

Web je više od teksta. Slike se pojavljuju u obliku logotipa, gumba, fotografija, grafikona, oglasa i ikona. Mnoge od ovih slika su statičke, izrađene s pomoću alata poput Photoshopa i nikada se ne mijenjaju. No mnoge su dinamički izrađene – od oglasa za Amazonov referalni program koji sadrži vaše ime do grafikona burzovnih kretanja na stranicama Yahoo! Finance.

PHP podržava rad s grafikom kroz proširenja GD i Imlib2. U ovom poglavlju pokazat ćemo vam kako napraviti dinamičke slike s pomoću PHP-a i GD proširenja.

Dodavanje slike na stranicu

Uobičajena je zabluda da mješavina teksta i grafika proizlazi iz samo jednog HTTP zahtjeva. Jer u konačnici, kad pogledate stranicu vidite samo jednu stranicu koja sadrži ovakvu mješavinu. Važno je razumjeti da se uobičajena Web stranica koja sadrži tekst i slike stvara s pomoću niza HTTP zahtjeva koje upućuje preglednik Weba, a na koje odgovara Web poslužitelj. Svaki od odgovora može sadržavati jedan i samo jedan tip podataka, a svaka slika zahtijeva zasebni HTTP zahtjev i odgovor Web poslužitelja. Zato, ako vidite stranicu koja sadrži nešto teksta i dvije slike, znate da su bila potrebna tri HTTP zahtjeva i odgovora na njih kako bi se ta stranica prikazala.

Uzmite kao primjer ovu HTML stranicu:

```
<html>
  <head>
    <title>Example Page</title>
  </head>
  <body>
    This page contains two images.
    
    
  </body>
</html>
```

Niz zahtjeva poslanih Web poslužitelju za ovu stranicu izgledao bi otprilike ovako:

```
GET /page.html HTTP/1.0
GET /image1.jpg HTTP/1.0
GET /image2.jpg HTTP/1.0
```

Web poslužitelj šalje odgovor za svaki od ovih zahtjeva. U pripadajućim odgovorima zaglavlja s tipom sadržaja izgledaju ovako:

```
Content-Type: text/html
Content-Type: image/jpeg
Content-Type: image/jpeg
```

Za umetanje slike generirane s pomoću PHP-a na HTML stranicu, zamislite da je PHP zapis koji generira sliku ustvari sama slika. Prema tome, ako imamo skripte *image1.php* i *image2.php* koje stvaraju slike, možemo izmijeniti prethodni HTML tako da izgleda ovako (imena slika sada su PHP proširenja):

```
<html>
<head>
  <title>Example Page</title>
</head>
<body>
  This page contains two images.
  
  
</body>
</html>
```

Umjesto pozivanja stvarnih slika sa Web poslužitelja, *img* oznake sada se odnose na PHP skripte koje generiraju slike.

Osim toga, skriptama možete prosljediti varijable, pa ćete umjesto dvije skripte za generiranje dvije slike *img* oznake moći napisati ovako:

```


```

Zatim, unutar pozvane PHP datoteke *image.php* možete pristupiti do `$_GET['num']` kako biste generirali odgovarajuću sliku.

Proširenje GD

Prije no što započnete generiranje slika s pomoću PHP-a, morate prvo provjeriti imate li uopće mogućnosti za generiranje slika u PHP instalaciji. U ovom poglavlju razmatrat ćemo upotrebu GD proširenja, koje omogućava PHP-u korištenje open source GD grafičke biblioteke dostupne na adresi <http://www.boutell.com/gd/>. Od PHP-a 4.3, inačica GD-a (GD 2.0 ili novija) se podrazumijevano isporučuje uz PHP.

Učitajte poznatu stranicu `phpinfo()` i potražite odjeljak s naslovom „GD.“ Trebali biste vidjeti nešto slično sljedećem:

gd

GD Support	enabled
GD Version	2.0 or higher
FreeType Support	enabled
FreeType Linkage	with freetype
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

Obratite posebnu pažnju na navedene formate slika. To su formati koje ćete moći generirati. Dogodile su se tri veće prerade GD-a i pripadajućeg API-ja. Inačice prije GD-a 1.6 podržavaju samo GIF format. Inačica 1.6 i novije podržavaju JPEG, PNG i WBMP, ali ne podržavaju GIF (GIF format datoteke koristi patentirane algoritme za čije je korištenje potrebno platiti). Inačici GD-a 2.x dodano je nekoliko novih osnovnih funkcija za crtanje.

Sve GD 1.x inačice ograničene su na 8-bitne boje. To znači, slike koje stvarate ili obrađujete s pomoću GD-a 1.x mogu sadržavati samo 256 različitih boja. Za jednostavne tablice i grafikone to je više nego dovoljno, ali ako radite sa fotografijama ili drugim slikama koje imaju više od 256 boja, rezultati će biti manje nego zadovoljavajući. Kako biste dobili odgovarajuću podršku za boje, koristite nadogradnju GD 2.x ili umjesto nje koristite biblioteku *Imlib2* i odgovarajuće PHP proširenje. API za *Imlib2* proširenje je ponešto drugačiji od API-ja GD proširenja i nije obrađen u ovom poglavlju.

Osnovni grafički koncepti

Slika je pravokutnik koji se sastoji od piksela različitih boja. Boje su zadane svojim položajem u *paleti*, polju boja. Svaki unos na paleti sastoji se od tri različite vrijednosti boje – jedne za crvenu, jedne za zelenu i jedne za plavu. Svaka vrijednost varira od 0 (boja nije prisutna) do 255 (boja u punom intenzitetu).

Datoteke slika rijetko se sastoje od jednoznačno pohranjenih piksela i paleta. Umjesto toga, postoje različiti *formati datoteka* (GIF, JPEG, PNG itd.) koji pokušavaju malo sažeti podatke kako bi načinili manje datoteke.

Različiti formati datoteka različito obrađuju transparentnost slike, koja kontrolira da li se i na koji način pozadina vidi kroz sliku. Neki podržavaju i *alpha kanal*, dodatnu vrijednost za svaki piksel koja zadaje transparentnost u toj točki. Drugi jednostavno koriste jedan unos na paleti za zadavanje transparentnosti.

Umekšavanje rubova (engl. *antialiasing*) je tehnika pomicanja ili bojanja piksela tako da čine postepeni prijelaz između oblika i njegove pozadine. Ona ublažava grube i nazupčane rubove koji mogu pokvariti sliku. Neke funkcije koje se primjenjuju na slikama implementiraju umekšavanje rubova.

