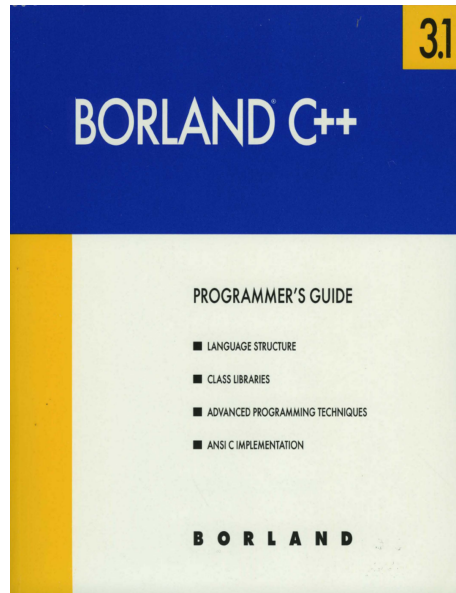


0.1 Programming setup

Development was done with Borland C++ 3.1 using a 80386 computer. John Carmack took care of the runtime code. John Romero programmed many of the tools (TED5 map editor, IGRAB asset packer, MUSE sound packer). Jason Blochowiak wrote important sub-systems of the game (Input manager, Sound manager, User manager).

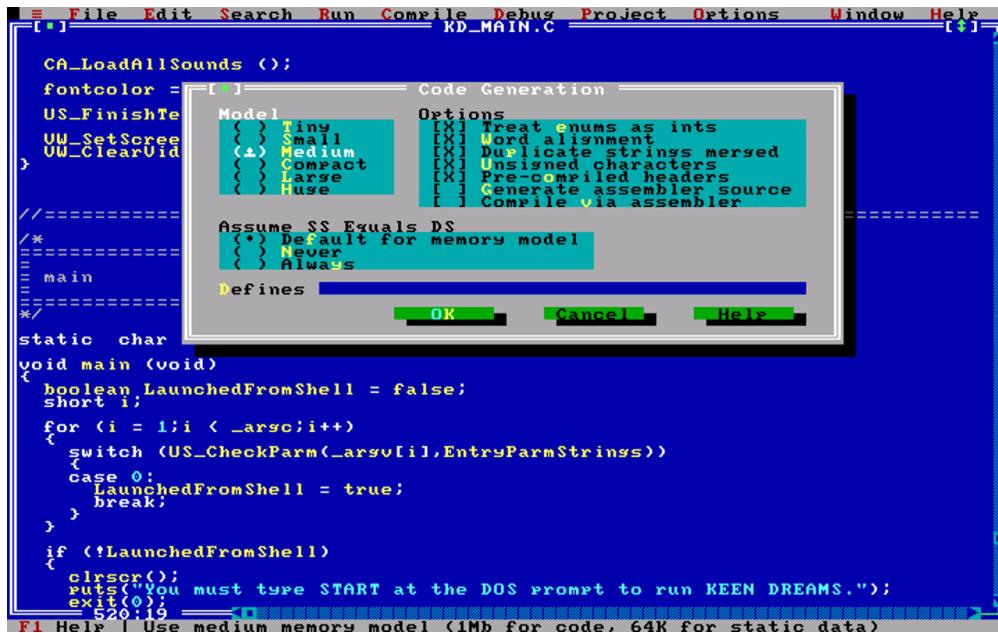
Borland's solution was an all-in-one package. The IDE, BC.EXE, despite some instabilities allowed crude multi-windows code editing with pleasant syntax highlights. The compiler and linker were also part of the package under BCC.EXE and TLINK.EXE. There was no need to enter command-line mode however. The IDE allowed to create a project, build, run and debug. The software came with two thick manuals, explaining everything regarding the IDE (238 pages) and programming in C++ (483 pages).



Borland C++ 3.1 Programmer guide.

0.1. PROGRAMMING SETUP

The IDE allowed to create a project, build, run and debug. It contained a "high resolution" 50x80 text mode, which doubled the standard vertical resolution.



The screenshot shows the IDE with the 'Code Generation' dialog box open. The background is a C source file named 'KD_MAIN.C'. The dialog box has two main sections: 'Model' and 'Options'. The 'Model' section has radio buttons for 'Tiny', 'Small', 'Medium' (selected), 'Compact', 'Large', and 'Huge'. The 'Options' section has checkboxes for 'Treat enums as ints', 'Word alignment', 'Duplicate strings merged', 'Unsigned characters', 'Pre-compiled headers', 'Generate assembler source', and 'Compile via assembler'. Below these is a section for 'Assume SS Equals DS' with radio buttons for 'Default for memory model' (selected), 'Never', and 'Always'. At the bottom of the dialog are 'OK', 'Cancel', and 'Help' buttons. The status bar at the bottom of the IDE reads: 'F1 Help | Use medium memory model (1Mb for code, 64K for static data)'.

```
File Edit Search Run Compile Debug Project Options Window Help
KD_MAIN.C

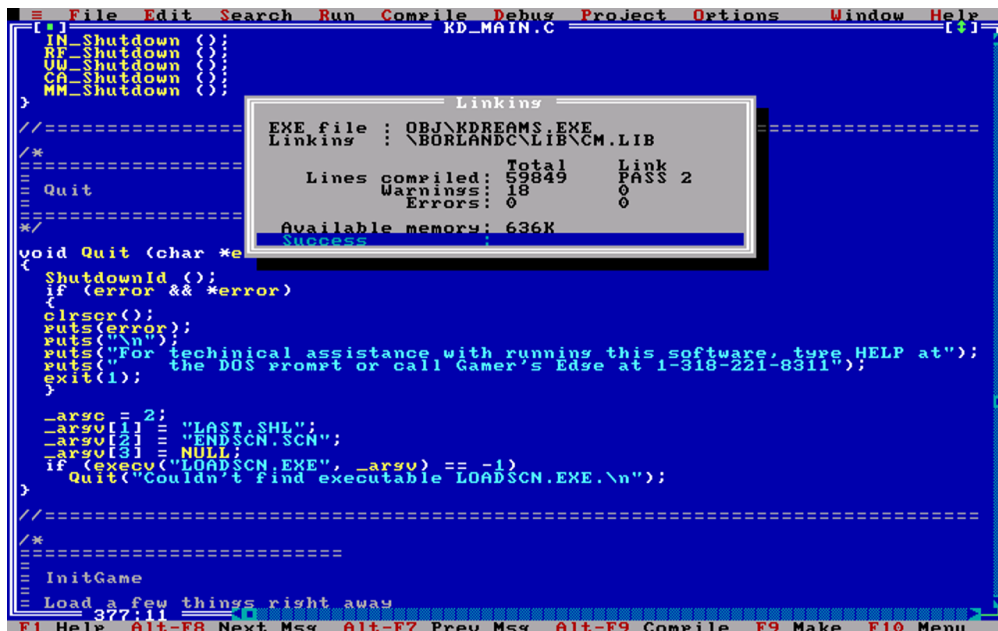
CA_LoadAllSounds (<);
fontcolor = < >
US_FinishTe
UW_SetScree
UW_ClearUId
>

//=====
/*
=====
main
=====
*/

static char
void main (void)
{
    boolean,LaunchedFromShell = false;
    short i;
    for (i = 1;i < _argc;i++)
    {
        switch (US_CheckParm(_argv[i],EntryParmStrings))
        {
            case 0:
                LaunchedFromShell = true;
                break;
        }
    }

    if (!LaunchedFromShell)
    {
        clrscr();
        puts("You must type START at the DOS prompt to run KEEN DREAMS.");
        exit(0);
    }
}

520:19
F1 Help | Use medium memory model (1Mb for code, 64K for static data)
```



