Documentation

KEENGINE

It's a keen engine!

Written by *K1n9_Duk3*

Compiled on February 18, 2017

http://k1n9duk3.shikadi.net

Contents

About KEENGINE

KE**EN**GINE ("the software") is a game engine developed in a collaborate effort by Rübennase and K1n9_Duk3, with additional contributions by Keenfansite ("the authors").

THE SOFTWARE IS PROVIDED "AS IS". THE AUTHORS WILL TAKE ABSOLUTELY NO RESPONSIBILITY OF ANY KIND. YOU ARE USING THE SOFTWARE AT YOUR OWN RISK!

Commercial exploitation of *any* kind requires prior written permission from the authors of the software.

Preface

This documentation addresses all those who want to use KE**EN**GINE for their own game projects. If all you want is to play the games, you do not need to read this. Please refer to the in-game help texts instead.

On the following pages, you will find manuals for the various tools of the engine and information on some of the data structures used by the engine.

If something is left unclear after reading this documentation, or if you find any errors in this document, please contact us, so we can improve this documentation. Possible ways of contacting K1n9_Duk3 are:

- via e-mail to `k1n9duk3@arcor.de`
- via the PCKF `http://www.pckf.com/`

Part I.

The Game

1. Customizing Your Game

1.1. General Settings

All the general settings for a game using KE**EN**GINE are stored in the `game.ini` file in the game's data folder or archive. This file contain various settings, ranging from simple things, like the name of your game, to more complex settings that might totally break your game it you set them to the wrong values, like the tile width and height.

Documenting all the possible settings in `game.ini` would be a fair amount of work, and there is also the danger of the developers forgetting to add new options to this documentation. To give modders an idea of what data is actually read from the file, KE**EN**GINE now has the `/dumpini` parameter. If that parameter is used, KE**EN**GINE will create a new `game.ini` file in the user directory, which contains all the data from your game's `game.ini` file(s), as well as every other entry KE**EN**GINE tried to read from the file(s) but could not find. In that case, the default values for those entries are written to the exported file.

Note: If an entry in the exported file is empty (i.e. there is nothing after the "="), then that entry is supposed to contain a string. It is probably meant to store either some text or a file name.

TODO: document all the entries from game.ini!

1.2. The Demo Loop

1.2.1. Structure Of The Demo Loop

The demo loop is started directly after the KE**EN**GINE intro animation finishes. The following elements are part of the demo loop:

1. Game intro

   This executes the Lua script in `Scripts\intro.lua` in your data folder or archive. By default, that script will show an animation similar to the "Terminator" intro found in **Commander Keen 4-6**. If you know how to write Lua scripts, you can easily create your own intro animation if you like.

2. Title screen

   This shows the title screen for 600 frames (which means for 10 seconds if the refresh rate is set to 60 Hz).

3. High scores

   This shows the high scores demo. The demo file `Demos/highscore.dmo` in your data folder or archive is played in a special demo mode that does not show the "Demo" logo, but does draw the current names and scores. The text will be drawn on top of the regular tiles, but behind the actors and foreground tiles.

4. "Star Wars" text scroller

5. User demo

   This plays the demo file `demo.dmo` from the user directory (by default, the user directory is the directory `keengine.exe` is located in). This file is overwritten every time the user starts a new game from the main menu.

6. Developer demos

   This plays the demo files `Demos\demo*.dmo` from your data folder or archive. At startup, the game scans the `Demos` folder for the files `demo0.dmo`, `demo1.dmo` and so on, so it knows how many demo files there are. It searches for files that are numbered sequentially, starting with 0, and will stop if it cannot find a demo file for the current number. The total number of developer demos will be written to the log file.

   After a demo file is finished, the next demo file is played. If there are no more demo files to play, the demo loop wraps around and starts with the game intro again.

If the user presses escape at any point during the demo loop, the game will stop whatever part of the demo loop was active and open up the main menu. If you select the "return to demo" option from the main menu, the demo loop will continue with the next element in the loop.