S 256 mogućih vrijednosti za svaku od boja postoji ukupno 16,777,216 mogućih boja za svaki piksel. Neki formati datoteka ograničavaju broj boja koje mogu postojati na paleti (npr. GIF podržava ne više od 256 boja); drugi dozvoljavaju upotrebu koliko god boja trebate. Oni su poznati kao formati *prirodnih boja*, jer 24-bitna boja (po 8 bitova za crvenu, zelenu i plavu) daje više nijansi nego što ih ljudsko oko može razlikovati.

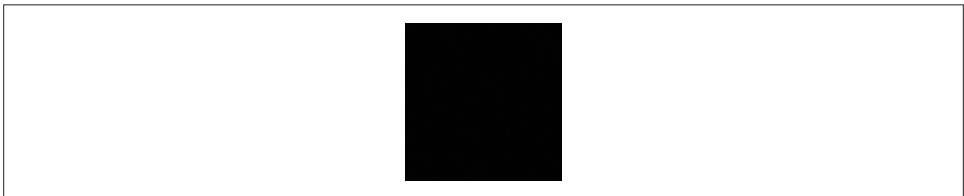
Izrada slika

Za sada počnimo s najjednostavnijim mogućim GD primjerom. Primjer 9-1 je skripta koja generira crnom bojom ispunjen četverokut. Kôd radi s bilo kojom inačicom GD-a koja podržava PNG format slika.

Primjer 9-1. Crni četverokut na bijeloj pozadini (black.php)

```
<?php
$im = ImageCreate(200,200);
$white = ImageColorAllocate($im,0xFF,0xFF,0xFF);
$black = ImageColorAllocate($im,0x00,0x00,0x00);
ImageFilledRectangle($im,50,50,150,150,$black);
header('Content-Type: image/png');
ImagePNG($im);
?>
```

Primjer 9-1 ilustrira osnovne korake u generiranju bilo koje slike: stvaranje slike, alociranje boja, crtanje oblika i zatim spremanje ili slanje slike. Slika 9-1 prikazuje rezultat primjera 9-1.



Slika 9-1. Crni četverokut na bijeloj pozadini

Kako biste vidjeli rezultat, jednostavno usmjerite preglednik na PHP stranicu *black.php*. Za postavljanje ove slike na Web stranicu, koristite:

```

```

Struktura grafičkih programa

Većina programa za generiranje dinamičkih slika slijedi iste osnovne korake koji su istaknuti u primjeru 9-1.

Možete izraditi sliku u 256 boja s pomoću funkcije `ImageCreate()`, koja vraća kao odgovor identifikator slike:

```
$image = ImageCreate($sirina, $visina);
```

Sve boje koje se koriste na nekoj slici moraju biti alocirane s pomoću funkcije `ImageColorAllocate()`. Prva alocirana boja postaje boja pozadine za sliku.*

```
$color = ImageColorAllocate($slika, $crvena, $zelena, $plava);
```

Argumenti su vrijednosti komponenata boje. U primjeru 9-1 napisali smo vrijednosti boja u heksadecimalnom obliku, kako bismo poziv funkcije približili HTML načinu predstavljanja boja "#FFFFFF" i „#000000“.

U GD-u postoji mnogo osnovnih funkcija za crtanje. Primjer 9-1 koristi `ImageFilledRectangle()` u kojoj zadajete dimenzije pravokutnika navođenjem koordinata gornjeg lijevog i donjeg desnog kuta:

```
ImageFilledRectangle($slika, $glx, $gly, $ddx, $ddy, $boja);
```

Sljedeći korak je slanje Content-Type zaglavlja pregledniku s odgovarajućim tipom sadržaja za format slike koja se pravi. Kad je to učinjeno, pozvat ćemo odgovarajuću izlaznu funkciju. Funkcije `ImageJPEG()`, `ImagePNG()` i `ImageWBMP()` stvaraju datoteke za slike u formatu JPEG, PNG i WBMP:

```
ImageJPEG($image [, $ime_datoteke [, $kvaliteta ]]);  
ImagePNG($image [, $ime_datoteke ]);  
ImageWBMP($image [, $ime_datoteke ]);
```

Ukoliko nije navedeno *ime_datoteke*, slika se šalje pregledniku. Argument *kvaliteta* za JPEG datoteke je broj od 0 (za najmanju kvalitetu) do 10 (za najveću kvalitetu). Što je kvaliteta lošija, JPEG datoteka je manja. Pretpostavljena vrijednost je 7.5.

U primjeru 9-1 postavljamo HTTP zaglavlje odmah nakon pozivanja funkcije koja generira izlaz, `ImagePNG()`. Ako zadate tip sadržaja na samom početku skripte, sve eventualne pogreške koje se dogode tretiraju se kao podaci koji pripadaju slici i preglednik prikazuje ikonu slomljene slike. Tablica 9-1 sadrži popis formata slika i njihove tipove sadržaja.

Tablica 9-1. Tipovi sadržaja za različite formate slika

Format	Tip sadržaja
GIF	image/gif
JPEG	image/jpeg
PNG	image/png
WBMP	image/vnd.wap.wbmp

* Ovo je točno za slike sa paletom boja. Slike sa prirodnom paletom boja izrađene sa `ImageCreateTrueColor()` ne podliježu ovom pravilu.

Promjena izlaznog formata

Kao što ste mogli zaključiti, generiranje slike drugog formata zahtijeva samo dvije promjene u skripti: navođenje drugačijeg tipa sadržaja i upotreba druge funkcije za generiranje slika. Primjer 9-2 je primjer 9-1 promijenjen tako da generira JPEG sliku umjesto PNG slike.

Primjer 9-2. JPEG inačica crnog kvadrata

```
<?php
$im = ImageCreate(200,200);
$white = ImageColorAllocate($im,0xFF,0xFF,0xFF);
$black = ImageColorAllocate($im,0x00,0x00,0x00);
ImageFilledRectangle($im,50,50,150,150,$black);
header('Content-Type: image/jpeg');
ImageJPEG($im);
?>
```

Utvrđivanje podržanih formata slika

Ukoliko pišete kôd koji mora biti prenosiv na sustave koji možda podržavaju neke druge formate slika, upotrijebite funkciju `ImageTypes()` da biste provjerili koji tipovi slika su podržani. Ova funkcija vraća polje bitova. Možete koristiti operator nad bitovima `AND (&)` da biste provjerili je li odgovarajući bit postavljen. Konstante `IMG_GIF`, `IMG_JPG`, `IMG_PNG` i `IMG_WBMP` odgovaraju bitovima za te formate slika.

