Cirkeldiagram med PHP og GD-Lib

I denne artikel vil jeg beskrive, hvordan med ved hjælp af PHP og GD-Lib kan lave "dynamiske" cirkeldiagrammer. Denne artikel kræver lidt kendskab til PHP, GD-Lib og matematik i forvejen, hvis man vil forstå, hvad der sker.

Skrevet den **09. Feb 2009** af **mikl-dk** I kategorien **Programmering / PHP** | ★★★☆☆

Indledende kommentarer

I denne artikel vil jeg beskrive, hvordan med ved hjælp af PHP og GD-Lib kan lave "dynamiske" cirkeldiagrammer. Denne artikel kræver lidt kendskab til PHP, GD-Lib og matematik i forvejen, men hvis man gider læse dokumentationen på de links, jeg har givet, kan man sagtens lave dette uden foregående kendskab. Det kræver også, at man har lidt forstand på cirkler, trigonometri, men det ridser jeg lige op, så vi har alle med.

Man skal være opmærksom på, at der i denne artikel ikke er gjort noget i emnet inputvalidering, så man skal give scriptet den rigtige type oplysninger for det virker.

Teori om cirkler

Når man skal lave et cirkeldiagram, skal man, som navnet siger, lavet en illustrativt opdelt cirkel ved hjælp af farver. For at gøre det, laver man nogle streger fra centrum af cirklen og ud til cirkelperiferien ("selve cirklen" så at sige). Derefter farver man disse områder. Det, der rent praktisk er lettest at gøre, er at først "tegne" et stykke ved hjælp af streger og herefter fylde det med farve, og derefter næste osv.

Hvis man kigger på illustrationen nedenfor kan man se nogle betegnelser. Hvis man bruger nogle simple trigonometriske regler, kan man finde punktet (x,y) som er det punkt i cirkelperiferien, som den linie fra centrum skal gå ud til ud fra den angivne antal grader. Jeg forklarer ikke trigonometrien, da det godt kan være svært at forstå, hvis man endnu ikke har lært det.

Illustration af cirkelteori
Illustration af vinkler i cirkel
Illustration af punkter i cirkel

Indledende udregninger

Den måde, man finder ud af, hvor store "cirkelstykkerne" skal være er, at man tager summen af inputtallene. Dette gøres med php-funktionen array_sum (URL: http://php.net/array_sum). Da vi ved, at der er 360° i en cirkel bruges denne oplysning til at finde forholdet mellem antallet af grader samt summen af input. Det forhold kalder vi "degreesPrUnit", og er givet ved:

```
$sumOfNumbers = array_sum($numbers);
```

```
$degreesPrUnit    = 360 / $sumOfNumbers;
```

Dette tal skal vi senere bruge, når vi skal finde de tilhørende x- og y-værdier. Vi skal også indledende sætte nogle indstillinger for scriptet - så som fx en margin, hvor mange pixels en forklaringslinie skal fylde (en farveboks samt en forklaringstekst (kommer senere)). Jeg har kaldt dem følgende, og har givet dem følgende værdier, og sat sammen med overstående giver det følgende:

```
reset($numbers);
reset($names);

$countNumbers = count($numbers);
$countNames = count($names);
$settingSizePrNameDescription = 15;
$settingMargin = 5;
$width = $settingMargin + (2 * $radius) + $settingMargin;
$height = $settingMargin + (2 * $radius) + $settingMargin +
($settingSizePrNameDescription * $countNumbers) + $settingMargin;
$centerCircle = round($width / 2);
$sumOfNumbers = array_sum($numbers);
$degreesPrUnit = 360 / $sumOfNumbers;
$numbersDegrees;
$namesOnSlices;
```

Først sættes pointeren til starten i array'erne med funktionen reset (URL: http://php.net/reset). Derefter sætter jeg, at der skal 15 px til hver række forklaring. Jeg sætter ydermere margin til 5 px. Nu har jeg nok oplysninger til at regne bredden og højden ud. Bredden er margin i venstre side + cirklens bredde (2*r) + margin i højre side. Højden på billedet skal være det samme som bredden plus størrelsen på hver forklaring ganget med antallet af forklaringer (antal numre, der er i input). De sidste to variabelinitialiseringer er blot for at sætte dem, så vi ved, at de senere bliver brugt.