1.2.2. Recording A Demo

As mentioned above, the game automatically starts recording a demo every time you start a new game from the main menu. It will keep recording until you end a game or you load a saved game, so it is possible to record complete playthroughs as a demo. If you die while recording a demo, the demo will also store whether you selected to restart at a checkpoint, restart the map, or return to the world map.

If you hold down the Tab key while selecting "new game" in the main menu, the game will show you a list of all the level files in the `Levels` subdirectory or your data directory or archive. This allows you to record a demo for any level, even those that would normally be inaccessible when playing the game, like the high score level.

If you want to keep a recording, be sure to rename the `demo.dmo` file in your user directory and/or copy it into another directory.

If you want to use your new recording as the demo for the high scores, you need to rename it to `highscore.dmo` and copy it into the `Demos` subdirectory of your data directory or archive. If you want to use the demo as one of the developer demos, copy it into the `Demos` subdirectory and rename it so that it has the correct number in the name.

1.2.3. The "Star Wars" text scroller

This will show the text from the associated text file (that is `ScrollerText.txt` in the `Text` subdirectory by default). Unlike the other text files (see chapter 9), this does not use any special commands. However, there are some things you need to keep in mind:

- The game will automatically create line breaks when a line is too long to fit on the screen. However, lines can only be broken by an empty space. Using really long words or words-connected-by-hyphens will cause errors.
- Each line in the text file represents a paragraph. Each paragraph is drawn block-aligned by default. The last line of each paragraph will be left-aligned.
- If a line in the text file starts with a space or a tab, the line will be drawn centered.

You can change the font (and font size), text color, background image, background music, and even the text file by editing the settings in `game.ini`.

If no background image is selected, the game will create a random arrangement of "stars" (basically a few pixels of random brightness on black background) and use that as the background image for the text. That background will be different every time the text scroller is started.

Note: The text file for the scroller can be changed by language files.

2. Preparing Your Game For Release

2.1. Cleaning Up

Before you start adding files to a Zip archive, you should clean up your project. This includes:

- removing unused files
- making sure that PNG images that do not need transparency (like the background images and the game's title screen) do not have transparency
- compressing all PNG images with Ken Silverman's "PNGOUT" (`http://advsys.com/ken`) or Ardfry Imaging's "PNGOUTWin" (`http://pngoutwin.com`)
- deleting all the hidden `Thumbs.db` files from the `Data` directory and its subdirectories

Note: If you get an error when running PNGOUT or PNGOUTWin on a PNG file, you are probably dealing with a PNG file using 16 or 24 bits per channel. Since the image will be converted to 8 bits per channel by the BlitzMax libraries anyway, you might as well make sure that all PNG files you create are using 8 bits per channel. Plus, 16 or 24 bits per channel are by definition two or three times larger than 8 bits per channel.

Contrary to what you might have been told, size *does* matter in some situations. Some people might be grateful if you don't make them waste their time and/or money by letting them download unnecessarily large files.

2.2. Creating A Package

There are multiple ways to create a package that can then be released:

- Add your `Data` folder and `keengine.exe` to a zip file.

- Create a zip file containing all the files and directories from your `Data` folder and rename the zip file to `Data` (without the `.zip` extension), then add the `Data` archive and `keengine.exe` to a zip file.

If you do not like the idea of renaming a zip file to a file with no extension, or you wish to split your game data into several directories and/or archives, you must make sure that the people playing the game run it with the correct `addpath` parameters. You can achieve this by doing one of the following:

- Write a batch file that runs the game with the correct parameters.
- Write your own front-end that runs KEENGINE with the correct parameters.
- Create an installer for your game that creates a desktop icon to run the game with the correct parameters.

Note that if you decide to create an installer, you also need to adjust the user directory (i.e. the directory where saved games, screenshots, settings and the log file will be stored). This is required because the installer will probably install the game in your "Program Files" directory and Windows usually will not allow programs to write or modify files in the "Program Files" directory unless the user is an administrator.

