

**1. Introduction.** It is very common nowadays that we use rounded corner background images in web pages. I have no idea how the concept of “Rounded rectangle” was introduced into graphical user interface. But I think it was probably done by Steve Jobs. Search “Round Rects Are Everywhere” on Google.

This program generates a rounded bitmap. User can specify parameters such as the width, height and radius of the rectangle. Interestingly, a radius of zero indicates a regular rectangle.

For help, please see “nifty -help”.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
<Data types 3>;
<Global variables 8>;
<Function prototypes 4>;
int main(int argc, char *argv[]){ BYTE *imgbits = 0;
  <Parse arguments 9>;
  imgbits = ( BYTE * ) malloc(width * height * (BITS_PER_PIXEL/8));
  if (imgbits) {
    fill(imgbits, width, height, fgcolor);
    rounded_corner(imgbits, width, height, radius, bgcolor, top);
    write_image(imgbits, width, height, outfile);
    free(imgbits);
  }
}
```

**2. Bitmap Related Structures and Operations.**

```

#define BYTE unsigned char
#define WORD unsigned short
#define DWORD unsigned long
#define LONG long
#define BI_RGB 0_L
#define BITS_PER_PIXEL 24

```

**3. <Data types 3> ≡**

```

#pragma pack (1)
typedef struct tagBITMAPFILEHEADER {
    WORD bfType;
    DWORD bfSize;
    WORD bfReserved1;
    WORD bfReserved2;
    DWORD bfOffBits;
} BITMAPFILEHEADER;
typedef struct tagBITMAPINFOHEADER {
    DWORD biSize;
    LONG biWidth;
    LONG biHeight;
    WORD biPlanes;
    WORD biBitCount;
    DWORD biCompression;
    DWORD biSizeImage;
    LONG biXPelsPerMeter;
    LONG biYPelsPerMeter;
    DWORD biClrUsed;
    DWORD biClrImportant;
} BITMAPINFOHEADER;
#pragma pack ()

```

This code is used in section 1.

**4. <Function prototypes 4> ≡**

```

void write_image(BYTE * pBits, int nSrcWidth, int nSrcHeight, const char *filename);

```

See also sections 6, 10, and 12.

This code is used in section 1.

5. Procedure *write\_image*. Write the image bits into a BMP file.

```

void write_image(BYTE * pBits, int nSrcWidth, int nSrcHeight, const char *filename)
{
    BITMAPINFOHEADER bmih;
    FILE *fp;
    BITMAPFILEHEADER bmfh;

    memset(&bmih, 0, sizeof(BITMAPINFOHEADER));
    bmih.biSize = sizeof(BITMAPINFOHEADER);
    bmih.biBitCount = BITS_PER_PIXEL;
    bmih.biPlanes = 1;
    bmih.biWidth = nSrcWidth;
    bmih.biHeight = nSrcHeight;
    bmih.biCompression = BI_RGB;
    memset(&bmfh, 0, sizeof(BITMAPFILEHEADER));
    bmfh.bfType = #4d42;
    bmfh.bfOffBits = sizeof(BITMAPINFOHEADER) + sizeof(BITMAPFILEHEADER);
    bmfh.bfSize = bmfh.bfOffBits + nSrcWidth * nSrcHeight * (BITS_PER_PIXEL/8);
    fp = fopen(filename, "wb");
    if (fp) {
        fwrite(&bmfh, 1, sizeof(BITMAPFILEHEADER), fp);
        fwrite(&bmih, 1, sizeof(BITMAPINFOHEADER), fp);
        fwrite(pBits, 1, nSrcWidth * nSrcHeight * 3, fp);
        fclose(fp);
    }
}

```

6. ⟨Function prototypes 4⟩ +≡

```

void rgb(unsigned long color, BYTE * r, BYTE * g, BYTE * b);

```

7. Procedure *rgb*. extract R, G, B values from an integer.

