporcji zawartości obrazu zbliżonych do oryginału. Parametry obu z tych funkcji przypominają `imagecopy`, z dwoma dodatkowymi umieszczonymi "w środku":

- `$dst_im` - docelowy zasób - obraz
- `$src_im` - źródłowy zasób - obraz
- `$dst_x`, `$dst_y` - punkt docelowy kopiowania, w tym miejscu znajdzie się lewy górny róg umieszczonego obrazu
- `$src_x`, `$src_y` - lewy górny róg obszaru do skopiowania w ramach obrazu źródłowego
- `$dst_w`, `$dst_h` - szerokość i wysokość obrazu docelowego
- `$src_w`, `$src_h` - szerokość i wysokość obrazu źródłowego

Ostatnie cztery parametry tak naprawdę decydują o tym, jak skalowanie będzie wyglądać.

Przy skalowaniu obrazu często występuje wymóg zachowania proporcji. Oznacza to, że "z góry" ustalamy tylko jedną z dwóch wartości docelowych, wysokość lub szerokość, druga wyliczana jest z proporcji na podstawie rozmiarów obrazu źródłowego i wartości pierwszej. Przykład - określamy z góry szerokość:

```
$dst_w=300;  
$dst_h=floor($src_h*$dst_w/$src_w);
```

lub odwrotnie - określamy z góry wysokość:

```
$dst_h=150;  
$dst_w=floor($src_w*$dst_h/$src_h);
```