Writing a front-end has the benefit of allowing you to hide the text window that usually pops up immediately after starting the game. But keep in mind that this should only be done for the final release version. You need to be able to see the text window while playtesting, because that window is the only place where you will be able to see error messages from the Lua script parser.

2.2.1. Sample Batch File

To create a batch file, you need to open a tool that will allow you to edit plain text files, like "Notepad". To open Notepad in Windows, hold down the Windows key and press R. Type `notepad` or `notepad.exe` into the small window and press Enter. Enter the following text in Notepad:

```
@keengine.exe addpath data.zip
```

Now you need to save this as a batch file. Select "Save As.." from the "File" menu. You may need to select "All files" as file type (to prevent the program from appending the extension `.txt`), then save the file as `start game.cmd` or `start`

`game.bat`. The file name does not really matter, as long as it ends with `.cmd` (for Windows XP and above) or `.bat` (for Windows 95/98/Me).

2.2.2. Sample Front-End Code

You can create a simple front-end or launcher program that hides the text window and passes the correct parameters to KE**EN**GINE in literally a few lines of code.

Pascal/Delphi Code

```
program launcher;

uses
  Windows;

begin
  WinExec('cmd /c keengine.exe addpath data.zip', SW_HIDE);
end.
```

C/C++ Code

```
#include <windows.h>

void main (void)
{
  WinExec("cmd /c keengine.exe addpath data.zip", SW_HIDE);
}
```

2.2.3. Providing Alternatives

Since the BlitzMax libraries only support sound stored in WAV or OGG format, you will probably end up with a `Music` subdirectory that is several times larger than the rest of the game. You might want to consider providing different versions of your game for download:

- A complete package containing everything. This is what most people would download.

- A basic package containing only the game files, but no music at all. This is for those with terribly slow internet connections.
- The high-quality version of the game music. The same music as in the complete package.
- A low-quality version of the music. This would contain shortened and highly compressed versions of the music. For those who have terribly slow internet connections but still want to have music in the game.

Providing these alternatives will not only let people with slow internet connections download the game more quickly, it will also help reduce the traffic for the webserver hosting the files. The downside is that you would need to upload more files and that those files take up more space on the webserver.

Part II.

The Tools

3. Map Editor

3.1. Basics

The Map Editor works like most Paint programs. That means you can use the mouse to select the individual parts that make up the map and place them at the desired location simply by clicking the mouse. If you have worked with one of the many editors for **Commander Keen** or similar games, you should be able to get used to KE**EN**GINE's map editor fairly quickly.

3.2. Getting Started

The Map Editor is part of the engine and is included in the main executable (named `keengine.exe` by default). So if you obtained a game based on KE**EN**GINE, you automatically have access to the tool required to edit or create new levels.

To run the Map Editor, you need to run the main executable with the parameter `mapedit`.

3.3. Creating A New Map

Select the option "New Map" from the main menu. This will open the Level Properties window (see figure 3.1). If you accidentally selected "New Map" from the main menu, simply press Escape or select the "Cancel" button to return to the main menu.

3.3.1. Level Properties

The red dot marks the currently selected option. The cursor keys ↑ and ↓ can be used to navigate through the menu options. Pressing Enter lets you edit the

currently selected option. Press Enter again to accept your changes, or press Escape to discard them.

Figure 3.1.: Level Properties window

- **Level size:** The entries "Width" and "Height" define the size of the map (measured in tile units). The minimal size that is displayed here is automatically calculated from the resolution and tile size defined in your `game.ini` file. You should not create levels smaller than the minimal size, as those levels will not be displayed correctly in the game.

  There is no maximum size, but only levels up to a size of 4096 × 4096 can use features like teleporters and switches in the entire level. Besides, a level as huge as this will be larger than all levels of all **Commander Keen** games *combined*. You also need to be aware of memory issues. Every level takes up $width \times height \times 16$ bytes when loaded in the game or the editor, so you need at least 268 MB of free memory just to be able to load a 4096 × 4096 map itself.

