

creating a gauge part 2

Ok, we got our main function and some dummy functions to get some cpu and bandwidth values.

Our graphics will be created using the libgd toolkit, which makes dealing with the images quite easy.

To be able to use it, include gd.h into your program.

```
#include <gd.h>
```

MakeImage

```
void MakeImage(int cpu, int bandwidth)
{
    FILE *fh=fopen("input.png","r");
    FILE *out=fopen("output.jpg","w");
    if (fh & out)
    {
        im=gdImageCreateFromPng(fh);
        fclose(fh);
        drawpointer(im,cpu,1);
        drawpointer(im,bandwidth,2);
        gdImageJpeg(im, out,-1);
        fclose(out);
        gdImageDestroy(im);
    }
}
```

This creates an image from a png file (input.png), draws 2 pointers into the image and writes a jpg file.

The actual drawing will be done in drawpointer.

drawpointer

We have the cpu and bandwidth values stored as an integer from 0-100.

These values need to be converted into angle for our pointer.

```
angle=(58.0+(246.0*percent)/100.0);
angle=(angle/180.0)*3.1415;
```

The first line converts the percent value into an angle between 58 and 246. All functions that will follow need the angle not in \hat{A}° , they need it in a 0- 2π range.

The second line does the converting.

Optimizing the above lines is left for the reader.

The pointer will be drawn as a solid bright red polygon surrounded by a darker red line.

To get the coordinates for the polygon, the following lines are needed:

```
points[0].x=-4.0*cos(angle)+XOffset;
points[0].y=-4.0*sin(angle)+YOffset;
points[1].x=4.0*cos(angle)+XOffset;
points[1].y=4.0*sin(angle)+YOffset;
points[2].x=-sin(angle)*radius+2.0*cos(angle)+XOffset;
points[2].y=cos(angle)*radius+2.0*sin(angle)+YOffset;
points[3].x=-sin(angle)*radius-2.0*cos(angle)+XOffset;
points[3].y=cos(angle)*radius-2.0*sin(angle)+YOffset;
gdImageFilledPolygon(im, points, 4, red_bright);
gdImagePolygon(im, points, 4, red_dark);
```

This will create the needed coordinates and will draw them to the image. XOffset and YOffset will be calculated depending on the given parameters (1 or 2 as the third argument). The complete function drawpointer:

```
void drawpointer(gdImagePtr im, int percent, int factor)
{
    double angle=0;
    double radius=55.0;
    gdPoint points[4];
    double XOffset=66.0+132*(factor-1);
    double YOffset=61.0;
    int red_bright;
    int red_dark;
    int white;
    angle=(58.0+(246.0*percent)/100.0);
    angle=(angle/180.0)*3.1415;
    red_bright = gdImageColorAllocate(im, 250, 50, 0);
    red_dark = gdImageColorAllocate(im, 150, 20, 0);
    white = gdImageColorAllocate(im, 255, 255, 255);
    points[0].x=-4.0*cos(angle)+XOffset;
    points[0].y=-4.0*sin(angle)+YOffset;
    points[1].x=4.0*cos(angle)+XOffset;
    points[1].y=4.0*sin(angle)+YOffset;
    points[2].x=-sin(angle)*radius+2.0*cos(angle)+XOffset;
    points[2].y=cos(angle)*radius+2.0*sin(angle)+YOffset;
    points[3].x=-sin(angle)*radius-2.0*cos(angle)+XOffset;
    points[3].y=cos(angle)*radius-2.0*sin(angle)+YOffset;
    gdImageFilledPolygon(im, points, 4, red_bright);
    gdImagePolygon(im, points, 4, red_dark);
}
```

The last remaining step is to get the actual cpu and bandwidth information. This will be the last part of this tutorial.