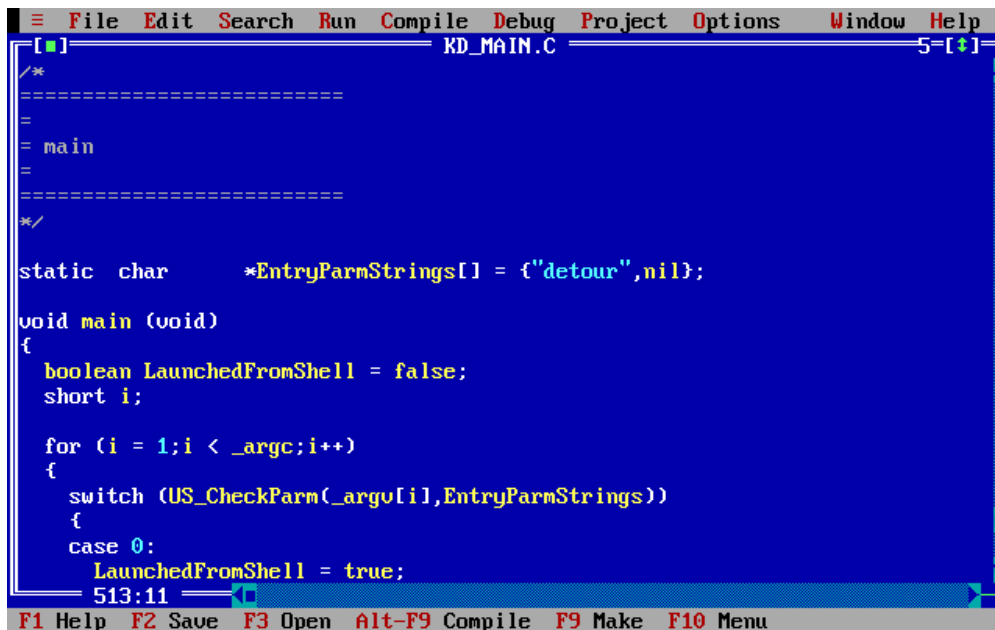


## 0.1 Programming

Development was done with Borland C++ 3.1 (but the language used was C) which by default ran in EGA mode 3 offering a screen 80 characters wide and 25 characters tall.