  The level size – like all the other level properties – can be changed afterwards. The change will be applied at the right and/or bottom edge of the level. If the size is increased, empty tiles will appear there. If the size is decreased, the

tiles and map items beyond the new edge will be deleted. Resizing the level will not insert any new "Edge of Map" tiles.

- **Title:** Here you can enter a name for your level. This name does not need to match the level's file name and it may contain any symbol.

  The title will be used by default to display the message for entering a level, and it will also be displayed in the in-game status menu. There are settings in `game.ini` and the language files that can override the title stored in the level file during the actual game. However, if you start a level directly from the main menu, the game will still use the title from the level file.

- **TileSet:** This defines which TileSet you want to use to create your level. You can only select from the TileSets that were present in your `Tileset` subdirectory when the Map Editor was started.

  However, the Map Editor does not check if the TileSet files can be loaded properly. So you might encounter an error message when the Map Editor creates the level and loads the TileSet.

  You can change the TileSet for an existing level, too, but there are some issues you need to be aware of:

  1. The new TileSet will not be loaded until you save the map and re-open it in the Map Editor.

  2. Changing the TileSet only makes sense if the layout of the tiles in the old and the new TileSet are almost identical. Otherwise you will end up with garbage.

  3. If you accidentally changed the TileSet and then saved the map file, you should still be able to change it back to the previous setting.

- **Background and Background-Scrolling:** This allows you to select a background image that will be drawn behind all of the layers of the map. The background allows you to create an illusion of depth for your map. Like the TileSet, you can only select images that were found in the `Backgrounds` subdirectory when the editor was started.

  The background image will be drawn "tiled" in-game, which means the image is repeated in all directions. The Map Editor does not display the background image at all.

  X-Dir and Y-Dir only have an effect if a background image was selected. The selectable options have the following effect:

  - **NONE:** The background does not "move" in this direction at all. This makes the background appear very far away. This setting is useful if the background shows a sky with a sun or a moon.
  - **NORMAL:** The background "moves" at the same speed as the rest of the level. This makes the background appear very close and makes the map look flat.
  - **PARALLAX:** The background "moves" at half the speed. This gives the map some depth and is usually a neat effect.

- **Music:** You can select the background music for the level here. As before, you can only select from the files that were present in the `Music` subdirectory when the Map Editor was started. The music is only played in the game, not in the Map Editor.

3.3.2. Initializing

When you select the "OK" button in the level properties menu, the Map Editor generates a new map that is almost completely empty. It only fills the borders of the Map-Layer with "Edge Of Map" tiles. See section 3.4.4 at page 19 for further information.

3.4. Editing A Map

3.4.1. Overview

After you have created a new map or loaded an existing map, the Map Editor will look similar to what is shown in figure 3.2. The individual components are:

- Top-left corner: Displays the name of the map layer that is currently selected.
- Top-right corner: Displays the current zoom level.
- Bottom-left corner: Displays the level coordinates of the mouse cursor.
- Bottom-right corner: Displays additional information for the tile (or Map Item) at the current mouse position.
- Left side: Shows the TileSet (or Map Items) for you to select from.
- Right side: Shows the actual map.

Figure 3.2.: The Map Editor

3.4.2. Controls

The core of the Map Editor is (currently) the only part of the engine that requires a mouse for user input.

Note to Mac users: You need a mouse with at least *two* buttons.

- Use the left mouse button to place tiles in the currently selected layer. You can hold the mouse button down and move the mouse cursor around to fill larger areas of the map. Of course, this only works if the mouse cursor is placed above the map area and the current selection of tiles does exceed the limits of the map.
- The right mouse button is used to select the tile(s) that you want to place in the map. You can select tiles from the map as well as from the TileSet (or MapItems). By holding down the right mouse button, you can select a

rectangular block of any size. Release the right mouse button to accept your selection.