Primjer 9-3 generira PNG datoteke ako je PNG podržan, JPEG datoteke ako PNG nije podržan, te GIF datoteke ukoliko ni PNG niti JPEG nisu podržani.

Primjer 9-3. Provjera podrške za format slike

```
<?php
$im = ImageCreate(200,200);
$white = ImageColorAllocate($im,0xFF,0xFF,0xFF);
$black = ImageColorAllocate($im,0x00,0x00,0x00);
ImageFilledRectangle($im,50,50,150,150,$black);
if (ImageTypes() & IMG_PNG) {
    header("Content-Type: image/png");
    ImagePNG($im);
} elseif (ImageTypes() & IMG_JPG) {
    header("Content-Type: image/jpeg");
    ImageJPEG($im);
} elseif (ImageTypes() & IMG_GIF) {
    header("Content-Type: image/gif");
    ImageGIF($im);
}
?>
```

Čitanje postojeće datoteke

Ako želite početi s nekom postojećom slikom i zatim je promijeniti, upotrijebite funkciju `ImageCreateFromDPEG()` ili `ImageCreateFromPNG()`:

```
$image = ImageCreateFromJPEG(ime_datoteke);  
$image = ImageCreateFromPNG(ime_datoteke);
```

Osnovne funkcije za crtanje

GD sadrži funkcije za crtanje osnovnih točaka, linija, lukova, pravokutnika i poligona. Ovaj odlomak opisuje osnovne funkcije podržane u GD 2.x.

Osnovna funkcija je `ImageSetPixel()` koja zadaje boju piksela:

```
ImageSetPixel(slika, x, y, boja);
```

Postoje dvije funkcije za crtanje linija, `ImageLine()` i `ImageDashedLine()`:

```
ImageLine(slika, pocetak_x, pocetak_y, kraj_x, kraj_y, boja);  
ImageDashedLine(slika, pocetak_x, pocetak_y, kraj_x, kraj_y, boja);
```

Postoje dvije funkcije za crtanje pravokutnika, jedna koja jednostavno crta konture lika i jedna koja popunjava pravokutnik određenom bojom:

```
ImageRectangle(slika, glx, gly, ddx, ddy, boja);  
ImageFilledRectangle(slika, glx, gly, ddx, ddy, boja);
```

Zadajte položaj i veličinu pravokutnika tako što ćete zadati koordinate za gornji lijevi i donji desni ugao. Možete proizvoljno nacrtati poligone s pomoću funkcija `ImagePolygon()` i `ImageFilledPolygon()`:

```
ImagePolygon(slika, točke, broj, boja);  
ImageFilledPolygon(slika, točke, broj, boja);
```

Objekti funkcije uzimaju polje *točaka*. Ovo polje ima dvije cjelobrojne vrijednosti (koordinate *x* i *y*) za svaki vrh poligona. Argument *broj* je broj vrhova u polju.

Funkcija `ImageArc()` crta luk (odsječak elipse):

```
ImageArc(slika, x_središta, y_središta, širina, visina, pocetak, kraj, boja);
```

Elipsa je definirana svojim središtem, širinom i visinom (visina i širina su jednake za krug). Početna i završna točka luka date su kao stupnjevi koji se broje suprotno od smjera kazaljke na satu počevši od 3 sata. Cijelu elipsu crtate s pomoću vrijednosti početak postavljene na 0 i kraj na 360.

Postoje dva načina za popunjavanje nacrtanih oblika. Funkcija `ImageFill()` izvodi ujednačanu popunu, mijenjajući boju piksela počevši od zadanog položaja. Bilo kakva promjena u boji piksela označava granicu popune. Funkcija `ImageFillToBorder()` omogućava da prosljedite određenu boju za granice popune:

```
ImageFill(slika, x, y, boja);  
ImageFillToBorder(slika, x, y, boja_ruba, boja);
```

Još jedna stvar koju ćete možda željeti učiniti sa slikama je njihova rotacija. To će možda biti od pomoći ukoliko pokušavate izraditi brošuru u Web stilu. Funkcija s pomoću koje se to može postići zove se `imagerotate()` a njezina sintaksa je sljedeća:

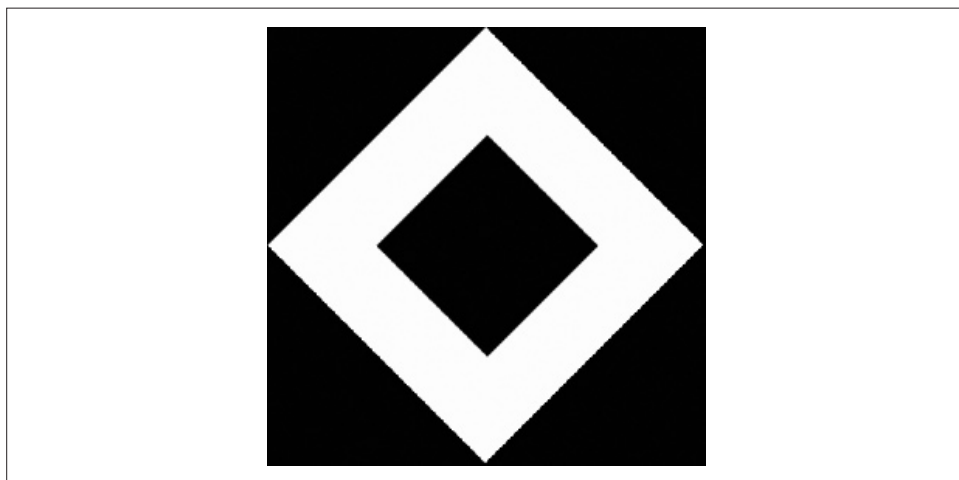
```
Imagerotate(slika, kut, boja pozadine)
```

Kôd u primjeru 9-4 prikazuje crni kvadrat koji smo vidjeli ranije, ali rotiran za 45 stupnjeva korištenjem ove funkcije. Opcija boje pozadine, korištena za zadavanje boje nepokrivenog područja nakon rotacije lika, postavljena je na 1 kako bi se prikazao kontrast između crne i bijele boje. Obavezno se poigrajte s ovom funkcijom da biste vidjeli rezultate prije nego što bilo koji dio koda za obradu ove slike stavite u upotrebu.