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 subsystems 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<sup>1</sup>.



```

[ ] KD_MAIN.C 5-[ ]
/*
=====
=
= main
=
=====
*/

static char    *EntryParmStrings[] = {"detour",nil};

void main (void)
{
    boolean LaunchedFromShell = false;
    short i;

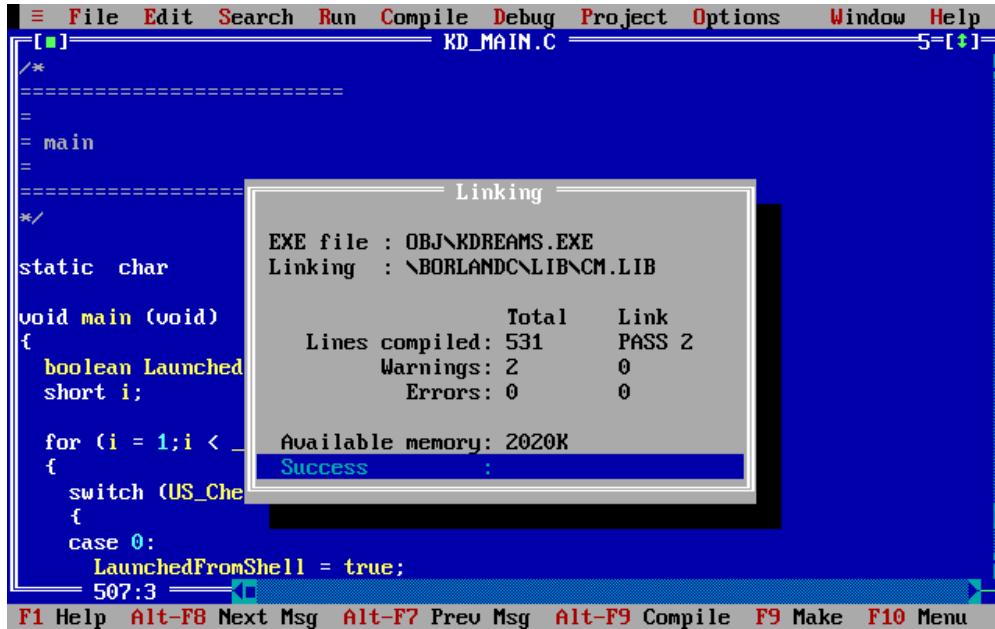
    for (i = 1;i < _argc;i++)
    {
        switch (US_CheckParm(_argv[i],EntryParmStrings))
        {
            case 0:
                LaunchedFromShell = true;
        }
    }
}
513:11
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```

**Figure 1:** Borland C++ 3.1 editor

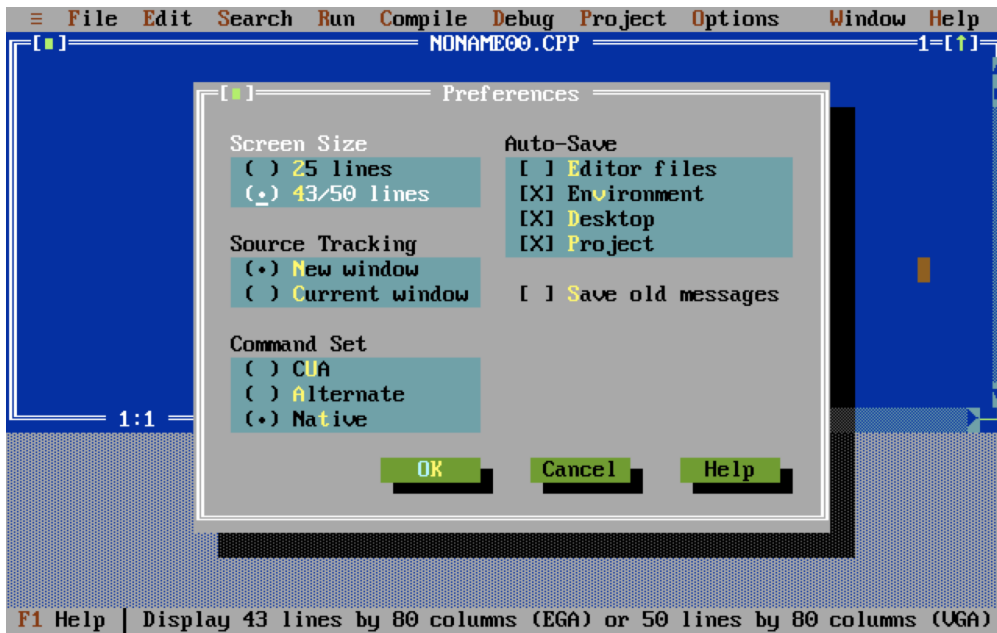
<sup>1</sup>Source: Borland C++ 3.1 User Guide.

There was no need to enter command-line mode however. The IDE allowed to create a project, build, run and debug.

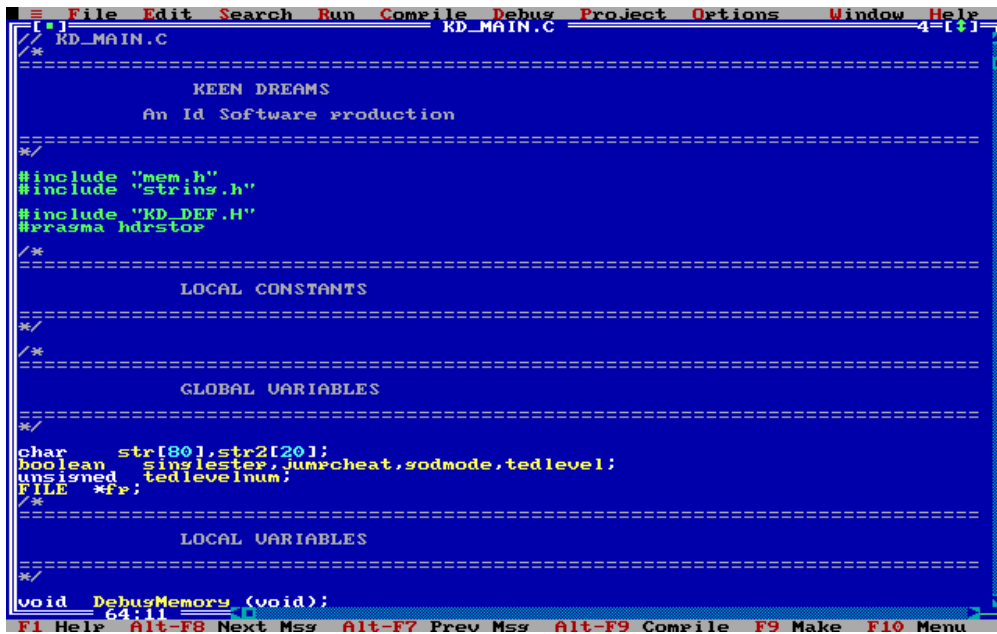


**Figure 2:** Compiling Keen Dreams with Borland C++ 3.1

Another way to improve screen real estate was to use "high resolution" 50x80 text mode.



The comments still fit perfectly on screen since only the vertical resolution is doubled.



The file KD\_MAIN.C opened in both modes demonstrates the readability/visibility trade-off.