- The mouse wheel is used to scroll the TileSet or the MapItems window (depending on which is currently visible). Holding down the Ctrl key while using the mouse wheel will change the zoom level for the map field.

Note: You can cancel a selection by clicking the left mouse button while the right mouse button is still held down.

The following keys can be used while editing a map:

- **Cursor keys:** scroll the map in all four directions
- **PgUp**/**PgDn**: scroll the TileSet or MapItems window up/down
- **Z**: toggles displaying collision data and TileType icons
- **T**: changes transparency for inactive layers
- **F2**: changes to Background Layer
- **F3**: changes to Level Map Layer
- **F4**: changes to Foreground Layer
- **F5**: changes to Actor/Info Layer
- **F9**: shows the Level Properties menu
- **Esc**: shows the Load/Save menu
- **+**/**Num +**: zoom in *or* increase MapItem value
- **-**/**Num -**: zoom out *or* decrease MapItem value

3.4.3. The Layers And Their Purpose

Only the Level Map layer and the Actor/Info layer have an actual purpose. The other layers are purely cosmetic.

As the name suggests, the Background layer contains the tiles that are to be drawn in the background (i.e. behind the tiles in the Level Map layer). The foreground layer stores tiles that are to be drawn in the foreground (i.e. in front of the Level Map layer). Unlike the tilesets from the **Commander Keen** games, you can place

any tile in any layer, but some tiles will not work properly unless placed in the Level Map layer.

The Level Map layer defines the structure of the map. Only the collision data of the tiles from the Level Map layer is used in the game. All tiles that are not used for purely cosmetic reasons need to be placed in the Level Map layer. This includes walls, switches, doors (teleporters), goodies and hazards.

The Actor/Info layer is used to store objects such as the player and the enemies via MapItems[1]. In addition, this also stores the target coordinates for teleporters and switches, WorldMap information, dialogue numbers and other information.

3.4.4. "Edge Of Map" – What Is It Good For?

The "Edge Of Map" tiles show you the border/edge of the map in the level editor. This is helpful while scrolling through the map in the editor, but the tiles were originally introduced for a different purpose.

Having an actor accidentally cross the limits of the map can lead to problems (earlier version of KE**EN**GINE just crashed in such situations). But the default (and currently only) way of exiting a level is to have the player walk into the left or right border of the level (i.e. into the area of the "Edge Of Map" tiles). To prevent design errors, the border of the map is filled with these blocking "Edge Of Map" tiles. To create an exit, you need to erase them at the desired location.

Note: You should only erase the tiles from the inner column. Leave the outer column blocking, so that the other actors will not be able to cross the level border at that point.

Having blocking tiles at the top and the bottom of the level is necessary to prevent the player from jumping out of the level and then walking/falling either to the left or to the right until the level is won.

The two outermost rows/columns of the level are (usually) never drawn in the game. The camera is not allowed to scroll far enough for these tiles to become visible.

[1] MapItems: see chapter 8 on page 43

3.4.5. Switches, Teleporters, Gem Sockets And Triggers

For switches, teleporters, gem sockets and other triggers to work properly in the game, you need to assign destination coordinates to them. Switches, gem sockets and triggers will toggle the tiles in the Level Map layer at the assigned coordinates. The tiles will be replaced by their ToggleTiles[2]. Teleporters will teleport the player to the assigned coordinates. All destination coordinates are drawn as hexadecimal numbers on a white background.

Note: Destination coordinates have a limited range for both the x- and the y-coordinate. Both values must be in range 0 to 4095_{10} (or FFF_{16}).

To select a destination coordinate, you must either select the MapItem or select an existing destination coordinate from the map. The selection needs to contain *exactly one single tile*. Now you can right-click on any spot in the map, and instead of selecting tiles or map items from the map, the editor will change the selected coordinates to the coordinates of the spot you right-clicked on. These coordinates need to be placed directly on top of the switch or teleporter.