The screenshot shows the IDE with the 'Linking' dialog box open. The background is the same C source file 'KD_MAIN.C'. The dialog box shows the 'EXE file' as 'OBJ\KDREAMS.EXE' and the 'Linkins' as '\BORLANDC\LIB\CM.LIB'. It also displays a table of compilation statistics: 'Lines compiled: 59849', 'Warnings: 18', 'Errors: 0', 'Link PASS: 2', and 'Available memory: 636K'. At the bottom of the dialog is a 'Success' message. The status bar at the bottom of the IDE reads: 'F1 Help Alt-F8 Next Mss Alt-F7 Prev Mss Alt-F9 Compile F9 Make F10 Menu'.

```
File Edit Search Run Compile Debug Project Options Window Help
KD_MAIN.C

IN_Shutdown (<);
RF_Shutdown (<);
UW_Shutdown (<);
CA_Shutdown (<);
MM_Shutdown (<);
>

//=====
/*
=====
Quit
=====
*/

void Quit (char *e
{
    ShutdownId (<);
    if (error && *error)
    {
        clrscr();
        puts(error);
        puts("\n");
        puts("For technical assistance with running this software, type HELP at");
        puts("the DOS prompt or call Gamer's Edge at 1-318-221-8311");
        exit(1);
    }

    _argc = 2;
    _argv[1] = "LAST.SHL";
    _argv[2] = "END$CN.SCN";
    _argv[3] = NULL;
    if (execv("LOAD$CN.EXE", _argv) == -1)
    {
        Quit("Couldn't find executable LOAD$CN.EXE.\n");
    }
}

//=====
/*
=====
InitGame
Load a few things right away
377:11
F1 Help Alt-F8 Next Mss Alt-F7 Prev Mss Alt-F9 Compile F9 Make F10 Menu
```

0.2 Graphic Assets

All graphic assets were produced by Adrian Carmack. All of the work was done with Deluxe Paint (by Brent Iverson, Electronic Arts) and saved in ILBM¹ files (Deluxe Paint proprietary format). All assets were hand drawn with a mouse.



Figure 1: Deluxe Paint was used to draw all assets in the game.

0.2.1 Tile planar arrangement

Before diving into the details of game assets, let's first explore the basic principles behind platform games. Each level, or map, in a platform game is composed of "tiles". In Commander Keen, there are both background and foreground tiles. A background tile, or just tile, is an image that measures 16x16 pixels. Each tile occupies 128 bytes (2 x 16 x 4 planes) of storage space in the graphics assets file.

¹InterLeaved BitMap.

Individual tile images are stored in a planar format, with blue, green, red and intensity bits separated. This arrangement, called "graphic planar", stores complete planes sequentially, with each plane containing data for the entire tile.

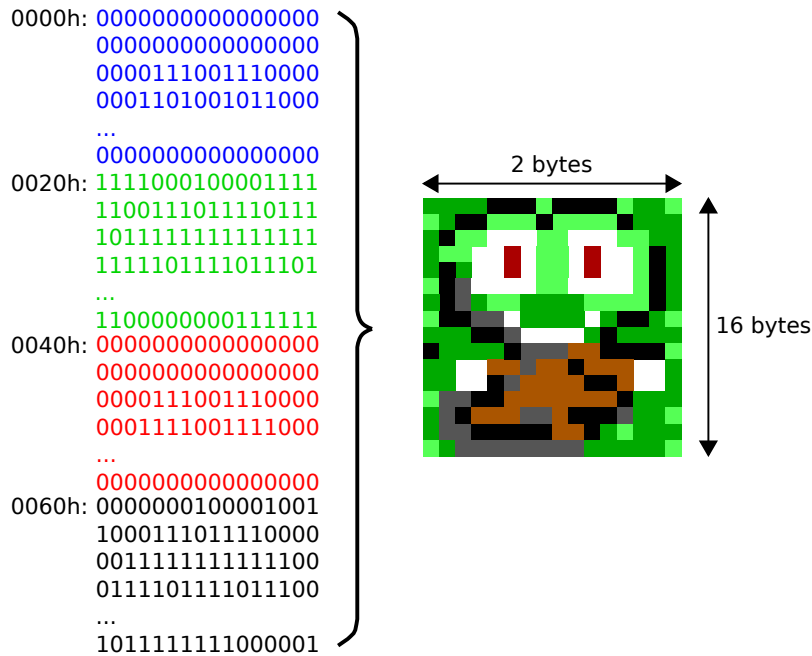


Figure 2: Graphic Planar data storage.