Nu oprettes billedet med de størrelser, vi har regnet ud, med funktionen imagecreatetruecolor (URL: http://php.net/imagecreatetruecolor). Jeg initialiserer også nogle farver (en til baggrund, en til at tegne cirklen samt andre småting og til sidst 10 til forskellige cirkelstykker - hvis man skal have flere tal ind i cirkeldiagrammet, initialiserer man blot flere på samme måde) med funktionen imagecolorallocate (URL: http://php.net/imagecolorallocate). Derefter tegner jeg en hvid baggrund med funktionen imagefilledrectangle (URL: http://php.net/imagefilledrectangle). Til sidst tegner jeg selve cirklen med funktionen imageellipse (URL: http://php.net/imageellipse).

```
// billedet oprettes
$img = imagecreatetruecolor($width, $height);

// farver initialiseres
$colorsWhite = imagecolorallocate($img, 0xFF, 0xFF, 0xFF);
$colorsBlack = imagecolorallocate($img, 0x00, 0x00, 0x00);
$colorsCircle[1] = imagecolorallocate($img, 0xFF, 0x00, 0x00);
$colorsCircle[2] = imagecolorallocate($img, 0x00, 0xFF, 0x00);
$colorsCircle[3] = imagecolorallocate($img, 0x00, 0x00, 0xFF);
$colorsCircle[4] = imagecolorallocate($img, 0xFF, 0xFF, 0x00);
$colorsCircle[5] = imagecolorallocate($img, 0xFF, 0x00, 0xFF);
$colorsCircle[6] = imagecolorallocate($img, 0x00, 0xFF, 0xFF);
$colorsCircle[7] = imagecolorallocate($img, 0xEE, 0xEE, 0xEE);
$colorsCircle[8] = imagecolorallocate($img, 0xCC, 0xCC, 0xCC);
$colorsCircle[9] = imagecolorallocate($img, 0x99, 0x99, 0x99);
$colorsCircle[10] = imagecolorallocate($img, 0x66, 0x66, 0x66);

// baggrundsfarve
```

```
imagefilledrectangle($img, 0, 0, $width, $height, $colorsWhite);

// selve cirklen tegnes
imageellipse($img, $centerCircle, $centerCircle, 2 * $radius, 2 * $radius,
$colorsBlack);
```

Herefter løber jeg alle inputtallene igennem og ganger med det forhold vi før fandt ud af, degreesPrUnit, og lagrer i et array, numbersDegrees. Samtidig regner jeg ud, hvor mange procent tallet (der bruger jeg funktionen number_format til at gøre tallet pænere - URL: http://php.net/number-format) er i forhold til summen og laver fylder i et array, numbersTmp, der bliver brugt, sådan, at hvis der ikke er angivet noget navn til et nummer, så bliver inputtallet skrevet som en form for "erstatning".

```
foreach($numbers as $value)
{
    $numbersDegrees[] = $value * $degreesPrUnit;
    $numbersProcentOfSum[] = number_format(($value/$sumOfNumbers) * 100, 2, ",",
".");
    $numbersTmp[] = $value;
}

foreach($names as $value)
{
    $namesOnSlices[] = $value;
}
```

Nu begynder jeg at tegne cirkelafsnittene og farve dem. Til at tegne linier med, bliver funktionen imageline brugt (URL: http://php.net/imageline). Jeg laver gradeantallet om til radianer med funktionen deg2rad da funktionerne sin og cos tager antal radianer som argument (URL's: http://php.net/deg2rad , http://php.net/cos og http://php.net/sin). Jeg fylder afsnittene med funktionen imagefilltoborder (URL: http://php.net/imagefilltoborder). Jeg skriver det tilhørende navn (eller blot tallet, hvis der ikke er noget navn) samt procent i parentes bagefter med funktionen imagestring (URL: http://php.net/imagestring).

```
$indexColor = 1;
$indexDegree = 0;

foreach($numbersDegrees as $value)
{
    $degree = $value + $indexDegrees;
    $indexDegrees = $degree;

    // koordinater til at farve cirkelafsnit beregnes i følgende afsnit kode
        // konvertér fra grader til radian
        $rad = deg2rad($indexDegrees);
        // x-koordinaten beregnes
        $x = (cos($rad) * ($radius - 5)) + $centerCircle;
        // y-koordinaten beregnes
        $y = (sin($rad) * ($radius - 5)) + $centerCircle;
        imagefilltoborder($img, $x, $y, $colorsBlack, $colorsCircle[$indexColor
+ 1]);

    // koordinater til at tegne stregerne til afgrænsning af cirkelafsnit
beregnes i følgende afsnit kode
        // konvertér fra grader til radian
        $rad = deg2rad($degree);
        $x = (cos($rad) * ($radius - 1)) + $centerCircle;
        $y  (sin($rad) * ($radius - 1)) + $centerCircle;
```

```
        imageline($img, $centerCircle, $centerCircle, $x, $y, $colorsBlack);

    // koordinater til at tegne den sorte firkant farven til forklaring
forklaringen af data
    imagerectangle($img, $settingMargin, (2 * $radius) + (2 * $settingMargin) +
(($indexColor - 1) * $settingSizePrNameDescription), ($settingMargin) + 10, ((2
* $radius) + (2 * $settingMargin) + (($indexColor - 1) *
$settingSizePrNameDescription)) + 10, $colorsBlack);

     // forklaringsboksene skal også have lidt farve
     imagefilltoborder($img, $settingMargin + 1, ((2 * $radius) + (2 *
$settingMargin) + (($indexColor - 1) * $settingSizePrNameDescription)) + 1,
$colorsBlack, $colorsCircle[$indexColor + 1]);

    // hvis der ikke er noget tilhørende navn
    if ($namesOnSlices[$indexColor - 1] == "")
        $namesOnSlices[$indexColor - 1] = $numbersTmp[$indexColor - 1];

    // info skrives
    imagestring($img, 2, 20, (2 * $radius) + (2 * $settingMargin) +
(($indexColor - 1) * $settingSizePrNameDescription), $namesOnSlices[$indexColor
- 1] . " (" . $numbersProcentOfSum[$indexColor - 1] . "%)", $colorsBlack);

    // næste!
    $indexColor++;
}

// farver sidste slice - samme princip som alle de andre
$rad = deg2rad(1);
$x = (cos($rad) * ($radius - 5)) + $centerCircle;
$y = (sin($rad) * ($radius - 5)) + $centerCircle;
imagefilltoborder($img, $x, $y, $colorsBlack, $colorsCircle[2]);

// billedet tegnes
imagepng($img);
imagedestroy($img);
```

