

About internal fonts

Internal vector fonts have been decoded from original Borland `.CHR` fonts and turned to arrays of coordinates; that is, glyphs.

I'm not aware of the legal status of the `.CHR` fonts that accompanied Borland Turbo C and other compilers. Anyhow, even assuming that they are still copyrighted under U.S. law, their glyphs are not; please see [this Wikipedia article](#) for details. It is therefore legal to incorporate the glyphs of `.CHR` fonts in `SDL_bgi`.

Fonts were taken from Borland Turbo C++ 3.0, available [here](#), and converted using the ancillary program `tmp/chr_decoder.c`.

The 8x8 `DEFAULT_FONT` was dumped from the original font using the ancillary program `tmp/dumpchar.c`.

Loading external fonts

Font output has been tested quite thoroughly using the `tmp/fnttest.c` program, running it in Turbo C++ 3.0 (in DOSBox) and with `SDL_bgi`. Output is pixel-perfect using the internal fonts; however, positioning is slightly off when loading some external `.CHR` fonts (samples kindly provided by Bernie Hirsch).

I am unable to find the cause of this strange behaviour. I do not exclude the possibility of a BGI undocumented feature, a bug in Turbo C++ 3.0, or a bug in the fonts.

Turbo C++ `outtextxy()` bug

Turbo C++ 3.0 has a bug in `outtextxy()`, as shown by the following code:

```
settextstyle (DEFAULT_FONT, VERT_DIR, 0);
settextjustify (LEFT_TEXT, BOTTOM_TEXT);
outtextxy (300, 300, "Hello world");
settextjustify (RIGHT_TEXT, BOTTOM_TEXT);
/* same x position as LEFT_TEXT: */
outtextxy (300, 300, "Hello world");
```

This bug only affects `DEFAULT_FONT`.

For compatibility reasons, `SDL_bgi` intentionally implements this bug. If you want to display text at the expected position, do:

```
outtextxy (x + 2 * textheight(string), y, string);
```