Foreground tiles, or "masked tiles", are similar to normal tiles but contain five planes instead of four. The extra plane stores the mask, and the order of planes is mask, blue, green, red, intensity. Consequently, a single masked tile requires 160 bytes (128 + 32) of storage. A mask bit of '0' means the background tile color is erased and replaced by the foreground tile color, while a mask bit of '1' blends the background color with the foreground color. If the foreground color is '0', the background color remains visible.

By combining tiles on the screen, a map is created. These maps define the entire game world in terms of background and foreground tiles. Maps also contain a list of all actors and their starting positions within the world. Essentially, everything the player encounters while progressing through the levels is specified in a map.

0.2.2 Assets Workflow

After the graphic assets were generated, a tool (IGRAB) packed all ILBM files into a compressed data asset file (EGA-file) and generated a HEAD table, DICT file (KDR-files²), and a C header file containing asset IDs. The engine references an asset directly using these IDs.

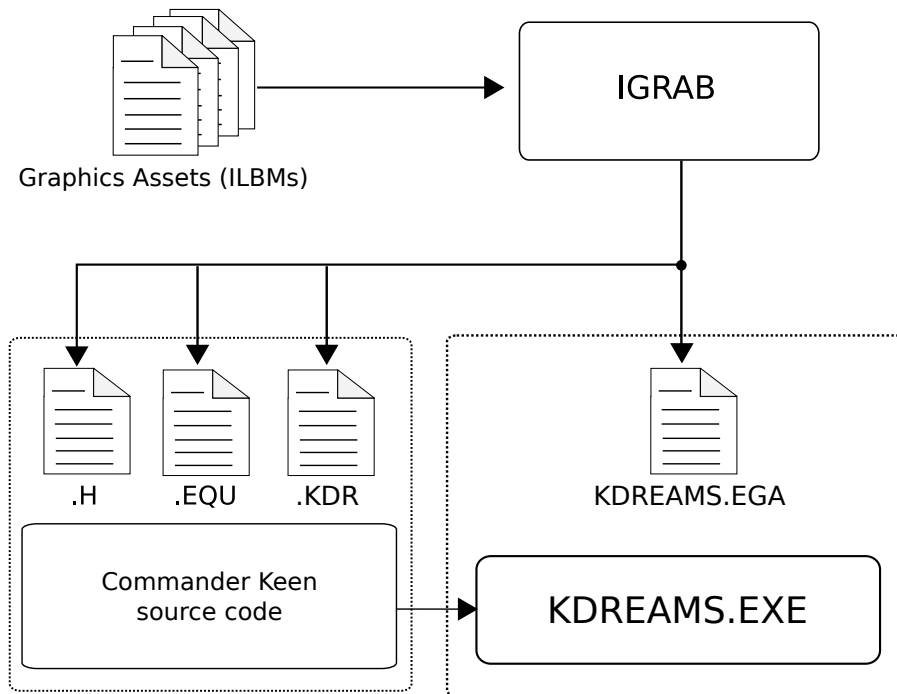


Figure 3: Asset creation pipeline for graphics items

In the engine's code, asset usage is hardcoded via an enum. This enum serves as an offset into the HEAD table, which then provides an offset within the data asset file. With this indirection layer, assets could be regenerated and reordered at will with no modification in the source code.

Trivia : The HEAD and DICT files in the source code must match the data asset file from the shareware version of *Keen Dreams*. However, the latest release of the source code (v1.93) does not match with shareware version v1.13. To ensure that the HEAD, DICT, and data asset file are compatible, you will need to retrieve a specific git commit, which will be explained in the next chapter.

²both KDR-files are located in the `static` folder of the source code.

```
////////////////////////////////////
//
// Graphics .H file for .KDR
// IGRAB-ed on Fri Sep 10 11:18:07 1993
//
////////////////////////////////////

#define CTL_STARTUPPIC          4
#define CTL_HELPU PIC          5
#define CTL_DISKUPPIC          6
#define CTL_CONTROLSUPPIC      7
#define CTL_SOUNDUPPIC         8
#define CTL_MUSICUPPIC         9
#define CTL_STARTDNPIC        10
#define CTL_HELPDNPIC         11
#define CTL_DISKDNPIC         12
#define CTL_CONTROLSDNPIC     13
...

#define CURSORARROWSPR        71
#define KEENSTANDRSPR         72
#define KEENRUNR1SPR          73
#define KEENRUNR2SPR          74
#define KEENRUNR3SPR          75
#define KEENRUNR4SPR          76
...

//
// Data LUMPs
//
#define CONTROLS_LUMP_START    4
#define CONTROLS_LUMP_END      68
#define KEEN_LUMP_START        72
#define KEEN_LUMP_END          212
#define WORLDKEEN_LUMP_START   213
#define WORLDKEEN_LUMP_END     240
#define BROCCOLASH_LUMP_START  241
#define BROCCOLASH_LUMP_END    256
#define TOMATO_LUMP_START      257
#define TOMATO_LUMP_END        260
...
```

0.2.3 Assets file structure

Figure 4 illustrates the structure of the `KDREAMS.EGA` asset file. The first section contains data tables for pictures and sprites, followed by the font, and all sprite and tile graphs.

| | |
|---------------|----------------------------|
| pictable[] | STRUCTPIC |
| picmtable[] | STRUCTPICM |
| spritetable[] | STRUCTSPRITE |
| font | STARTFONT |
| pictures | STARTPICS |
| mask pictures | STARTPICSM STARTSPRITES |
| sprites | |
| tile-8 | STARTTILE8 |
| mask tile-8 | STARTTILE8M STARTTILE16 |
| tile-16 | |
| mask tile-16 | STARTTILE16M |

Figure 4: File structure of `KDREAMS.EGA` archive file.