```

File Edit Search Run Compile Debug Project Options Window Help
KD_MAIN.C 5-[+]
/*
=====
=
= main
=
=====
*/

static char    *EntryParmStrings[] = {"detour",nil};

void main (void)
{
    boolean LaunchedFromShell = false;
    short i;

    for (i = 1;i < _argc;i++)
    {
        switch (US_CheckParm(_argv[i],EntryParmStrings))
        {
            case 0:
                LaunchedFromShell = true;
        }
    }
}
513:11
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```

```

File Edit Search Run Compile Debug Project Options Window Help
KD_MAIN.C 4-[+]
/*
=====
=
= main
=
=====
*/

static char    *EntryParmStrings[] = {"detour",nil};

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);
    }

    InitGame();
    DemoLoop(); // DemoLoop calls Quit when everything is done
    Quit("Demo loop exited???");
}
-
530:4
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg 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 ILBM<sup>2</sup> files (Deluxe Paint proprietary format). All assets were hand drawn with a mouse.

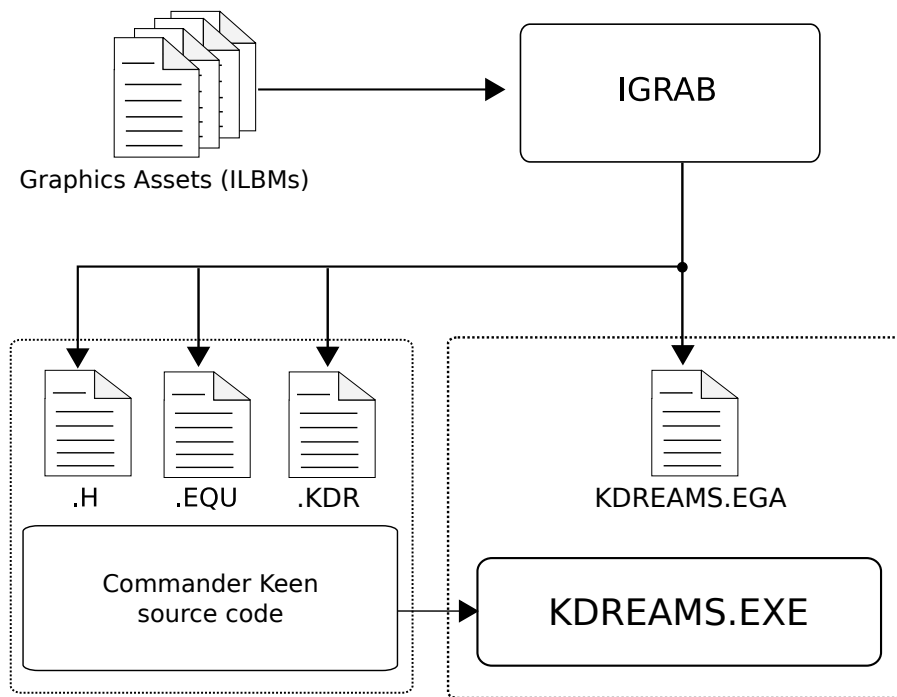


**Figure 3:** Deluxe Paint was used to draw all assets in the game.

### 0.2.1 Assets Workflow

After the graphic assets were generated, a tool (IGRAB) packed all ILBMs together in an archive and generated a header table file (KDR-format) and C header file with asset IDs. The engine references an asset directly by using these IDs.

<sup>2</sup>InterLeaved BitMap.



**Figure 4:** Asset creation pipeline for graphics items

```

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

typedef enum {
    #define CTL_STARTUPPIC          4
    #define CTL_HELPUPPIC          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 BOOBUSWALKR4SPR        366
    #define BOOBUSJUMPSPR         367
}

```

In the engine code, asset usage is hardcoded via an enum. This enum is an offset into the HEAD table which contains an offset in the DATA archive. The HEAD table files are stored in the \static folder as \*.KDR files.

## 0.2.2 Assets file structure

Figure 5 shows the structure of the KDREAMS.EGA asset file.

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

**Figure 5:** File structure of KDREAMS . EGA asset file.



The `pictable[]` contains the width and height in bytes for each picture in the asset file. Note that a width of 5 bytes means a width of 40 pixels on the screen. The same size structure is applied for mask pictures.

index	width	height
0	5	32
1	5	32
2	5	32
3	5	32
4	5	32
5	5	32
6	5	32
7	5	32
...	...	...
64	5	24

**Table 1:** content of `pictable[]`.

The `spritetable[]` contains, beside width and height, also information on the sprite center, hit boundaries and number of shifted sprites, which will be explained later (section ?? on page ??).

The font segment contains a table for the height (same for all characters) and width of the font, as well as a reference where the character data is located.

```
// ID_VW.H

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





Figure 7: Picture asset data.