Optimering af koden

Man kan sagtens optimere og arbejde med koden, men meget af det kræver, at man laver nogle skarpere regler for input - det optimale ville være at lave en funktion til at validere input, og så optimere funktionerne endnu mere. Dette vil jeg ikke beskrive i denne artikel, da denne blot er for at give en idé om, hvordan det kunne laves, og på den måde vise princippet bag brug af GD-Lib, PHP og matematik til noget smart og praktisk.

Samlet kode

```
function cirkeldiagram($numbers, $names, $radius)
{
    reset($numbers);
```

```
    reset($names);

    $countNumbers = count($numbers);
    $countNames = count($names);
    $settingSizePrNameDescription = 15;
    $settingMargin = 5;
    $width = $settingMargin + (2 * $radius) + $settingMargin;
    $height = $settingMargin + (2 * $radius) + $settingMargin +
($settingSizePrNameDescription * $countNumbers) + $settingMargin;
    $centerCircle = round($width / 2);
    $sumOfNumbers = array_sum($numbers);
    $degreesPrUnit = 360 / $sumOfNumbers;
    $numbersDegrees;
    $namesOnSlices;

    // billedet oprettes
    $img = imagecreatetruecolor($width, $height);

    // farver initialiseres
    $colorsWhite = imagecolorallocate($img, 0xFF, 0xFF, 0xFF);
    $colorsBlack = imagecolorallocate($img, 0x00, 0x00, 0x00);
    $colorsCircle[1] = imagecolorallocate($img, 0xFF, 0x00, 0x00);
    $colorsCircle[2] = imagecolorallocate($img, 0x00, 0xFF, 0x00);
    $colorsCircle[3] = imagecolorallocate($img, 0x00, 0x00, 0xFF);
    $colorsCircle[4] = imagecolorallocate($img, 0xFF, 0xFF, 0x00);
    $colorsCircle[5] = imagecolorallocate($img, 0xFF, 0x00, 0xFF);
    $colorsCircle[6] = imagecolorallocate($img, 0x00, 0xFF, 0xFF);
    $colorsCircle[7] = imagecolorallocate($img, 0xEE, 0xEE, 0xEE);
    $colorsCircle[8] = imagecolorallocate($img, 0xCC, 0xCC, 0xCC);
    $colorsCircle[9] = imagecolorallocate($img, 0x99, 0x99, 0x99);
    $colorsCircle[10] = imagecolorallocate($img, 0x66, 0x66, 0x66);

    // baggrundsfarve
    imagefilledrectangle($img, 0, 0, $width, $height, $colorsWhite);

    // selve cirklen tegnes
    imageellipse($img, $centerCircle, $centerCircle, 2 * $radius, 2 * $radius,
$colorsBlack);

    foreach($numbers as $value)
    {
        $numbersDegrees[] = $value * $degreesPrUnit;
        $numbersProcentOfSum[] = number_format(($value/$sumOfNumbers) * 100, 2,
",", ".");
        $numbersTmp[] = $value;
    }

    foreach($names as $value)
    {
        $namesOnSlices[] = $value;
    }

    $indexColor = 1;
    $indexDegree = 0;
```

```
    foreach($numbersDegrees as $value)
    {
        $degree = $value + $indexDegrees;
        $indexDegrees = $degree;

        // koordinater til at farve cirkelafsnit beregnes i følgende afsnit kode
            // konvertér fra grader til radian
            $rad = deg2rad($indexDegrees);
            // x-koordinaten beregnes
            $x = (cos($rad) * ($radius - 5)) + $centerCircle;
            // y-koordinaten beregnes
            $y = (sin($rad) * ($radius - 5)) + $centerCircle;
            imagefilltoborder($img, $x, $y, $colorsBlack,
$colorsCircle[$indexColor + 1]);

        // koordinater til at tegne stregerne til afgrænsning af cirkelafsnit
beregnes i følgende afsnit kode
            // konvertér fra grader til radian
            $rad = deg2rad($degree);
            $x = (cos($rad) * ($radius - 1)) + $centerCircle;
            $y  (sin($rad) * ($radius - 1)) + $centerCircle;
            imageline($img, $centerCircle, $centerCircle, $x, $y,
$colorsBlack);
        // koordinater til at tegne den sorte firkant farven til forklaring
forklaringen af data
        imagerectangle($img, $settingMargin, (2 * $radius) + (2 *
$settingMargin) + (($indexColor - 1) * $settingSizePrNameDescription),
($settingMargin) + 10, ((2 * $radius) + (2 * $settingMargin) + (($indexColor -
1) * $settingSizePrNameDescription)) + 10, $colorsBlack);

         // forklaringsboksene skal også have lidt farve
         imagefilltoborder($img, $settingMargin + 1, ((2 * $radius) + (2 *
$settingMargin) + (($indexColor - 1) * $settingSizePrNameDescription)) + 1,
$colorsBlack, $colorsCircle[$indexColor + 1]);

        // hvis der ikke er noget tilhørende navn
        if ($namesOnSlices[$indexColor - 1] == "")
        $namesOnSlices[$indexColor - 1] = $numbersTmp[$indexColor - 1];

        // info skrives
        imagestring($img, 2, 20, (2 * $radius) + (2 * $settingMargin) +
(($indexColor - 1) * $settingSizePrNameDescription), $namesOnSlices[$indexColor
- 1] . " (" . $numbersProcentOfSum[$indexColor - 1] . "%)", $colorsBlack);

        // næste!
        $indexColor++;
    }

    // farver sidste slice - samme princip som alle de andre
    $rad = deg2rad(1);
    $x = (cos($rad) * ($radius - 5)) + $centerCircle;
    $y = (sin($rad) * ($radius - 5)) + $centerCircle;
    imagefilltoborder($img, $x, $y, $colorsBlack, $colorsCircle[2]);

    // billedet tegnes
```

```
    imagepng($img);
    imagedestroy($img);
}

// kunne kaldes på følgende måde:
$stats = explode(",", $_GET["s"]);
$navne = explode(",", $_GET["n"]);
header("Content-type: image/png");
cirkeldiagram($stats, $navne, $_GET["r"]);
```