Both the `pictable[]` and `picmtable[]` contain the width and height for each picture in the asset file. The `spritemap[]` includes not only width and height but also information on the sprite's center, hitbox, and number of bit shifts (explained section "??" on page ??).

```
typedef struct
{
    int width,
    height,
    orgx,orgy,
    xl,yl,xh,yh,
    shifts;
} spritetabletype;
```

All sprite placement occurs from the origin, which is offset by (`orgx`, `orgy`) from the sprite's top-left corner. The parameters (`xl`, `xh`, `yl`, `yh`) define the sprite's hitbox, used for collision detection.

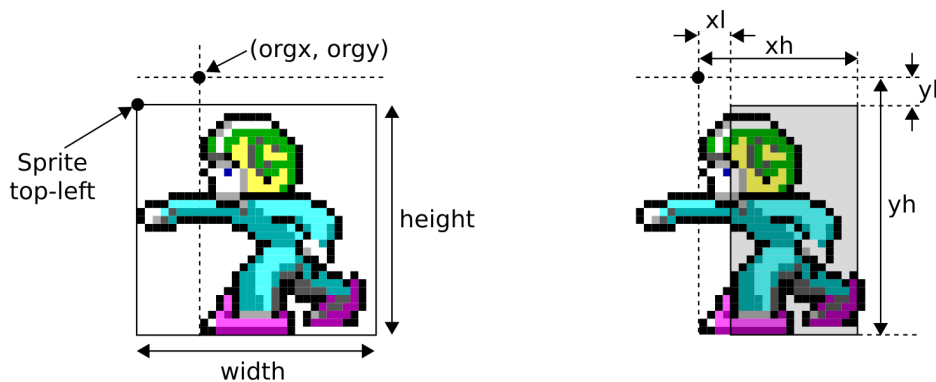


Figure 5: Sprite dimensions, origin and hitbox.

The font segment contains a table that stores the height and the width of the each character, along with a reference to where the character data is located in the archive file.

```
typedef struct
{
    int height;
    int location[256];
    char width[256];
} fontstruct;
```

The remainder of the archive file holds all graphic assets, including pictures, sprites, and

tiles. Each asset contains four planes, aligned with the EGA architecture. All masked tiles and sprites also include an additional mask plane.

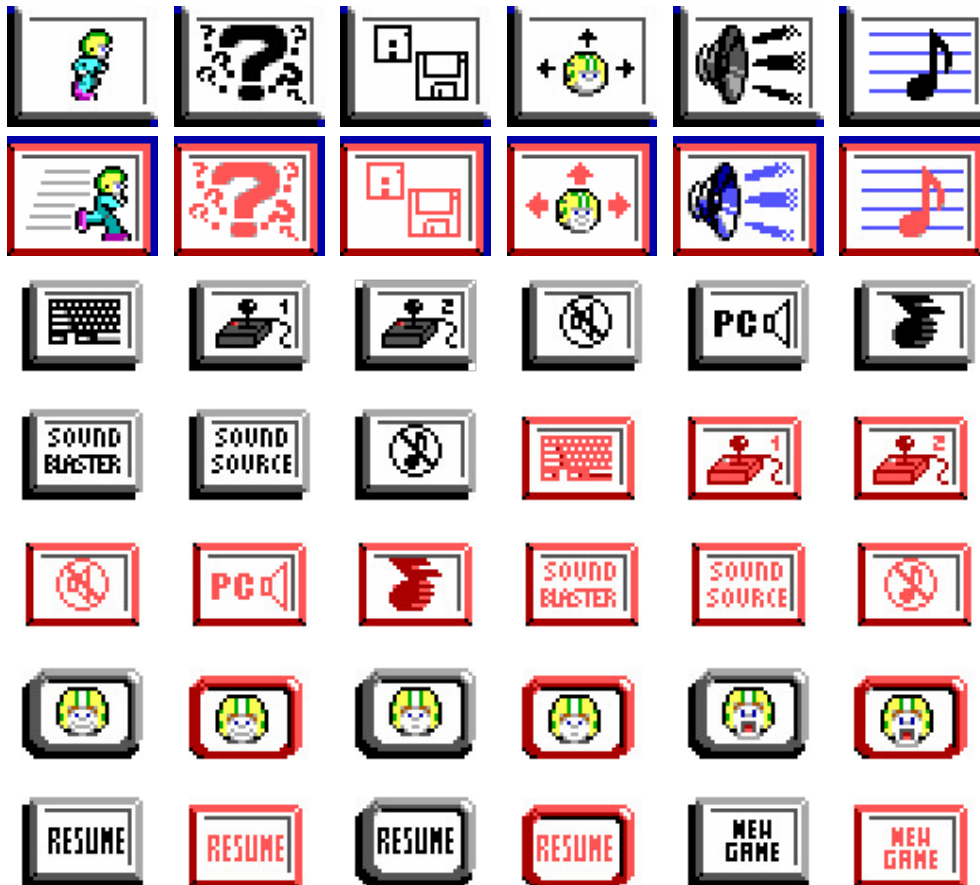


Figure 6: Picture assets.