Slika 9-2 prikazuje rezultat.

Primjer 9-4. Primjer rotacije slike

```
<?php
$im = ImageCreate(200,200);
$white = ImageColorAllocate($im,0xFF,0xFF,0xFF);
$black = ImageColorAllocate($im,0x00,0x00,0x00);
ImageFilledRectangle($im,50,50,150,150,$black);
header('Content-Type: image/png');
$im_rotated = imagerotate($im, 45, 1);
ImagePNG($im_rotated);
?>
```



Slika 9-2. Crni kvadrat rotiran za 45 stupnjeva

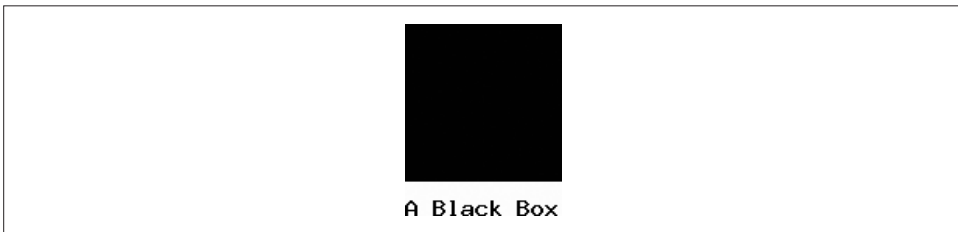
Slike s tekstom

Često je uz slike potrebno dodati i tekst. GD za ovu namjenu ima ugrađena pisma. Primjer 9-5 dodaje tekst na sliku crnog kvadrata.

Primjer 9-5. Dodavanje teksta na sliku

```
<?php
$image = ImageCreate(200,200);
$white = ImageColorAllocate($im,0xFF,0xFF,0xFF);
$black = ImageColorAllocate($im,0x00,0x00,0x00);
ImageFilledRectangle($im,50,50,150,150,$black);
ImageString($im,5,50,160,"A Black Box",$black);
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

Slika 9-3 prikazuje rezultat primjera 9-5.



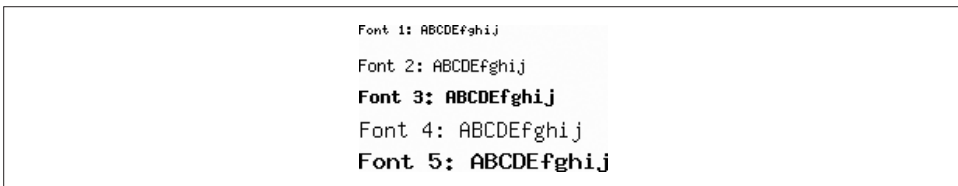
Slika 9-3. Slika crnog kvadrata s dodanim tekstom

Funkcija `ImageString()` dodaje tekst na sliku. Zadaite gornju lijevu točku teksta, kao i boju i pismo koje želite koristiti:

```
ImageString(slika, pismo, x, y, tekst, boja);
```

Pisma

Pisma u GD-u određena su brojevima. Pet ugrađenih pisama prikazano je na slici 9-4.



Slika 9-4. Izvorna GD pisma

Slijedi kôd koji će ilustrirati pisma:

```
<?php
$image = ImageCreate(200,200);
$black = ImageColorAllocate($im,0x00,0x00,0x00);

ImageString($im,1,10,10,"Font 1: ABCDEfghij",$black);
ImageString($im,2,10,30,"Font 2: ABCDEfghij",$black);
ImageString($im,3,10,50,"Font 3: ABCDEfghij",$black);
ImageString($im,4,10,70,"Font 4: ABCDEfghij",$black);
```

```
ImageString($im,5,10,90,"Font 5: ABCDEfghij",$black);
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

Možete izraditi i vlastita pisma te ih učitati u GD upotrebom funkcije `LoadFont()`. Ipak, ova pisma su binarna i ovise o arhitekturi sustava.

Korištenje TrueType pisama s pomoću TrueType funkcija u GD-u pruža više fleksibilnosti.

TrueType pisma

Za korištenje TrueType pisama s GD-om, PHP mora biti preveden s TrueType podrškom preko FreeType biblioteke. Provjerite stranicu `phpinfo()` (kao što je opisano ranije u ovom poglavlju) da biste vidjeli je li u odjeljku „GD“ zadan unos koji uključuje biblioteku FreeType.

Da biste na sliku dodali tekst u TrueType pismu pozovite `ImageTTFText()`:

```
ImageTTFText(slika, veličina, kut, x, y, boja, pismo, tekst);
```

Veličina se zadaje u pikselima. Kut je u stupnjevima, počevši od položaja 3 sata (0 daje vodoravni tekst, 90 daje okomiti tekst koji ide uz sliku, itd.). Koordinate `x` i `y` određuju donji lijevi kut teksta (za razliku od `ImageString()`, gdje koordinate zadaju gornji desni kut). Tekst može uključivati UTF-8* sekvence oblika `ê`; za ispis ASCII znakova.

U GD-u 1.x, `pismo` je ime datoteke s točnom putanjom uključujući i `.ttf` nastavak imena datoteke. U GD-u 2.x pisma se podrazumijevano nalaze u `/usr/share/fonts/truetype` a nastavak imena `.ttf` se automatski dodaje. Zadavanje veličine pisma također se ponešto razlikuje između GD 1.x i GD 2.x.

Podrazumijevano, rubovi teksta u TrueType pismu se umekšavaju. Ovo mnoga pisma čini jasnijim za čitanje, iako su malo zamućena. Izbjegavanje umekšavanja može otežati čitanje sitnog teksta.

Izbjegavanje umekšavanja možete isključiti zadavanjem negativnog indeksa boje (npr. -4 znači da je u upotrebi indeks boje vrijednosti 4, ali da u tekstu nije potrebno umekšavanje rubova).

U primjeru 9-6 korišteno je TrueType pismo za dodavanje teksta na sliku.

Primjer 9-6. Upotreba TrueType pisma

```
<?php
$im = ImageCreate(350, 70);
$white = ImageColorAllocate($im, 0xFF,0xFF,0xFF);
```

* UTF-8 je 8 bitna Unicode shema za šifriranje. Više o Unicodeu pronaći ćete na adresi <http://www.unicode.org>.