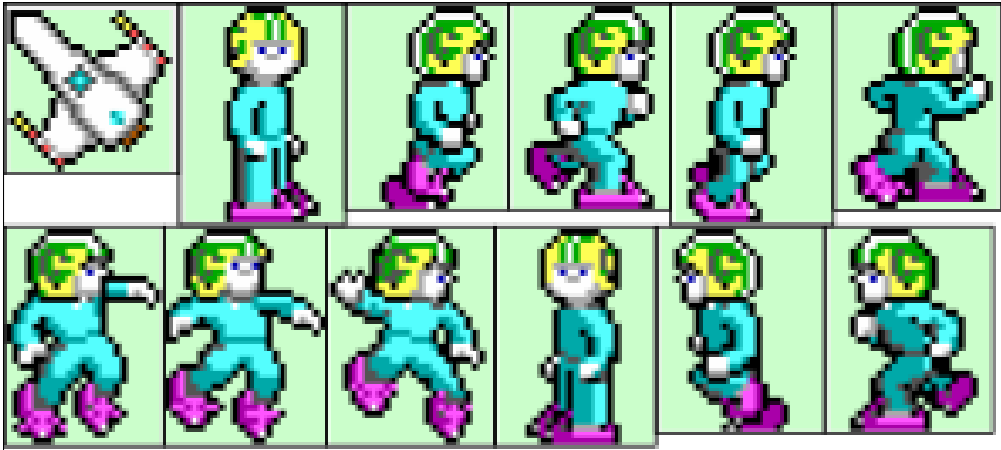


Figure 8: Sprite asset data.