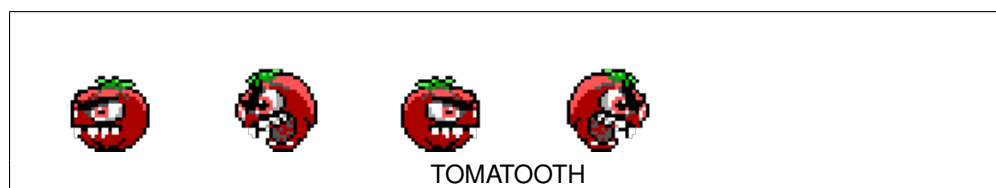
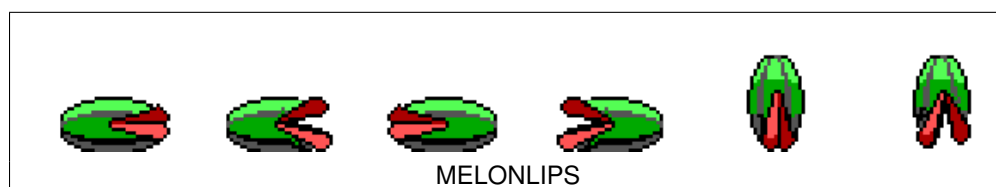
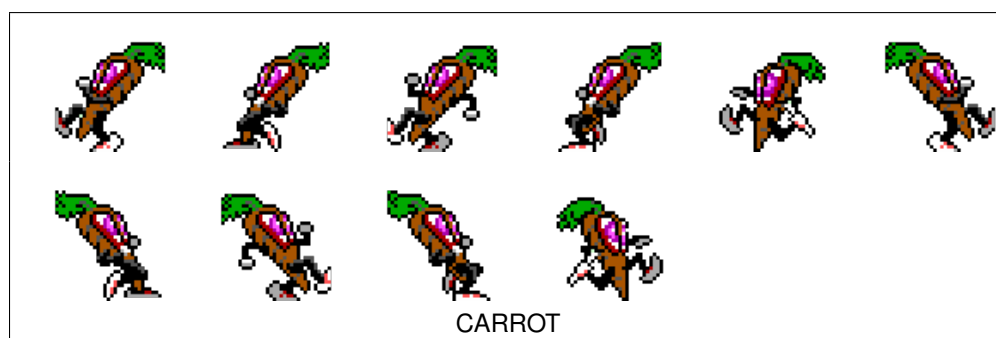
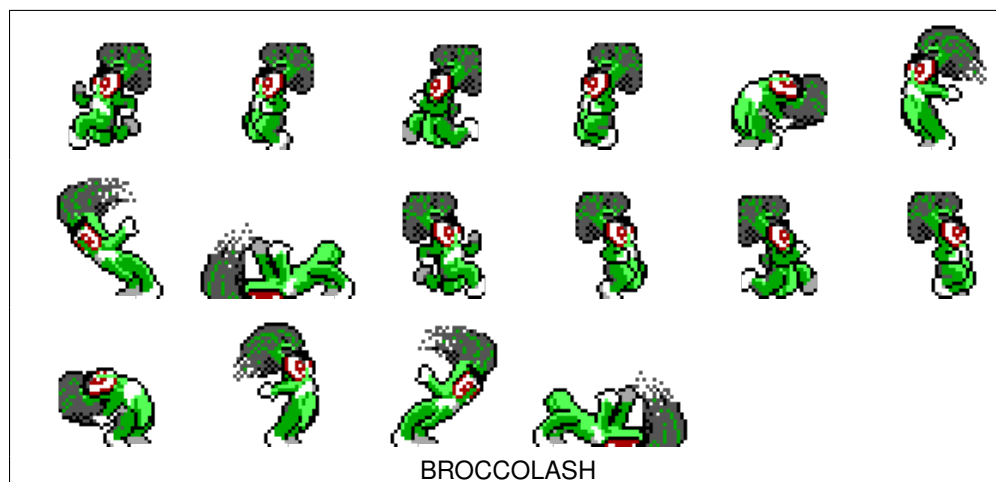


Figure 7: Sprite assets.

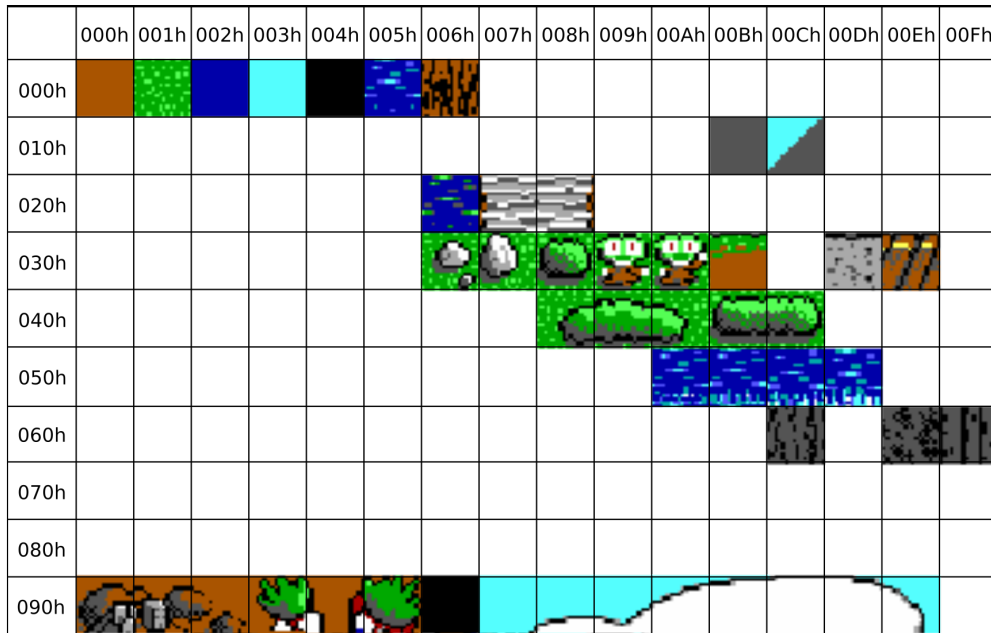


Figure 8: Background (Tile-16) and foreground (masked Tile-16) assets.

0.3 Maps

Maps were created using an in-house editor called TED5, short for Tile EDitor. Over the years TED5 had improvements and the same tool is later used for creating maps for both side-scrolling games and top-down games like *Wolfenstein 3D*.

TED5 is not stand-alone; in order to start, it needs an asset archive and the associated header (as described in Figure 3 on page 5). This way, tile IDs are directly encoded in the map.



TED5 allows the placement of tiles across multiple layers, referred to as "planes". In Commander Keen, there are three types of planes:

- Background plane tiles.
- Foreground plane tiles, which act as a mask over the background plane.
- Information plane tiles, which contain actor locations and special areas.

A level is created by placing tiles on each of these three layers. Foreground and background tiles can be enhanced with additional properties ("Tile info"), which controls interactions such as clipping, "deadly" tiles, and animated tiles.

