Supplements to:

CAIROTEP: ORTEP-III incorporates CAIRO for better outputs

1. CRTEP parameters

Position	Keyword	Value	Default	Description
0	back	[cairo| ps| hpgl]	cairo	plotting backend, [cairo] or the legacy [ps/hpgl] plotting routines.
For the CAIRO backend only:				
1	type	[png| svg| pdf| ps]	png	surface/image type for the cairo backend.
2	dpi	integer	72	resolution (dpi).
3	cw	float	(by ins)	canvas width (inch).
4	ch	float	(by ins)	canvas height (inch).
5	frag	[yes| no]	yes	fragmentation mode. [yes], to fragment paths on every pen move, resulting in more editable vector output; or [no], for lazy fragmentation only when the pen color/width changed, to reduce the size of vector files.
6	scale	integer	100	canvas scale (%).
7	fg	integer	0	initial pen color, 24-bit decimal or hexadecimal number.[*]
8	bg	integer	0xFFFFFF	canvas background, 24-bit decimal or hexadecimal number, [-1] for transparent.[*]
9	pw	float	1	initial pen width (pixel).[**]
For the PS/HPGL backend only:				
1	orient	[portrait| landscape]	portrait	orient for ps/hpgl documents.

[*] Hexadecimal number notation: #HHHHHH or 0xHHHHHH.

[**] The pen width value is specified in pixels for CAIRO, while the other dimension related values are specified in inches.

2. Extending CRTEP via

The following example shows a simple TIFF output routine using the LZW compression scheme with the help of LIBTIFF:

```
void crtep_save_tiff(cairo_surface_t *surface, const char
*outfname) {

    TIFF *tiffout;
    char *data = cairo_image_surface_get_data(surface);
    int width = cairo_image_surface_get_width(surface);
    int length = cairo_image_surface_get_height(surface);
    if((tiffout = TIFFOpen(outfname, "w")) == NULL) {
        // ...
    }

    TIFFSetField(tiffout, TIFFTAG_IMAGEWIDTH, width);
    TIFFSetField(tiffout, TIFFTAG_IMAGELENGTH, length);
    TIFFSetField(tiffout, TIFFTAG_ORIENTATION,
ORIENTATION_TOPLEFT);
    TIFFSetField(tiffout, TIFFTAG_SAMPLESPERPIXEL, 3);
    TIFFSetField(tiffout, TIFFTAG_BITSPERSAMPLE, 8);
    TIFFSetField(tiffout, TIFFTAG_PLANARCONFIG,
PLANARCONFIG_CONTIG);
    TIFFSetField(tiffout, TIFFTAG_PHOTOMETRIC, PHOTOMETRIC_RGB);
    TIFFSetField(tiffout, TIFFTAG_COMPRESSION, COMPRESSION_LZW);

    int rps = TIFFDefaultStripSize(tiffout, rps);
    if(rps > length) {
        rps = length;
    }
    TIFFSetField(tiffout, TIFFTAG_ROWSPERSTRIP, rps);

    if(cairo_image_surface_get_format(surface) ==
CAIRO_FORMAT_RGB24) {
        char* buf = malloc(width*3);
        char *pd = data, *pb = buf;
        int ir, i;
        for(ir = 0; ir < length; ir++) {
            pb = buf;
            for(i = 0; i < width; i++) {
                // BGRX to RGB:
                *pb = *(pd+2); pb++;
                *pb = *(pd+1); pb++;
                *pb =     *pd; pb++; pd+=4;
            }
            if(TIFFWriteScanline(tiffout, buf, ir, 0) != 1) {
                // ...
            }
        }
        free(buf);
    } else if(cairo_image_surface_get_format(crtep_surface) ==
CAIRO_FORMAT_ARGB32) {
        char* buf = malloc(width*3);
        char *pd = data, *pb = buf;
        int ir, i;
        for(ir = 0; ir < length; ir++) {
            pb = buf;
            for(i = 0; i < width; i++) {
                // BGRA (LE) to RGB:
                char ca = 0xFF-*(pd+3);
                *pb = *(pd+2) + 0xFF*ca/255.; pb++;
                *pb = *(pd+1) + 0xFF*ca/255.; pb++;
                *pb =     *pd + 0xFF*ca/255.; pb++; pd+=4;
            }
            if(TIFFWriteScanline(tiffout, buf, ir, 0) != 1) {
                // ...
            }
        }
        free(buf);
    }
    TIFFClose(tiffout);
}
// ...
cairo_surface_t *crtep_surface;
char *crtep_outfname;
// ...
crtep_save_tiff(crtep_surface, crtep_outfname);
```

3. CIFTEP options

Short	Long	Value	Default	Description
-h	--help	N/A	N/A	show this help message and exit.
N/A	--version	N/A	N/A	show program's version number and exit.
-o	--output	string	N/A	specify the output instruction file path.
-q	--quiet	N/A	N/A	suppress verbose message printing.
-c	--contents	[aunit|grow|cell]	aunit	contents scheme, choose from asymmetric unit, growing or unit cell.
-i	--view	[std|best|100|010|001]	best	initial view scheme, choose from standard view, best view, or the (100), (010), (001) plane.
-n	--normalize-uh	float	0.1	normalize the u values of all H atoms to [value], negative values (e.g. [-1]) mean no normalization.
-e	--probability	integer	50	ellipsoid probability (%)
-b	--no-label	N/A	N/A	suppress plotting atom labels, (0, 0, 0).

-m	--color-map	[mono| color]	color	color map for elements, [mono] for constant pen color, [color] for using the default built-in color map.
Projection control:				
-W	--width	float	10.7	canvas width (inch).
-H	--height	float	8.2	canvas height (inch).
-M	--margin	float	0.3	canvas margin (inch).
-D	--distance	float	30	viewing distance (inch), [0] for the parallel projection.
Rotations:				
-X	--rotation-x	integer	0	specify the rotation about the X axis (degree).
-Y	--rotation-y	integer	0	specify the rotation about the Y axis (degree).
-Z	--rotation-z	integer	0	specify the rotation about the Z axis (degree).
CRTEP execution:				
-t	--tep	N/A	N/A	invoke CRTEP after the instruction file generation.
-p	--tep-params	string	N/A	string passed to CRTEP as parameters. (the 2nd argument of CRTEP.)
-r	--tep-clear	N/A	N/A	clear the instruction file after the CRTEP execution.
N/A	--tep-exec	string	crtep	CRTEP executable path.
N/A	--tep-output	string	N/A	CRTEP output path. (the 3rd argument of CRTEP.)