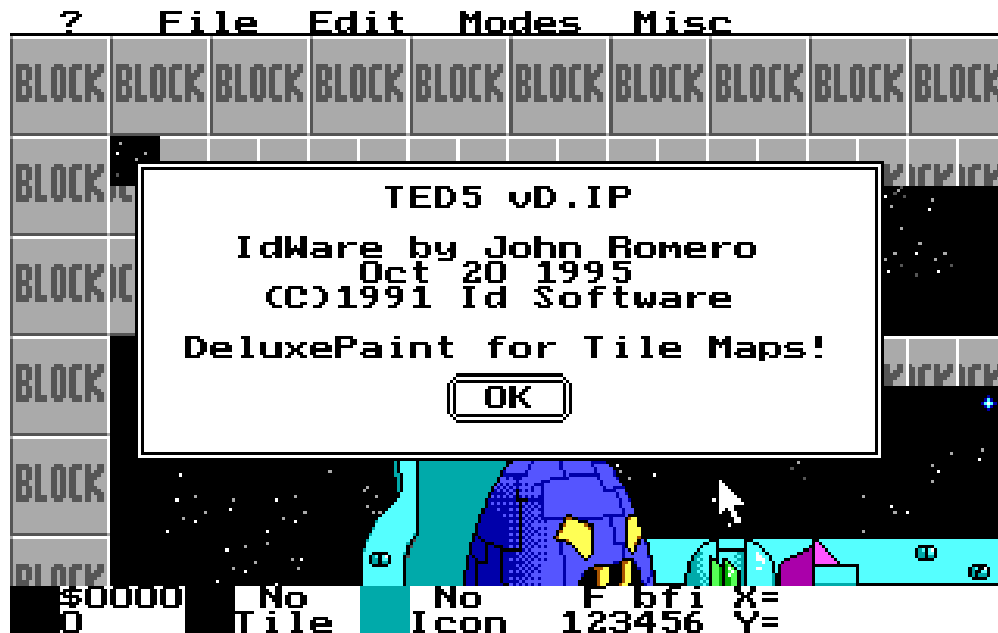


**Figure 9:** Background (Tile16) and foreground (masked Tile16) assets data.

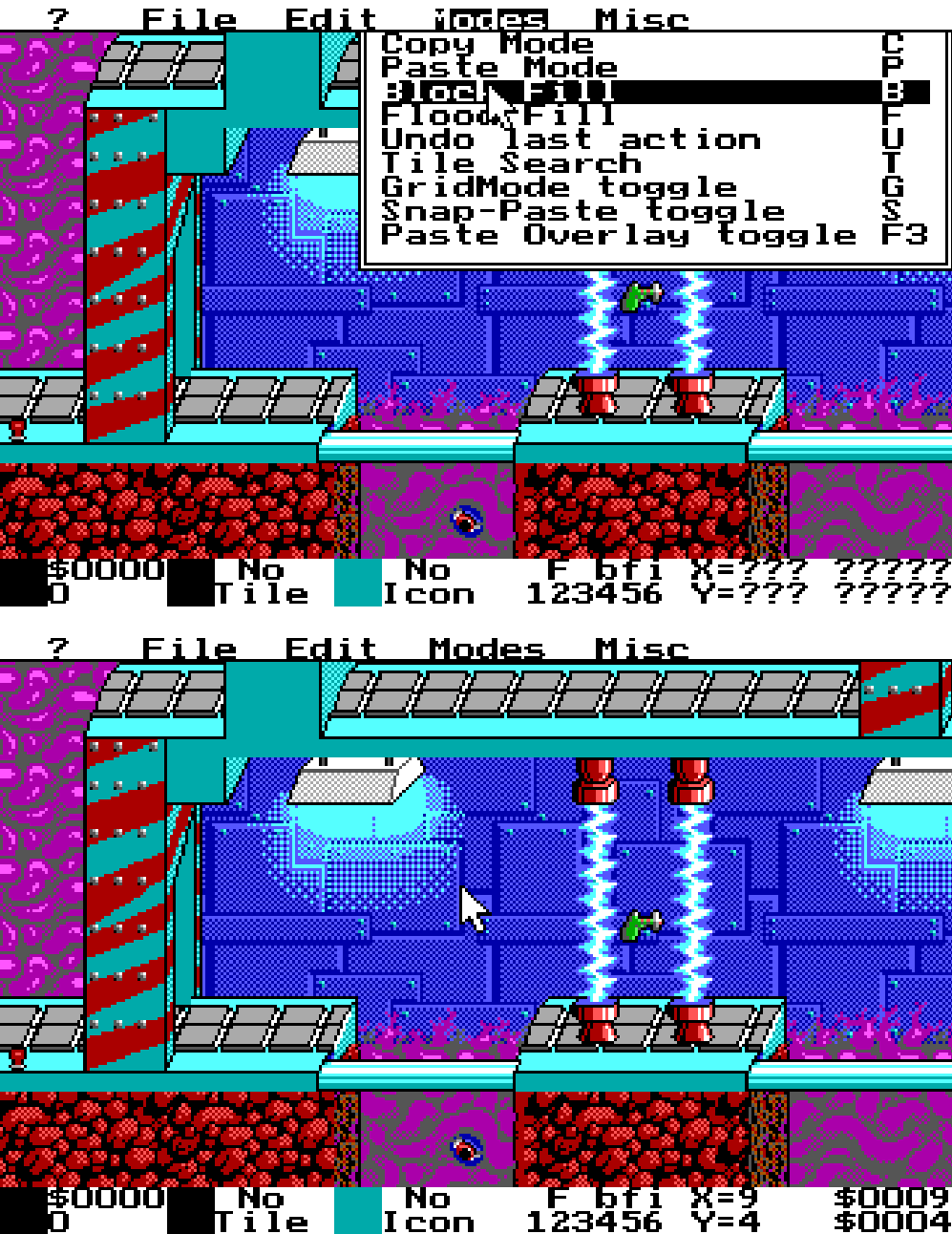
## 0.3 Maps

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 of 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 the graphic asset workflow Figure 4 on page 6). This way, texture IDs are directly encoded in the map.

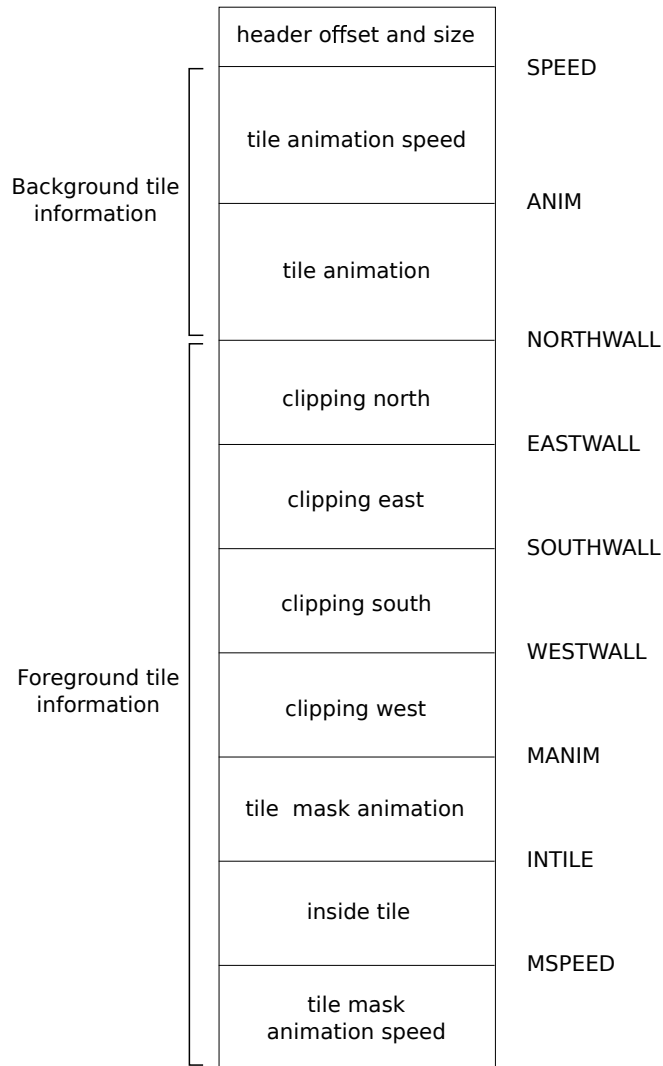


**Trivia :** The suffix, "vD.IP", was put in by the Rise of the Triad team in 1994. It stood for "Developers of Incredible Power".