Hvad kan jeg bruge det til?

Noget, dette ville være smart at bruge til er fx en afstemningsstatistik. Fx pga., at der er indbygget procentregning i, så tallene blot skal læses ud fra databasen og sættes ind i et array, scriptet så kan "fodres" med.

Det var alt!

Det var alt for denne gang. Skriv meget gerne ris/ros. Jeg har været lidt i tvivl om, hvor meget der skulle med, men dette var, hvad det endte med :)

Når du laver scriptet, så husk på, at der ingen inputvalidering er (betyder bl.a. at specialtegn som fx æ, ø og å ikke kan vises med imagestring - det betyder også, at hvis man vil skrive kommatal skal det gøres med engelsk komma - punktum - fx 45.6)

Et diagram lavet med overstående script

Dette er lavet med URL'en (blot for eksemplets skyld):
cirkeldiagram.php?r=50&s=50,45,87,54&n=Dansk,Svensk,Norsk,Finsk
<a
href="http://www.mikl.dk/artikler/php/cirkeldiagram_med_php_og_gdlib/cirkel_eksempel.png">Eksempel
på diagram

Kommentar af danielhep d. 24. Apr 2004 | 1

ja dette er en artikel, men der mangler meget forklaring til dem som er noops :)

Kommentar af j_jorgensen d. 23. Apr 2004 | 2

Og iøvrigt findes der MANGE php klasser derude til at tegne pæne grafer. Google er din ven :)

Kommentar af smooth d. 13. Apr 2004 | 3

Kommentar af mulbo d. 21. Apr 2004 | 4

nice der

Kommentar af ofirpeter d. 23. Nov 2004 | 5

Dumt at lægge et link ud til ens egen side, når man alligevel vælger at fjerne siden.

Kommentar af freakstyle d. 17. Apr 2004 | 6

Kommentar af wicez (nedlagt brugerprofil) d. 06. Oct 2004 | 7

Syntes nok at jeg havde set den før :P -> http://udvikleren.dk/article.php?aid=177&techid=6

meget velskrevet artikel!

Kommentar af sagetbob d. 27. Apr 2004 | 8

hm ok, men jeg kommer nok til at brugge lidt tid på google også....