When using teleporters, you need to make sure that there is free space at the desired destination, so the player does not end up stuck inside a wall after teleporting.

Destination coordinates (usually) only have an effect when they are placed on top of tiles with the correct TileType value. Some of those TileTypes will only work properly if they are placed correctly in relation to the tile the player is standing on. Gem sockets must be placed at the player's feet. Switches and teleporters need to be placed one tile above the player's feet. The ground must not be sloped, otherwise the tiles might not work as intended.

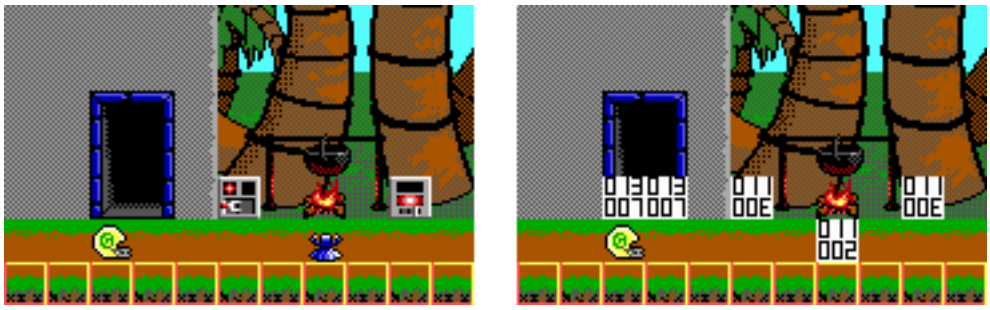

Figure 3.3.: Correct placement of teleports, switches and gem sockets

[2] see TileSet format

You can also create triggers by placing a destination coordinate on top of a "goodie" tile (such as ammo, bonus items, tokens, extra lives, life drops or key gems). In that case, the tile at the selected destination will be toggled when the player collects that goodie, and the destination coordinate is removed from the Info layer, so the trigger will only be activated once.

If you want to toggle objects that are made up of more than one tile, these tiles need to be in a rectangular shape and you need to select the coordinates of the top-left corner of that rectangle as the destination coordinates. The Map Edit will detect the top-left corner of the block automatically if you hold down the Shift key while right-clicking on the tiles. All tiles in that rectangle need to be set to the same ToggleGroup value for this to work, and that value must not be 0.

3.4.6. WorldMap Settings

To create a working WorldMap, you need three special types of information: level entrances, flag locations, and change points. The editor displays these settings as hexadecimal numbers on a yellow background.

- **Level entrance:** A level entrance defines the location on the world map from which the player can enter a new level. The actual tiles at this location and their TileType values are ignored.

  To place an entrance, you must either select the MapItem or select an existing level entrance from the map. All level entrances are marked with the letter "E" (for "Entrance") in the upper coordinate. The lower coordinate shows the (hexadecimal) level number. You can change the level number by using the + and - keys.

- **Flag locations:** The flag locations are important for two reasons. They define the location where the flag will be placed after the player has beaten that level. If a level does not have a flag location placed on the world map, the players will be able to enter said level from the WorldMap over and over again for as many times as they desire. If there is more than one flag location for the same level, the game will use the first flag location it encounters and will ignore the others.

  To place a flag location, you must either select the MapItem or select an existing flag location from the map. All flag locations are marked with the letter "F" (for "Flag") in the upper coordinate. The lower coordinate shows the (hexadecimal) level number. You can change the level number by using the + and - keys.

- **Change points:** These points indicate which tiles are to be toggled after finishing a level. However, they will only have an effect if that level also has a flag location placed on the world level, so that the level can only be finished once.

  To place a change point, you must either select the MapItem  or select an existing change point from the map. All change points are marked with the letter "C" (for "Change") in the upper coordinate. The lower coordinate shows the (hexadecimal) level number. You can change the level number by using the + and - keys.