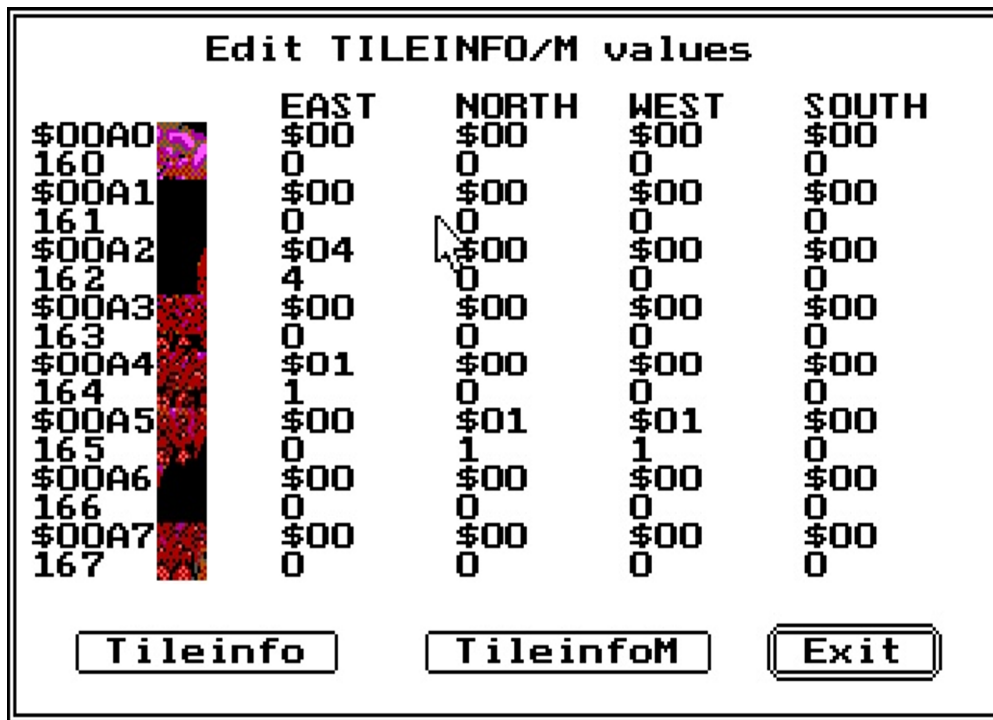


Figure 9: "Tile info" adding properties to tiles.

Similar to IGRAB, TED5 compresses all levels into a MAP archive and generates corresponding HEAD and DICT files.

0.3.1 Map header structure

The map header structure is embedded in the engine code and contains a header offset and size, which refer to the location and size within the KDREAMS.MAP archive file. The game supports a maximum of 100 maps. The `tileinfo[]` array contains properties data for each tile.

```
typedef struct
{
    unsigned    RLEWtag;                // RLE flag
    long        headeroffsets[100];
    byte        headersize[100];        // headers are very small
    byte        tileinfo[];
} mapfiletype;
```

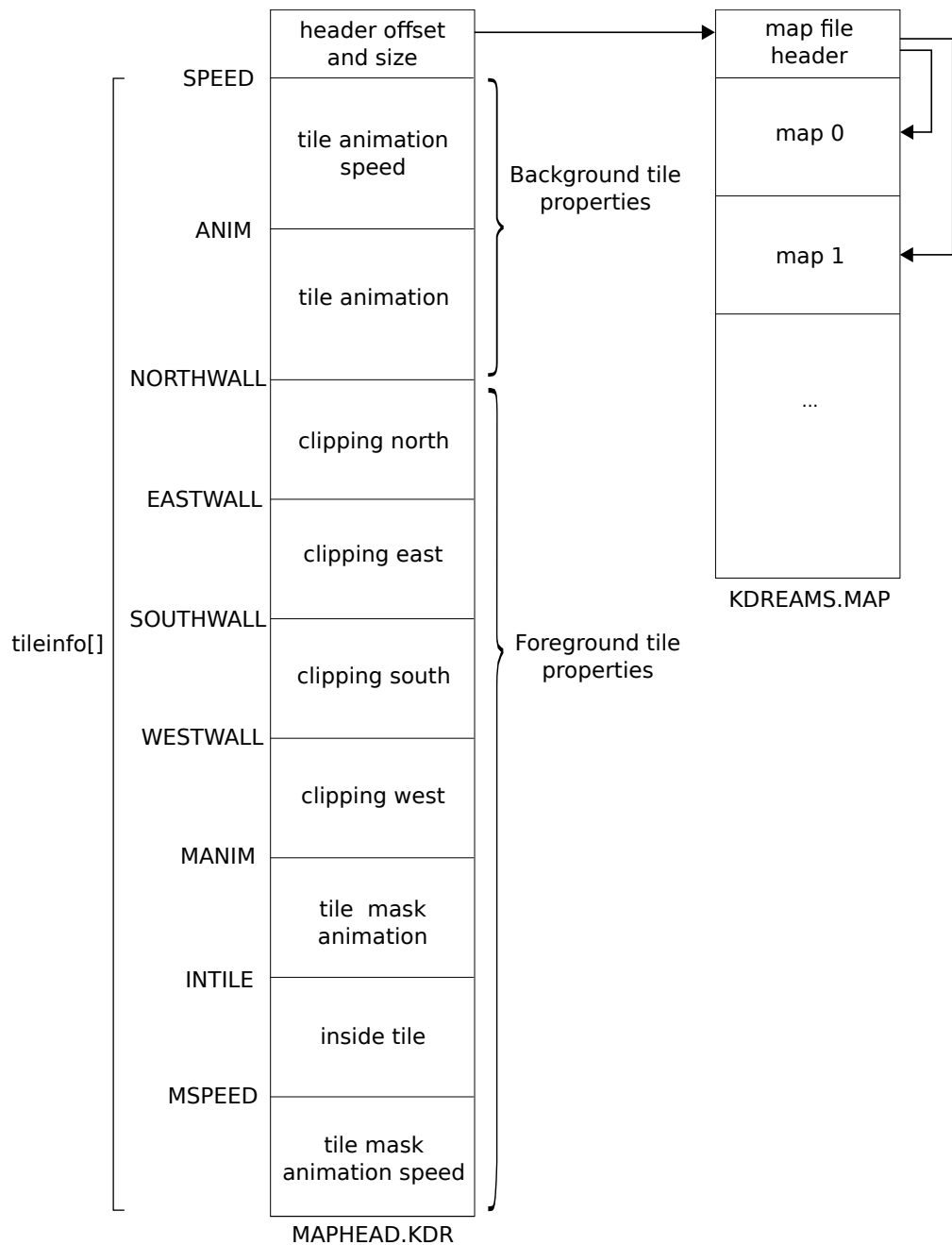


Figure 10: Structure of `MAPHEAD.KDR` header file.