Primjer 9-6. Upotreba TrueType pisma (nastavak)

```
$black = ImageColorAllocate($im, 0x00,0x00,0x00);  
ImageTTFText ($im, 20, 0, 10, 40, $black, 'courbi', 'The Courier TTF font');  
header('Content-Type: image/png');  
ImagePNG($im);  
?>
```

Slika 9-5 Prikazuje rezultat primjera 9-6



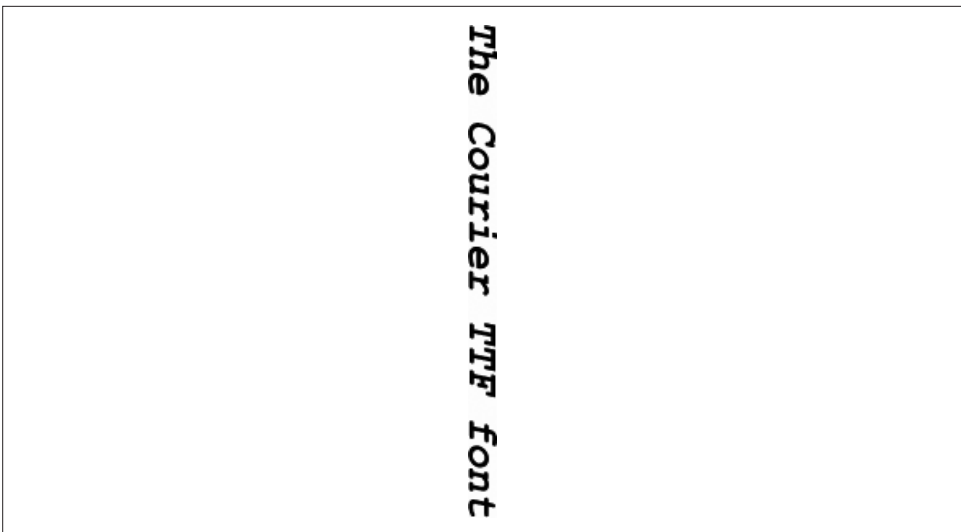
Slika 9-5. TrueType pismo Courier Bold Italic

Primjer 9-7 koristi ImageTTFText() za dodavanje okomitog teksta na sliku.

Primjer 9-7. Prikaz okomitog TrueType teksta

```
<?php  
$im = ImageCreate(70, 350);  
$white = ImageColorAllocate ($im, 255, 255, 255);  
$black = ImageColorAllocate ($im, 0, 0, 0);  
ImageTTFText ($im, 20, 270, 28, 10, $black, 'courbi', 'The Courier TTF font');  
header('Content-Type: image/png');  
ImagePNG($im);  
?>
```

Slika 9-6 prikazuje rezultat primjera 9-7.



Slika 9-6. Okomiti TrueType tekst

Dinamički generirani gumbi

Popularna primjena dinamički generiranih slika je u izradi sličica za gumb s prijelazom (ovo je također objašnjeno u poglavlju 1). Obično se koristi prazna slika pozadine gumba, a tekst se zatim dodaje preko nje, kao što je prikazano u primjeru 9-8.

Primjer 9-8. Izrada dinamičkog gumba

```
<?php
$font = 'times';
if (!$size) $size = 12;
$im = ImageCreateFromPNG('button.png');
// izračunava položaj teksta
$tsize = ImageTTFBBox($size,0,$font,$text);
$dx = abs($tsize[2]-$tsize[0]);
$dy = abs($tsize[5]-$tsize[3]);
$x = ( ImageSx($im) - $dx ) / 2;
$y = ( ImageSy($im) - $dy ) / 2 + $dy;
// ispisuje tekst
$black = ImageColorAllocate($im,0,0,0);
ImageTTFText($im, $size, 0, $x, $y, $black, $font, $text);
header('Content-Type: image/png');
ImagePNG($im);
?>
```

U ovom slučaju, prazan gumb (*button.png*) izgleda kao na slici 9-7.



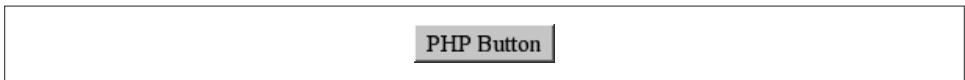
Slika 9-7. Prazan gumb

Skripta iz primjera 9-8 može biti pozvana sa stranice kao što je ova:

```

```

Ovaj HTML kôd generira gumb prikazan na slici 9-8.



Slika 9-8. Gumb s generiranim tekstualnim natpisom

Znak + u URL-u je šifrirana oznaka za prazan prostor. Prazni prostori nisu dozvoljeni u URL adresama pa moraju biti šifrirani. Koristite PHP-ovu funkciju `urlencode()` za šifriranje nizova znakova za gumb. Na primjer:

```

```

Spremanje dinamički generiranih gumba u keš memoriju

Generirati sliku je nešto sporije od slanja statičke slike. Za gumbе koji se ne mijenjaju može se implementirati jednostavan mehanizam pohranjivanja u keš memoriju.

Primjer 9-9 stvara gumb samo kada u keš memoriji nije pronađena datoteka za taj gumb. Varijabla `$path` sadrži mapu u koju Web poslužitelj može pisati i u koju mogu biti pohranjeni gumbi. Funkcija `filesize()` vraća veličinu datoteke, a `readfile()` šalje sadržaj datoteke pregledniku. Ova skripta nije sigurna jer koristi tekstualni parametar sa imenom datoteke (Poglavlje 12 objašnjava zbog čega i kako ovo popraviti).

Primjer 9-9. Spremanje dinamičkih gumba u keš memoriju

```
<?php
header('Content-Type: image/png');
$path = "/tmp/buttons";           // mapa za čuvanje gumba
$text = $_GET['text'];

if($bytes = @filesize("$path/$text.png")) { // šalje gumb iz keš memorije
    header("Content-Length: $bytes");
    readfile("$path/$text.png");
} else {                               // pravi, sprema i šalje novi gumb
    $font = 'times';
    if (!$_GET['size']) $_GET['size'] = 12;
    $im = ImageCreateFromPNG('button.png');
    $tsize = ImageTTFBBox($size, 0, $font, $text);
    $dx = abs($tsize[2]-$tsize[0]);     // centrira tekst
    $dy = abs($tsize[5]-$tsize[3]);
    $x = ( imagesx($im) - $dx ) / 2;
    $y = ( imagesy($im) - $dy ) / 2 + $dy;
    $black = ImageColorAllocate($im,0,0,0);
    ImageTTFText($im, $_GET['size'], 0, $x, $y, -$black, $font, $text);
    ImagePNG($im);                    // šalje sliku pregledniku
    ImagePNG($im,"$path/$text.png");   // sprema sliku u datoteku
}
?>
```

Još brža isporuka dinamičkih gumba

Primjer 9-9 još uvijek nije brz koliko bi mogao biti. Postoji naprednija tehnika spremanja u keš memoriju koja u potpunosti uklanja PHP iz upita nakon što je slika generirana.