TED5 allows placement of tiles on layers called "planes". In Commander Keen, layers are used for background, foreground and information planes. Note that foreground tiles are

always using a mask as they are overlayed with the background. The info plane contains the location of actors and special places. Each foreground and background tile could also be enriched with additional tile information such as tile clipping and animated tiles. Just like IGRAB, the TED5 tool generated a header table file (KDR-format) and a \*.MAP file containing the levels. Figure 10 shows the map header table structure, which is hard-coded in the source code.



**Figure 10:** File structure of MAPHEAD.KDR header file.

#### 0.3.1 Map header structure

The header offset and header size refer to the location and size in the `KDREAMS.MAP` file. A maximum of 100 maps is supported in the game.

```
/*
=====

                LOCAL CONSTANTS

=====
*/

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

#### 0.3.2 Background tile information

For background tile animation two information tables are required: tile animation and tile animation speed. The tile animation refers to the next tile in the animation sequence. So in case of tile #90 (see Table 2), the next animation tile is #91 (+1), followed by #92 (+1) and #93 (+1). After tile #93 (-3) the sequence is going back to tile #90. The animation speed is expressed in TimeCount, which is the number of ticks before the next tile is displayed.



index	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 2:** Background tile animation.

### 0.3.3 Foreground tile information

The foreground tiles contain, beside tile animation (similar like background tiles), also clipping and 'inside' tile information.

The clipping tables contain how Commander Keen is clipped against foreground tiles. 'Inside' tile information is used to climb on a pole and mimics Commander Keen going through a floor opening ('inside' the tile). In section ?? (see page ??) both the clipping and 'inside' logic is further explained.

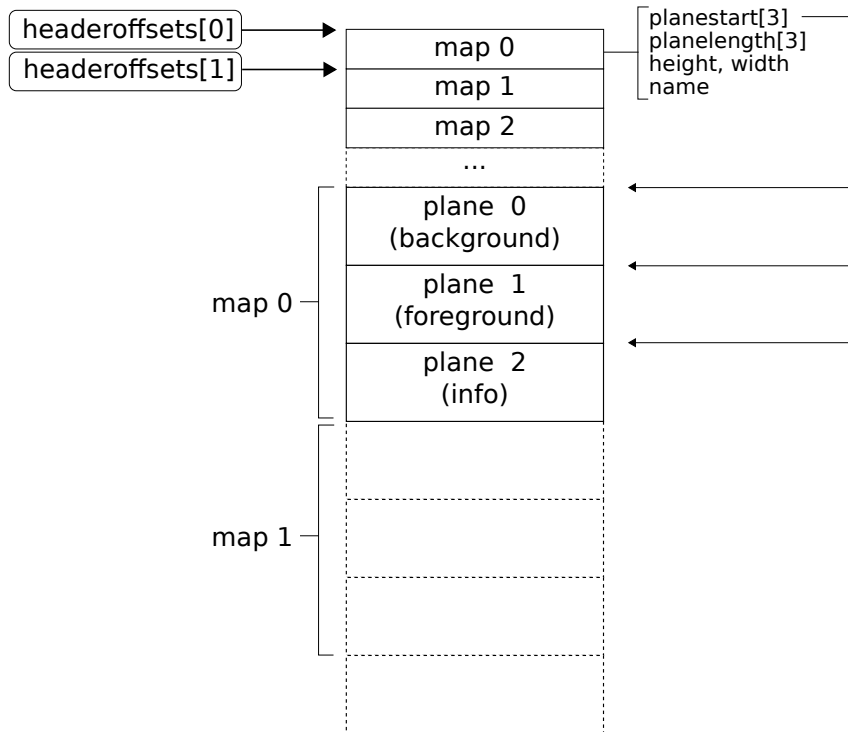
index	clip north	clip east	clip south	clip west	inside tile
...	...	...	...	...	...
238	0	1	5	0	0
239	0	0	0	0	0
240	0	0	5	0	0
241	0	0	0	0	128
242	1	1	1	1	128
243	1	0	1	0	0