For background tile animations, two information tables are defined: "tile animation" and "tile animation speed". The tile animation specifies the next tile in the animation sequence, relative to the current tile. For example, in Table 1, tile #90 animates to tile #91 (+1), followed by #92 (+1), and #93 (+1). After tile #93, the sequence returns to tile #90 (-3). The animation speed is expressed in number of ticks before the next tile is displayed.

| tile # | tile animation | tile animation speed |
|--------|----------------|----------------------|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| ... | ... | ... |
| 57 | 1 | 32 |
| 58 | -1 | 24 |
| ... | ... | ... |
| 90 | 1 | 8 |
| 91 | 1 | 8 |
| 92 | 1 | 8 |
| 93 | -3 | 8 |
| ... | ... | ... |

Table 1: Background tile animation.

Tiles also contain clipping information and a column called `intile`. The `intile` column tells what the tile "does" to the player, like kill, climb, points, etc. Values between 128-255 are used for foreground tiles only. The `intile` column is customized for each version of *Commander Keen*. In *Keen Dreams* it is used for e.g. climbing poles.

| tile # | clip north | clip east | clip south | clip west | intile |
|--------|------------|-----------|------------|-----------|--------|
| ... | ... | ... | ... | ... | ... |
| 238 | 0 | 1 | 5 | 0 | 0 |
| 239 | 0 | 0 | 0 | 0 | 0 |
| 240 | 0 | 0 | 5 | 0 | 0 |
| 241 | 0 | 0 | 0 | 0 | 128 |
| 242 | 1 | 1 | 1 | 1 | 128 |
| 243 | 1 | 0 | 1 | 0 | 0 |
| 244 | 0 | 0 | 2 | 0 | 0 |
| ... | ... | ... | ... | ... | ... |

Table 2: Foreground tile clipping and 'intile' tile information.

0.3.2 Map archive structure

The structure of the `KDREAMS.MAP` archive is illustrated in Figure 11. Each map contains a small header that includes the width, height, and name of the map, as well as a reference pointer to each of the three planes. Each plane consists of a map of tile numbers representing the background, foreground, and information planes.

```
typedef struct
{
    long        planestart[3];
    unsigned    planelength[3];
    unsigned    width,height;
    char        name[16];
} maptype;
```

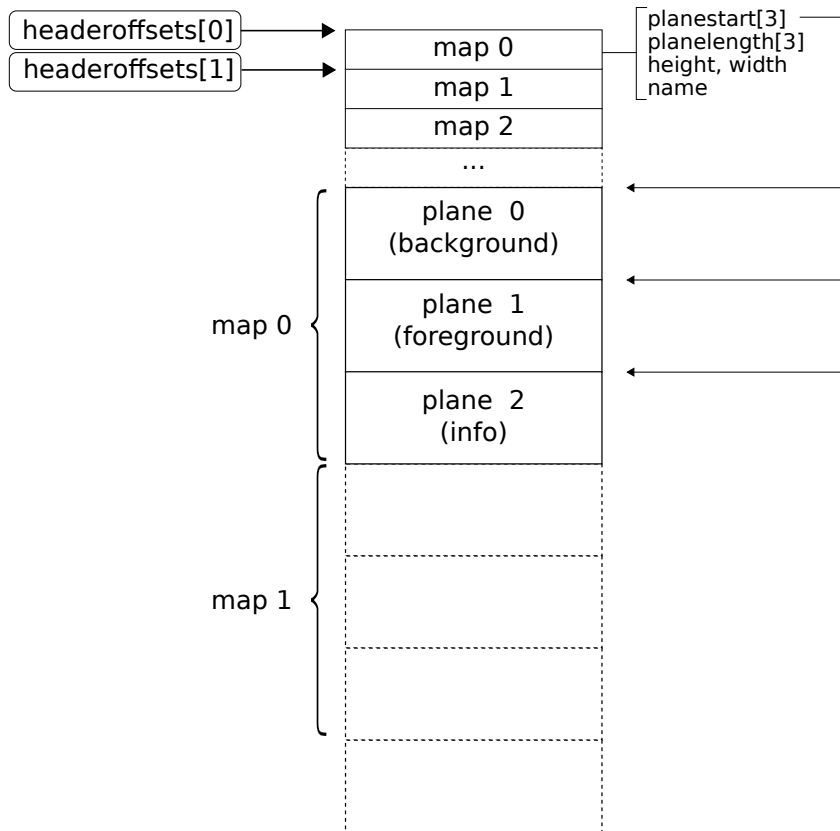


Figure 11: Structure of `KDREAMS.MAP` file.

0.4 Audio

The original Commander Keen Trilogy did only support the default PC Speaker. With the introduction of Commander Keen in Keen Dreams the team decided to support sound cards as well.

Trivia : Apogee, the publisher of Ideas from the Deep, didn't publish any game with AdLib support until *Dark Ages* in 1991. And even then it still had pc speaker music.

id Software intended for Keen Dreams to have music and digital effects support for the SoundBlaster & Sound Source devices. In fact, Bobby Prince composed the song "You've Got to Eat Your Vegetables!!" for the game's introduction. However, Softdisk Publishing wanted Keen Dreams to fit on a single 360K floppy disk, and in order to do this, id Software had to scrap the game's music at the last minute³. The team didn't even have time to remove the music setup menu.