Prvo izradite mapu `buttons` za gumbе negdje pod `DocumentRoot` Web poslužitelja i provjerite ima li Web poslužitelj dozvolu za pisanje u nju. Na primjer, ako je `DocumentRoot` mapa `/var/www/html`, izradite mapu `var/www/html/buttons`.

Drugo, u datoteku *httpd.conf* Apache poslužitelja dodajte sljedeći blok:

```
<Location /buttons/>
    ErrorDocument 404 /button.php
</Location>
```

To govori Apache poslužitelju da bi upit za datotekama koje ne postoje u mapi *buttons* trebao proslijediti skripti *button.php*.

Treće, spremite primjer 9-10 kao *button.php*. Ova skripta stvara nove gumbe, sprema ih u keš memoriju i šalje pregledniku. Ipak, u nekoliko stvari se razlikuje od primjera 9-9. Nema parametre obrasca u `$_GET` jer Apache stranice s pogreškama obrađuje tako što ih preusmjerava. Umjesto toga, moramo razdvojiti vrijednosti u `$_SERVER` kako bismo otkrili koji gumb generiramo. Kad smo već ovdje, brišemo `..` u imenu datoteke da bismo popravili sigurnosni nedostatak iz primjera 9-9.

Kada je *button.php* instalirana i stigne zahtjev poput *http://web.stranica.com/buttons/php.png*, Web poslužitelj provjerava postoji li datoteka *buttons/php.png*. Ako ne postoji, upit se preusmjerava skripti *button.php* koja stvara sličicu s tekстом „php“ i sprema je u *buttons/php.png*. Svaki sljedeći zahtjev za ovom datotekom opslužuje se iz keš memorije, bez izvođenja PHP skripte.

Primjer 9-10. Brža isporuka dinamičkih gumba

```
<?php
// pruža parametre preusmjeravanja, ako postoje
parse_str($_SERVER['REDIRECT_QUERY_STRING']);

$button_dir = '/buttons/';
$url = $_SERVER['REDIRECT_URL'];
$root = $_SERVER['DOCUMENT_ROOT'];

// odabir anstavka imena
$ext = substr($url, strrpos($url, '.'));

// briše mapu i nastavak imena iz $url
$file = substr($url, strlen($button_dir), -strlen($ext));

// ne dopustite '..' u imenu datoteke zbog sigurnosti sustava
$file = str_replace('..', '', $file);

// tekst gumba
$text = urldecode($file);

// stvara sliku
if(!isset($font)) $font = 'times';
if(!isset($size)) $size = 12;
$img = ImageCreateFromPNG('button.png');
$tsize = ImageTTFBBox($size, 0, $font, $text);
$dx = abs($tsize[2] - $tsize[0]);
$dy = abs($tsize[5] - $tsize[3]);
```

Primjer 9-10. Brža isporuka dinamičkih gumba (nastavak)

```
$x = ( ImageSx($im) - $dx ) / 2;  
$y = ( ImageSy($im) - $dy ) / 2 + $dy;  
$black = ImageColorAllocate($im,0,0,0);  
ImageTTFText($im, $size, 0, $x, $y, -1*$black, $font, $text);  
// šalje i sprema sliku  
header('Content-Type: image/png');  
ImagePNG($im);  
ImagePNG($im,$root.$button_dir."$file.png");  
ImageDestroy($im);  
?>
```

Jedini nedostatak mehanizma iz primjera 9-10 je što tekst gumba ne može sadržavati znakove koji nisu dozvoljeni u imenu datoteke. Ipak, ovo je najučinkovitiji način za pohranjivanje dinamički generiranih slika. Ako mijenjate izgled gumba i trebate osvježiti slike pohranjene u keš memoriji, jednostavno obrišite sve slike u mapi *buttons* i one će se iznova izraditi.

Možete otići i korak dalje pa omogućiti da skripta *button.php* podržava razne tipove slika. Jednostavno provjerite \$ext i pozovite odgovarajuću funkciju ImagePNG(), ImageJPEG() ili ImageGIF() na kraju skripte. Možete parsirati ime datoteke i dodati modifikatore kao što su boja, veličina i pismo ili ih proslijediti izravno u URL adresi. Zbog poziva parse_str() u primjeru, URL kao što je <http://web.stranica.com/buttons/hp.png?size=16> ispisuje „php“ u pismu veličine 16 točaka.

Promjena veličine slike

Postoje dva načina za promjenu veličine slike. Funkcija ImageCopyResized() je dostupna u svim inačicama GD-a, ali algoritam za promjenu veličine nije doraden i može uzrokovati nazupčane rubove. Funkcija ImageCopyResampled() je nova u GD-u 2.x i omogućava interpolaciju piksela kako bi slike čija je veličina promijenjena bile jasnije i glatkih rubova. Ona je sporija od funkcije ImageCopyResized(). Obje funkcije koriste iste argumente:

```
ImageCopyResized(dest, src, dx, dy, sx, sy, dw, dh, sw, sh);  
ImageCopyResampled(dest, src, dx, dy, sx, sy, dw, dh, sw, sh);
```

Parametri dest i src su identifikatori slike. Točka (dx, dy) je točka na odredišnoj slici kamo će područje biti kopirano. Točka (sx, sy) je gornji lijevi ugao izvorne slike. Parametri sw, sh, dw i dh zadaju širinu i visinu područja za kopiranje na izvornoj i odredišnoj slici.

Primjer 9-11 sliku *php.jpg* prikazanu na slici 9-9 smanjuje je na jednu četvrtinu prvobitne veličine. Rezultat te operacije prikazan je na slici 9-10.

Primjer 9-11. Promjena veličine s pomoću funkcije `ImageCopyResampled()`

```
<?php
$src = ImageCreateFromJPEG('php.jpg');
$width = ImageSx($src);
$height = ImageSy($src);
$x = $width/2; $y = $height/2;
$dst = ImageCreateTrueColor($x,$y);
ImageCopyResampled($dst,$src,0,0,0,0,$x,$y,$width,$height);
header('Content-Type: image/png');
ImagePNG($dst);
?>
```



Slika 9-9. Originalna slika `php.jpg`

Rezultat skripte iz primjera 9-11 prikazan je na slici 9-10.



