Compatibility

SDL_bgi has been designed to be functionally compatible with the old Borland Graphics Interface (GRAPHICS.H) for DOS, and with WinBGIm, which itself is a mostly complete GRAPHICS.H implementation.

SDL_bgi is a superset of both, and as far as I can say it provides the most compatible GRAPHICS.H implementation available. It should be noted, however, that SDL_bgi is not a Turbo C or Borland C++ emulator! Besides, SDL_bgi is also designed to be portable and to take advantage of modern graphics hardware, thanks to the SDL2 library.

Compatibility with GRAPHICS.H

Compatibility with the original GRAPHICS.H is nearly perfect, but 100% compatibility with old programs written for Turbo C or Borland C++ is simply *impossible* to attain. By design, Borland compilers were not portable; they were specifically designed for the PC/DOS platform. Hence, they implemented low-level details such as hardware key codes, memory models, DOS and BIOS calls, online assembly, and so on. Besides, even in the original Turbo C / Borland C++ different graphic drivers were not fully compatible with one another. For example, programs written for the IBM8514.BGI driver needed modifications to compile and run on the EGAVGA.BGI driver.

Full compatibility is only possible in a hardware emulator like DOSBox. If a program uses CONIO.H, DOS.H, BIOS.H and the like, chances are you won't be able to compile it. Please consider using DOSBox and one of the original Borland compilers that are available as freeware.

That said, SDL_bgi is almost perfectly compatible with the original GRAPHICS.H. It has been tested on the original BGIDEMO.C included in Turbo C 2.01 and Borland C++ 1.01, and on the sample programs available here. These sample programs were copied from the original Borland C Library Reference.

Nearly all functions are correctly implemented and work just like in old BGI; in most cases, output is pixel-perfect.

Differences

Some of the following differences might be eliminated in future releases of SDL_bgi.

- colour names with CGA_ and EGA_ prefix have the same value as standard colours. For example, the EGA_BROWN constant is 6, like BROWN, instead of 20 as in Turbo C or Borland C++. This difference should be irrelevant;
- these functions may be called, but have no effect:
  - _graphfreemem() is unneeded;
  - _graphgetmem() is unneeded;

    - `installuserdriver()` makes no sense in SDL2;
    - `registerbgidriver()` only made sense on the DOS platform;
    - `registerbgifont()` only made sense on the DOS platform;
    - `setaspectratio()` makes no sense on modern hardware;
    - `setgraphbufsize()` is unneeded;

- `initgraph()` always uses the SDL2 graphics driver, regardless of its first parameter;
- functions `registerbgidriver()` and `installuserdriver()` require an argument that must be defined at compile time. For instance, given this code:

```
errorcode = registerbgidriver(EGAVGA_driver);
```

you must add `-D EGAVGA_driver` to the `gcc` command line. You'll get a compiler warning, but the program will compile and run.

- the `size` and `colors` members of `struct palette` are defined as `Uint32` instead of `char`, because colours are implemented as ARGB integers in `SDL_bgi`:

```
struct palettetype {
  Uint32 size;                    // unsigned char in Turbo C / Borland C++
  Uint32 colors[MAXCOLORS + 1];  // signed char in Turbo C / Borland C++
};
```

- `setpalette()` and `setallpalette()` will not change the colours of pixels that have already been drawn. In other words, these functions only affect future drawings (no palette cycling);
- `putimage()` bitwise operations (`XOR_PUT`, `OR_PUT` etc.) are applied to RGB colour components. This is apparently not the same behaviour as in old Turbo C;
- `setusercharsize()` also works with `DEFAULT_FONT`;
- `setrgbpalette()` works on the extended ARGB palette. To change the RGB components of colours in the default palette, use `setpalette()` like this:

```
setpalette (RED, COLOR (0xa0, 0x10, 0x10));
```

Compatibility with WinBGIm

Most extensions introduced by WinBGIm have been implemented, with some differences; WinBGIm, in fact, is written in C++, while `SDL_bgi` is written in C.

When WinBGIm breaks C compatibility with `GRAPHICS.H` by providing C++ extensions, `SDL_bgi` follows the original C syntax.

Differences

- output stream `bgiout` and related functions `outstream()` and `outstreamxy()` are C++ features. Hence, they are not implemented;
- functions `clearmouseclick()`, `converttorgb()`, `printimage()`, `registermousehandler()`, and `setmousequeuestatus()` are not currently implemented;
- mouse functions are simplified in `SDL_bgi`, and do not provide the full range of options available in WinBGIm;
- functions `getwindowheight()` and and `getwindowwidth()` are Windows-specific, and can't be implemented portably;
- function `closegraph()` has no parameters in `SDL_bgi`;
- function `initwindow()` only uses the `width` and `height` parameters in `SDL_bgi`;
- functions `IS_BGI_COLOR()` and `IS_RGB_COLOR()` return a value that depends on the colour being used; their argument is ignored.