```

void rgb(unsigned long color, BYTE * r, BYTE * g, BYTE * b)
{
    *r = (BYTE)((color & #ff0000) >> 16);
    *g = (BYTE)((color & #ff00) >> 8);
    *b = (BYTE)(color & #ff);
}

```

8. ⟨Global variables 8⟩ ≡

```

unsigned long fgcolor = 0;
unsigned long bgcolor = #ffffff;
int width = 200;
int radius = 20;
int height = 20;
int top = 1;
char outfile[100] = "out.bmp";

```

This code is used in section 1.

9. Parse the arguments from user.

```
#define EQ(s,t) (strcmp(s,t) == 0)
⟨Parse arguments 9⟩ ≡
{
  int i = 0;
  for (i = 1; i < argc; i++) {
    if (EQ(argv[i], "-fg")) {
      fgcolor = strtol(argv[++i], 0, 16);
    }
    else if (EQ(argv[i], "-bg")) {
      bicolor = strtol(argv[++i], 0, 16);
    }
    else if (EQ(argv[i], "-r")) {
      radius = strtol(argv[++i], 0, 10);
    }
    else if (EQ(argv[i], "-w")) {
      width = strtol(argv[++i], 0, 10);
    }
    else if (EQ(argv[i], "-h")) {
      height = strtol(argv[++i], 0, 10);
    }
    else if (EQ(argv[i], "-top")) {
      top = 1;
    }
    else if (EQ(argv[i], "-bot")) {
      top = 0;
    }
    else if (EQ(argv[i], "-o")) {
      strcpy(outfile, argv[++i]);
    }
    else if (EQ(argv[i], "--help")) {
      usage();
    }
  }
  if (height < radius) height = radius;
}
```

This code is used in section 1.

10. ⟨Function prototypes 4⟩ +≡

```
void usage();
```

11. Procedure *usage*. Print help message.

```

void usage()
{
    fprintf(stderr, "\n"
        "_NIFTY_ Generate background image (*.bmp) with round corners\n"
        "\n"
        "Usage: _nifty_ [options]\n"
        "Options:\n"
        "    -fg<color>    css_hex_color_for_foreground, e.g. c7eacc\n"
        "    -bg<color>    css_hex_color_for_background, e.g. ffffff\n"
        "    -r            corner_radius, default 20\n"
        "    -w            output_width, default 200\n"
        "    -h            output_height, default 20\n"
        "    -top          generate_round_corners_at_top. This is default\n"
        "    -bot          generate_round_corners_at_bottom\n"
        "    -o<file>     output_bmp_file, default_name_is \"out.bmp\"\n"
        "    --help       print this message\n"
        "\n"
        "Example: _nifty_ -w800 -h30 -r20 -fgc7eacc -bot -oexample.bmp\n"
        "\n"
        "Please send bug report to <lichaoji@gmail.com>.\n"
    );
    exit(1);
}

```

**12. Drawing.**

(Function prototypes 4) +≡

```
void rounded_corner(BYTE * imgbits, int width, int height, int radius, unsigned long bcolor, int top);  
void fill(BYTE * imgbits, int width, int height, unsigned long fgcolor);
```

**13.** Procedure *fill*. Fill image buffer with single color.

```
void fill(BYTE * imgbits, int width, int height, unsigned long fgcolor)  
{  
    BYTE r, g, b;  
    int n = 0;  
    rgb(fgcolor, &r, &g, &b);  
    n = width * height;  
    while (n-- > 0) {  
        *imgbits++ = r;  
        *imgbits++ = g;  
        *imgbits++ = b;  
    }  
}
```

14. Procedure *rounded\_corner*.