244	0	0	2	0	0
...	...	...	...	...	...

**Table 3:** Foreground tile clipping and 'inside' tile information.

### 0.3.4 Map file structure

The structure of `KDREAMS.MAP` is explained in Figure 11. For each map there is a small header containing the width, height and name of the map as well as a reference pointer to each of the three planes. Each plane exists out of a map of tile numbers for foreground, background and info.

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



**Figure 11:** File structure of `KDREAMS.MAP` file.

## 0.4 Audio

### 0.4.1 Sounds

The original Commander Keen Trilogy did only support the default PC Speaker. Only with the introduction of Commander Keen Dreams the team decided to support sound cards.

**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.

As mentioned in Chapter ??, audio hardware was highly fragmented. Commander Keen supported three sound cards and the default PC speaker, which meant generating assets multiple times for each and packing them together with an in-house tool called MUSE into an AUDIOT archive (an id software proprietary format):



Figure 12: MUSE splash screen.

id Software intended for Keen Dreams to have music and digital effects support for the Sound Blaster & 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 minute. The team didn't even have time to remove the music setup menu.



**Figure 13:** Setup music menu in Keen Dreams, although there is no music in the game.

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

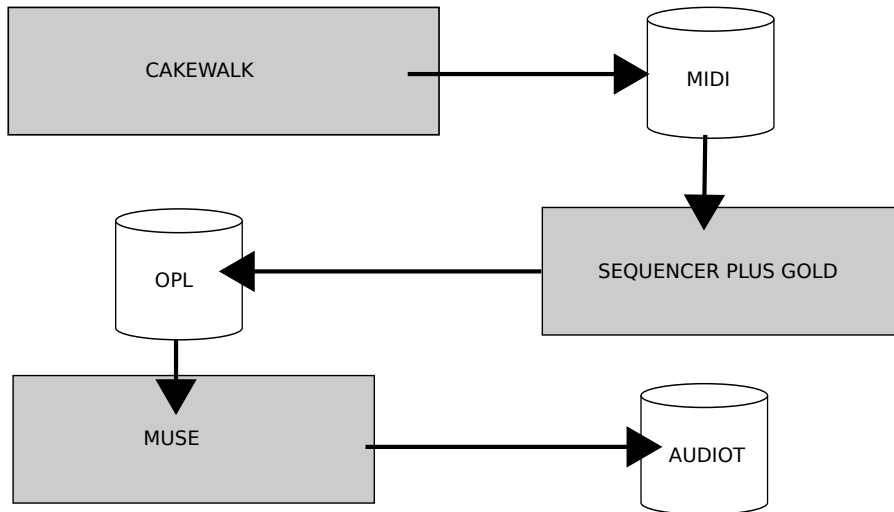
Two sets of each audio effect shipped with the game:

1. For PC Speaker
2. For AdLib

### 0.4.2 Sound effects

All sound effects are done by Robert Prince. In the early days of the OPL soundcards, the "gold standard" sequencing software was Sequencer Plus Gold ("SPG") by Voyetra.

The reason for this was it had an OPL instrument/instrument bank editor. To rough out compositions, Robert used Cakewalk ("CW").



**Figure 14:** Music and sound effect pipeline as used by Bobby Prince.

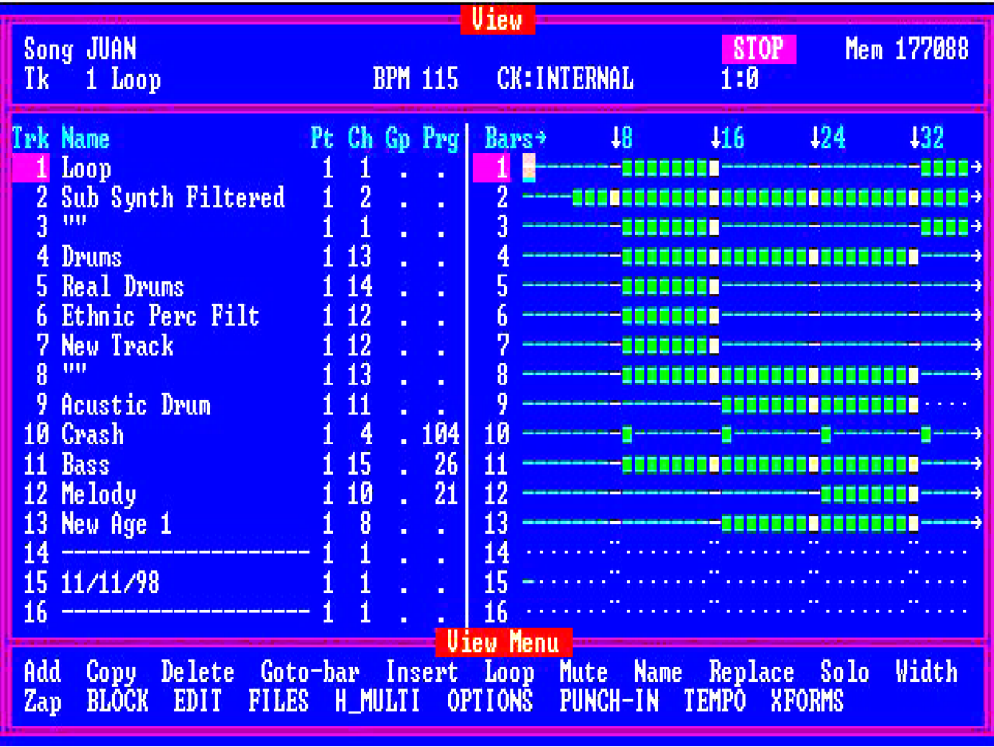


Figure 15: Sequencer Plus Gold ("SPG") by Voyetra.

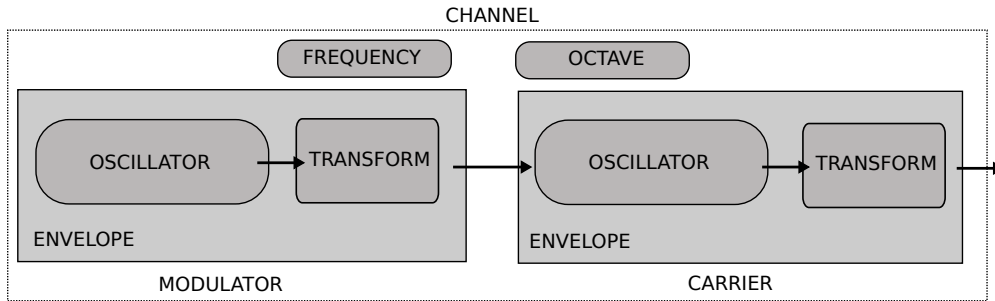