Slika 9-10. Slika smanjena na $\frac{1}{4}$ izvorne veličine

Podjelom visine i širine s 4 umjesto sa 2 dobit ćemo rezultat prikazan na slici 9-11.



Slika 9-11. Slika smanjena na $\frac{1}{16}$ izvorne veličine

Rad s bojama

Podrška za boje u GD 2.x značajno se poboljšala u odnosu na GD 1.x. U GD-u 1.x nije postojalo rješenje za alpha kanal, rad s bojama bio je prilično jednostavan, a biblioteka je podržavala slike sa 8-bitnom paletom (256 boja). Kad u GD-u 1.x stvarate slike sa 8-bitnom paletom, koristite funkciju `ImageCreate()`, a prva boja koju alocirate upotrebom funkcije `ImageColor Allocate ()` postaje boja pozadine.

U GD-u 2.x postoji podrška za slike u prirodnim bojama zaledno s alpha kanalom. GD 2.x ima 7-bitni (0-127) alpha kanal.

Za izradu slike u prirodnim bojama koristite funkciju `ImageCreateTrueColor()`:

```
$image = ImageCreateTrueColor($sirina, $visina);
```

Funkciju `ImageColorResolveAlpha()` koristite za izradu indeksa boja koji uključuje i transparentnost:

```
$color = ImageColorResolveAlpha($slika, $crvena, $zelena, $plava, $alpha);
```

Alpha vrijednost je između 0 (neprozirno) i 127 (prozirno).

Dok je većina ljudi naviknuta na 8-bitni (0-255) alpha kanal, ustvari je sasvim praktično da je GD 7-bitni (0-127). Svaki piksel predstavljen je s 32-bitnim cijelim brojem, sa četiri 8-bitna bajta razmještena na sljedeći način:

Visoki bajt			Niski bajt
{Alpha kanal}	{crvena}	{zelena}	{plava}

Za cijeli broj s predznakom, krajnji lijevi bit, ili najviši bit, koristi se za označavanje je li vrijednost negativna, što ostavlja samo 31 bit za stvarne informacije. Podrazumijevana cjelobrojna vrijednost u PHP-u je long s predznakom u koju možete pohraniti jedan unos GD paleta. Da li je cjelobrojna vrijednost pozitivna ili negativna govori nam je li umekšavanje rubova uključeno za taj unos paleta.

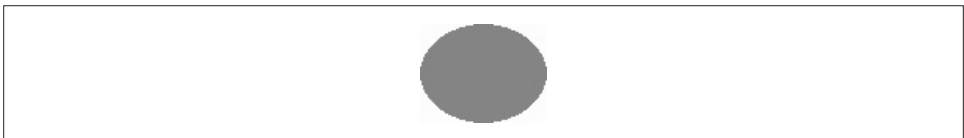
Za razliku od slika s određenom paletom boja, kod slika s prirodnom paletom boja i GD 2.x prva boja koju alocirate ne postaje automatski i boja pozadine. Pozovite `ImageFilledRectangle()` da biste popunili pozadinu slike bojom po želji.

Primjer 9-12 daje sliku s prirodnom paletom boja i crta poluprozirnu narandžastu elipsu na bijeloj pozadini.

Primjer 9-12. Narandžasta elipsa na bijeloj pozadini

```
<?php
$image = ImageCreateTrueColor(150,150);
$white = ImageColorAllocate($im,255,255,255);
ImageAlphaBlending($im, false);
ImageFilledRectangle($im,0,0,150,150,$white);
$red = ImageColorResolveAlpha($im,255,50,0,50);
ImageFilledEllipse($im,75,75,80,63,$red);
header('Content-Type: image/png');
ImagePNG($im);
?>
```

Slika 9-12 prikazuje rezultat primjera 9-12.



Slika 9-12. Narandžasta elipsa na bijeloj pozadini

Možete koristiti funkciju `ImageTrueColorToPalette()` za pretvaranje slike s paletom prirodnih boja u sliku s indeksom boja.

Primjena alpha kanala

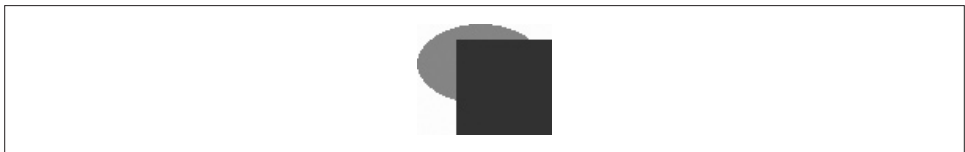
U primjeru 9-12, isključili smo alpha miješanje prije crtanja pozadine i elipse. Alpha miješanje je prekidač koji zadaje hoće li alpha kanal, ako je prisutan, biti primijenjen prilikom crtanja. Ukoliko je alpha miješanje isključeno, stari piksel zamjenjuje se novi pikselom. Ako alpha kanal postoji za novi piksel, on se primjenjuje, ali se gube sve informacije o pikselu preko kojeg je pisano a vezane su uz prvobitni piksel.

Primjer 9-13 ilustrira alpha miješanje na primjeru sivog pravokutnika s 50-postotnom prozirnošću preko narandžaste elipse.

Primjer 9-13. Sivi pravokutnik s 50-postotnim alpha kanalom

```
<?php
$im = ImageCreateTrueColor(150,150);
$white = ImageColorAllocate($im,255,255,255);
ImageAlphaBlending($im, false);
ImageFilledRectangle($im,0,0,150,150,$white);
$red = ImageColorResolveAlpha($im,255,50,0,63);
ImageFilledEllipse($im,75,75,80,50,$red);
$gray = ImageColorResolveAlpha($im,70,70,70,63);
ImageAlphaBlending($im, false);
ImageFilledRectangle($im,60,60,120,120,$gray);
header('Content-Type: image/png');
ImagePNG($im);
?>
```

Slika 9-13 prikazuje rezultat primjera 9-13 (alpha miješanje je još uvijek isključeno).



Slika 9-13. Sivi pravokutnik iznad narandžaste elipse

Ako promijenimo primjer 9-13 tako da uključimo alpha miješanje neposredno prije pozivanja `Image-FilledRectangle()`, dobit ćete rezultat prikazan na slici 9-14.