Figure 12: Setup music menu in Keen Dreams, although there is no music in the game.

³<https://vimeo.com/4022128>, at 2:00-4:55.

As a consequence, only two sets of each audio effect are shipped with the game:

1. For PC Speaker
2. For AdLib



Figure 13: Both Sound Blaster and Sound System are disabled.

Trivia : The song "You've Got to Eat Your Vegetables!!", written for *Keen Dreams*, would finally make its debut in *Commander Keen IV: Secret of the Oracle*.

All sound effects are done by Robert Prince. Sound effects are created using Cakewalk and inhouse MUSE tool, which is extensively described in *Wolfenstein 3D Blackbook*⁴ Just like described before, the MUSE tool packed all sound effects together in a compressed SOUND archive and generated a HEAD, DICT file and a C header file with asset IDs.

⁴See *Wolfenstein 3D Blackbook*, section 3.6.

0.5 Distribution

On December 14th, 1990 the first episode was released via Apogee. Episodes 1-5 are all published by Apogee Software. The game engine and first episode were given for free and encourages to be copied and distributed to a maximum number of people. To receive the other episodes, each player had to pay \$30 (for Episode 1-3) to *ideas from the Deep*.

Commander Keen in Keen Dreams was published as a retail title by Softdisk, as part of a settlement for using Softdisk resources to make their own game⁵.

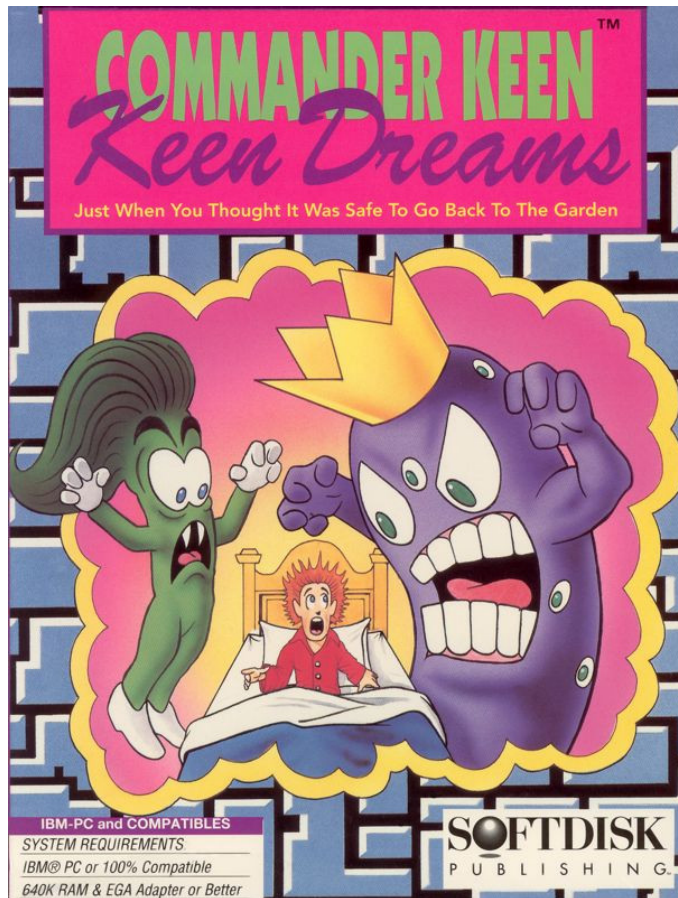


Figure 14: Retail version of Commander Keen in Keen Dreams by Softdisk.

⁵The settlement with Softdisk is explained in Appendix ??

In 1990, the Internet was still in its infancy and the best medium was the 3½-inch floppy disk. The shipped game can be divided in seven parts:

- KDREAMS.EXE: Game engine.
- KDREAMS.EGA: Contains all the assets (sprites, tiles) needed during the game.
- KDREAMS.AUD: Sound effect files.
- KDREAMS.MAP: Contains all level layouts.
- KDREAMS.CMP: Introduction picture of the game, a compressed LBM image file.
- Softdisk Help Library files, which are text screens, shown when starting and ending the game
- Several *.TXT files which can be read in DOS by typing the corresponding *.BAT file.

```
Directory of C:\KDREAMS\.  
.  
..  
BKGND SHL 285 16-05-2023 20:59  
FILE_ID DIZ 508 16-05-2023 20:59  
HELP BAT 34 16-05-2023 20:59  
HELPINFO TXT 1,038 16-05-2023 20:59  
INSTRUCT SHL 2,763 16-05-2023 20:59  
KDREAMS AUD 3,498 16-05-2023 20:59  
KDREAMS CFG 656 16-05-2023 21:00  
KDREAMS CMP 14,189 16-05-2023 20:59  
KDREAMS EGA 213,045 16-05-2023 20:59  
KDREAMS EXE 354,691 04-05-2023 19:40  
KDREAMS MAP 65,673 16-05-2023 20:59  
LAST SHL 1,634 16-05-2023 20:59  
LICENSE DOC 8,347 16-05-2023 20:59  
LOADSCN EXE 9,959 16-05-2023 20:59  
MENU SHL 447 16-05-2023 20:59  
NAME SHL 21 16-05-2023 20:59  
ORDER SHL 1,407 16-05-2023 20:59  
PRODUCTS SHL 4,629 16-05-2023 20:59  
QUICK SHL 3,211 16-05-2023 20:59  
README TXT 1,714 16-05-2023 20:59  
START EXE 17,446 16-05-2023 20:59  
VENDOR BAT 32 16-05-2023 20:59  
VENDOR DOC 11,593 16-05-2023 20:59  
VENDOR TXT 810 16-05-2023 20:59  
24 File(s) 717,630 Bytes.  
2 Dir(s) 262,111,744 Bytes free.  
C:\KDREAMS>
```

Figure 15: All Keen Dreams files as they appear in DOS command prompt.