  Unlike the regular way of toggling tiles (via a switch, a trigger, or a gem socket), the ToggleGroup information of the tiles is ignored completely. You need to place a change point on *every single tile* you want to toggle. The upside is that this allows you to toggle pretty much any tile at any location on the WorldMap.

Note: Only the tiles in the Level Map layer can be toggled. Be sure to design your WorldMap and your TileSet accordingly.

3.4.7. Messages

Messages are used to trigger a dialogue in the game. See the comments in `messages.ini` for further information on the dialogue system.

Messages on the WorldMap can be placed anywhere you want. Messages in a normal level need to be placed on top of a tile with the correct TileType value.

To place a Message, you must either select the MapItem  or an existing message from the map. All messages are drawn as a hexadecimal number on a light blue background. You can change the message number by using the + and - keys.

The message tile and the message info need to be placed one tile above the player's feet. The ground must not be sloped, otherwise the tiles might not work as intended.

A message event may also cause some tiles to be toggled, even without having to use scripts. To use that feature, simply place the desired destination coordinate above or below the message. Coordinates placed above the message will be toggled once and the destination coordinate will be removed from the Info layer after that. Coordinates placed below the message will be toggled every single time the dialogue is shown.

The ability to toggle tiles via message tiles was added before the script system for the dialogues was implemented. You could also perform the same operations (and more) by using a Lua script for the dialogue.

Note: Toggling tiles will only work in a normal level. For a WorldMap, you have no choice but to write a Lua script for the dialogue.

3.5. Developer's Notes

3.5.1. Configuration And Special Features

You can change some settings for the Map Editor in the section `EDITORS` in `game.ini`.

The entries `mapedit winwidth` and `mapedit winheight` define the window size of the Map Editor. A resolution of less than 640×480 is not recommended.

In that section, you can also define whether the TileSet graphics are to be loaded into video RAM before editing a map. Set the entry `preload pics` to 1 to enable this, or set it to 0 to disable this option. This entry will affect the Map Editor as well as the TileSet Editor.

The Map Editor is smart enough to detect if a layer is actually used in a level. If the entry `optimize maps` is set to 1, then any unused layers are not written to the file when the map is saved.

3.5.2. Unimplemented Features

The following features are still on the "To Do" list:

- fullscreen mode
- internal menus for loading and saving files (required for fullscreen mode)
- full mouse support for the menus
- undo option

4. TileSet Editor

4.1. Basics

Creating a TileSet is quite a complex task. Having previous experience with other TileSet creation utilities could be beneficial. On the other hand, the TileSet Editor in KE**EN**GINE is not based on any existing tool, so the names and options may not match those used in other editors.

4.2. Getting Started

The TileSet Editor is part of the engine and is included in the main executable (named `keengine.exe` by default). So if you obtained a game based on KE**EN**GINE, you automatically have access to the tool required to edit or create new TileSets.

To run the map editor, you need to run the main executable with the parameter `ted`.

Figure 4.1.: Main Menu Of The TileSet Editor

4.3. Creating A New TileSet

Select the option "New TileSet" from the TileSet Editor's main menu (see figure 4.1). This will open up another menu that will as you if you want to create an empty TileSet or create a tile for each cell in the TileSet image. You should select the second option, unless you are an advanced user and you want to create a tileset from an "optimized" image file.

After you made you selection, you will be asked to open a PNG image that contains the image data of your TileSet. That image can be any PNG image, as long as its width and height are multiples of the tile width and height defined in your `game.ini` file (16×16 pixels by default). It is recommended to use an image with a width of 288 pixels, as that will result in the TileSet showing the individual tile images in the same arrangement as in the original image. Figure 4.2 shows the difference this will make.