Slika 9-14. Slika s uključenim alpha miješanjem

Prepoznavanje boja

Za provjeru indeksa boje za određeni piksel na slici koristite funkciju `ImageColorAt()`:

```
$color = ImageColorAt($slika, $x, $y);
```

Za slike s 8-bitnom paletom boja funkcija vraća indeks boje koji zatim prosljeđuje funkciji `ImageColorsForIndex()` kako biste dobili stvarne RGB vrijednosti:

```
$values = ImageColorsForIndex($slika, $indeks);
```

Polje koje `ImageColorsForIndex()` vraća ima ključeve „red“, „green“ i „blue“. Ako pozovete `ImageColorsForIndex()` na boji sa slike sa paletom prirodnih boja, vraćeno polje ima dodatni ključ „alpha“.

Indeksi prirodnih boja

Indeks boje koji vraća `ImageColorResolveAlpha()` je ustvari 32-bitni long s predznakom čija prva tri bajta sadrže vrijednosti za crvenu, zelenu i plavu komponentu. Sljedeći bit označava je li za tu boju uključeno umekšavanje rubova, a ostalih sedam bitova sadrže vrijednost transparentnosti.

Na primjer:

```
$green = ImageColorResolveAlpha($im, 0, 0, 255, 127);
```

Ovaj kod postavlja `$green` na 2130771712, što je heksadecimalno `0x7F00FF00` a binarno `01111111000000001111111100000000`.

To je ekvivalentno sljedećem pozivu `ImageColorResolveAlpha()`:

```
$green = 127<<24 | 0<<16 | 255<<8 | 0;
```

Možete ispustiti dva unosa 0 u ovom primjeru tako da glasi:

```
$green = 127<<24 | 255<<8;
```

Za dekonstrukciju ove vrijednosti možete koristiti nešto poput ovoga:

```
$a = ($col & 0x7F000000) >> 24;  
$r = ($col & 0x00FF0000) >> 16;  
$g = ($col & 0x0000FF00) >> 8;  
$b = ($col & 0x000000FF);
```

Izravno upravljanje vrijednostima palete prirodnih boja je rijetko potrebno. Jedna primjena je za generiranje slike s pomoću koje se testiraju boje, a koja prikazuje čiste tonove crvene, zelene i plave. Na primjer:

```
$im = ImageCreateTrueColor(256,60);
for($x=0; $x<256; $x++) {
    ImageLine($im, $x, 0, $x, 19, $x);
    ImageLine($im, 255-$x, 20, 255-$x, 39, $x<<8);
    ImageLine($im, $x, 40, $x, 59, $x<<16);
}
ImagePNG($im);
```

Slika 9-15 prikazuje rezultat programa za testiranje boja.



Slika 9-15. Test boja

Očito je da će rezultat biti daleko živopisniji nego što vam možemo prikazati u ovoj crnobijeloj knjizi, pa iskušajte ovaj primjer sami. U ovom primjeru daleko je lakše jednostavno izračunati boju piksela nego pozivati `ImageColorResolveAlpha()` za svaku boju.

Tekstualni prikaz slike

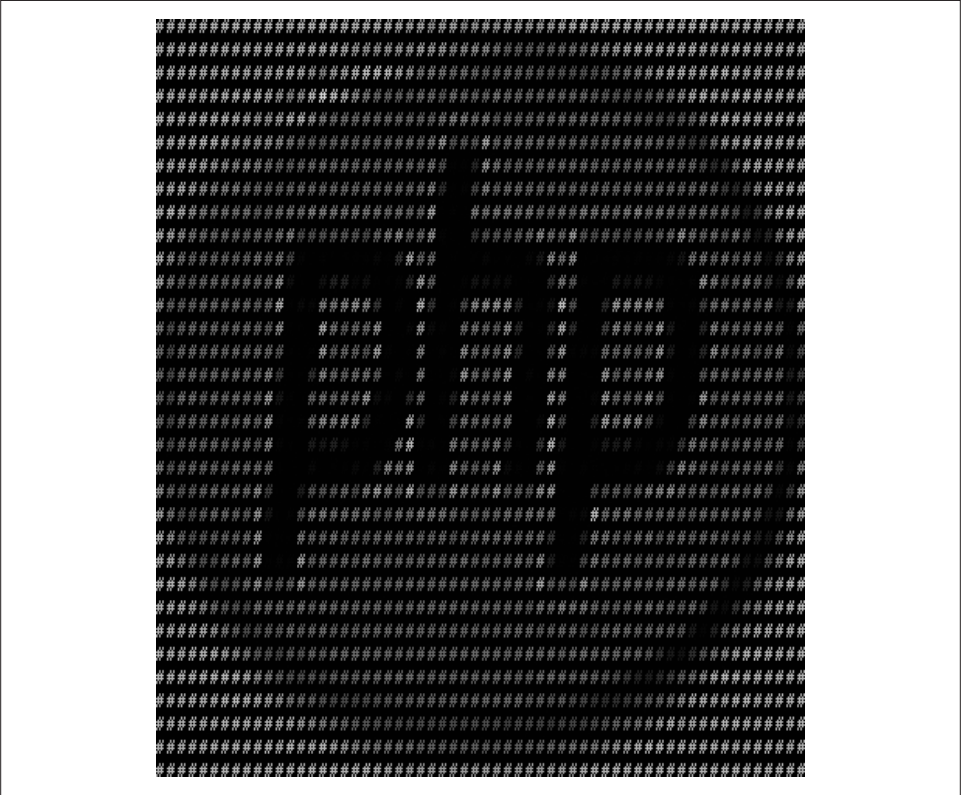
Zanimljiva upotreba funkcije `ImageColorAt()` je proći kroz svaki piksel na slici i provjeriti boju, a zatim učiniti nešto s podacima o njoj.

Primjer 9-14 prikazuje znak # u odgovarajućoj boji za svaki piksel.

Primjer 9-14. Pretvaranje slike u tekst

```
<html><body bgcolor=#000000><tt>
<?php
    $im = imagecreatefromjpeg('php-tiny.jpg');
    $dx = imagesx($im);
    $dy = imagesy($im);
    for($y = 0; $y < $dy; $y++) {
        for($x=0; $x < $dx; $x++) {
            $col = imagecolorat($im, $x, $y);
            $rgb = imagecolorsforindex($im,$col);
            printf('<font color=#%02x%02x%02x>#</font>',
                $rgb['red'], $rgb['green'], $rgb['blue']);
        }
        echo "<br>\n";
    }
}
```

Rezultat je ASCII prikaz slike, kao što je prikazano na slici 9-16.



Slika 9-16. ASCII prikaz slike