What goes in the AUDIOT archive is a music and sound effect format called IMF<sup>3</sup>. As it supports only the YM3812, it is tailored for the chip with zero abstraction layers. It consists of a stream of machine language commands with associated delays<sup>4</sup>.

The stream pilots the nine channels in the OPL2. A channel is able to simulate an instrument and play notes thanks to two oscillators, one playing the role of a modulator and the other the role of a carrier. There are many other ways to control a channel such as envelope, frequency or octave.

The way a channel is programmed is described in detail in Section ??, "??" on page ??.

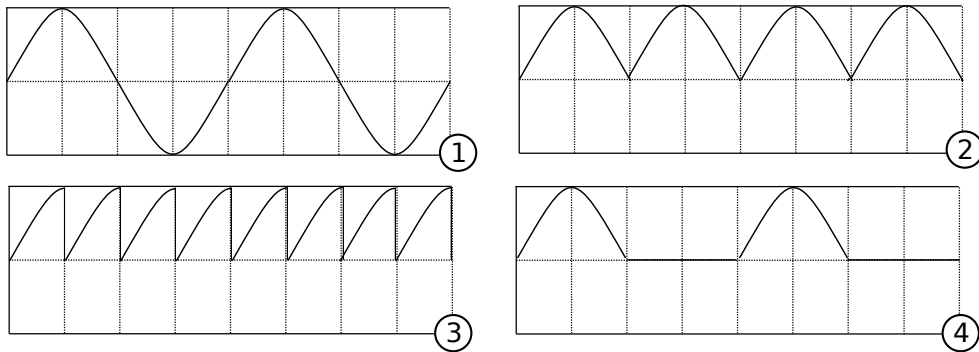
<sup>3</sup>Id software Music File.

<sup>4</sup>IMF format is explained in detail on page ??



**Figure 16:** Architecture of a YM3812 channel.

**Trivia :** The YM3812's unmistakable sonority is due to its peculiar set of waveform transformers (they are right after the output of each oscillator in the drawing). Four waveforms are available on the OPL2: Sin ①, Abs-sin②, Pulse-sin ③, and Half-sin ④.



**Figure 17:** The four waveform transforms available.

## 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 game<sup>5</sup>.

<sup>5</sup>The settlement with Softdisk is explained in detail in Appendix ??



**Figure 18:** Retail version of Commander Keen in Keen Dreams by Softdisk.

In 1990, the Internet was still in its infancy and the best medium was the 3½-inch floppy disk. The game shipped as follows:

The files 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 levels layouts.
- KDREAMS.CMP: Introduction picture of the game, which is a compressed LBM image file.



- Softdisk Help Library files, which are text screens, shown when starting the game
  - START.EXE: Decompress and show user guide, and then start the game.
  - LOADSCN.EXE: Shows the closing screen when quitting the game. Decompresses the ENDSCN.SCN chunk in LAST.SHL
  - \*.SHL: Text user guide assets.
- Several \*.TXT files which can be read in DOS by typing the corresponding \*.BAT file.

```

Directory of C:\KDREAMS\
.                <DIR>                16-05-2023 21:04
..               <DIR>                16-05-2023 20:52
BKGMND  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 19:** All Keen Dreams files as they appear in DOS command prompt.