```

void rounded_corner(BYTE * imgbits, int width, int height, int radius, unsigned long bgcolor, int top)
{
    int stride;
    int step;
    int x;
    BYTE r, g, b;
    rgb(bgcolor, &r, &g, &b);
    stride = width * BITS_PER_PIXEL/8;
    if (radius > height) radius = height;
    if (radius > width) radius = width;
    if (!top) {
        step = stride;
    }
    else {
        step = -stride;
        imgbits += stride * (height - 1);
    }
    for (x = radius; x > 0; x--, imgbits += step) {
        int y, i;
        BYTE * p, *q;
        y = (int) floor(((double) radius - sqrt(((double)(radius * radius - x * x))));
        if (y == radius) {
            if (x > 1) {
                int d;
                d = y - (int) floor(((double) radius - sqrt(((double)(radius * radius - (x - 1) * (x - 1)))));
                if (d > 2) {
                    y -= d/2;
                }
            }
        }
        p = imgbits;
        q = imgbits + stride;
        for (i = 0; i < y; i++) {
            *p++ = r;
            *p++ = g;
            *p++ = b;
            *--q = b;
            *--q = g;
            *--q = r;
        }
    }
}

```

**15. Index.**

- argc*: 1, 9.
- argv*: 1, 9.
- bfOffBits*: 3, 5.
- bfReserved1*: 3.
- bfReserved2*: 3.
- bfSize*: 3, 5.
- bfType*: 3, 5.
- bicolor*: 1, 8, 9, 12, 14.
- BI\_RGB: 2, 5.
- biBitCount*: 3, 5.
- biClrImportant*: 3.
- biClrUsed*: 3.
- biCompression*: 3, 5.
- biHeight*: 3, 5.
- biPlanes*: 3, 5.
- biSize*: 3, 5.
- biSizeImage*: 3.
- BITMAPFILEHEADER**: 3, 5.
- BITMAPINFOHEADER**: 3, 5.
- BITS\_PER\_PIXEL: 1, 2, 5, 14.
- biWidth*: 3, 5.
- biXPelsPerMeter*: 3.
- biYPelsPerMeter*: 3.
- bmfh*: 5.
- bmih*: 5.
- BYTE: 1, 2, 4, 5, 6, 7, 12, 13, 14.
- color*: 6, 7.
- d*: 14.
- DWORD: 2, 3.
- EQ: 9.
- exit*: 11.
- fclose*: 5.
- fgcolor*: 1, 8, 9, 12, 13.
- filename*: 4, 5.
- fill*: 1, 12, 13.
- floor*: 14.
- fopen*: 5.
- fp*: 5.
- fprintf*: 11.
- free*: 1.
- fwrite*: 5.
- height*: 1, 8, 9, 12, 13, 14.
- i*: 9, 14.
- imgbits*: 1, 12, 13, 14.
- LONG: 2, 3.
- main*: 1.
- malloc*: 1.
- memset*: 5.
- n*: 13.
- nSrcHeight*: 4, 5.
- nSrcWidth*: 4, 5.
- outfile*: 1, 8, 9.
- pack*: 3.
- pBits*: 4, 5.
- radius*: 1, 8, 9, 12, 14.
- rgb*: 6, 7, 13, 14.
- rounded\_corner*: 1, 12, 14.
- sqrt*: 14.
- stderr*: 11.
- step*: 14.
- strcmp*: 9.
- strcpy*: 9.
- stride*: 14.
- strtol*: 9.
- tagBITMAPFILEHEADER**: 3.
- tagBITMAPINFOHEADER**: 3.
- top*: 1, 8, 9, 12, 14.
- usage*: 9, 10, 11.
- width*: 1, 8, 9, 12, 13, 14.
- WORD: 2, 3.
- write\_image*: 1, 4, 5.
- x*: 14.
- y*: 14.

- ⟨Data types 3⟩ Used in section 1.
- ⟨Function prototypes 4, 6, 10, 12⟩ Used in section 1.
- ⟨Global variables 8⟩ Used in section 1.
- ⟨Parse arguments 9⟩ Used in section 1.



# NIFTY

	Section	Page
Introduction .....	1	1
Bitmap Related Structures and Operations .....	2	2
Drawing .....	12	6
Index .....	15	8