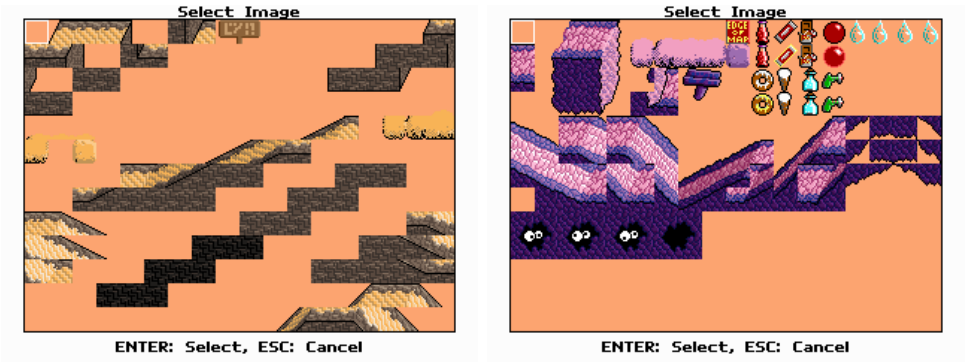

Figure 4.2.: TileSet Images With A Width Of 256 Pixels (Left Image) And 288 Pixels (Right Image)

Note: The TileSet Editor will allow you to select an image file from any location, but you cannot use images from any other location than your `Tileset` subdirectory! The engine will not be able to load the TileSet if the image file is not located in the same subdirectory as the TileSet file.

After having created the TileSet, the first thing you should do is define the "Edge Of Map" tile. This tile will be used by the Map Editor to fill the edges of the map (see section 3.4.4) and should *always* be blocking in all directions. So be sure to assign the proper Collision value (currently 1) to your "Edge" tile. The next section explains how to do that.

4.4. Editing A TileSet

After having loaded a TileSet or created a new TileSet, the TileSet Editor will enter its edit mode (see figure 4.3).

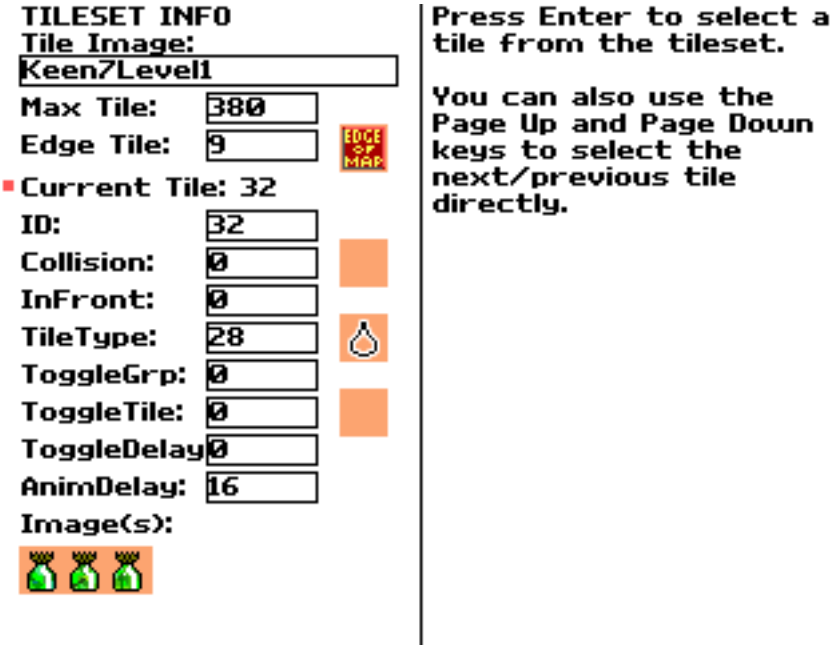

Figure 4.3.: TileSet Editor's Edit Mode

The currently selected menu option will be marked by a small red square. You can use the cursor keys ↑ and ↓. The upper entries "Tile Image", "Max Tile" and "Edge Tile" are TileSet properties. "Current Tile" shows the number of the currently selected tile. To select another tile, you can either use the PgUp and PgDn keys, or you can select the "Current Tile" menu entry and press Enter to bring up a new window that will let you select another tile.

Below the "Current Tile" entry, you will either see the words "NOT IN LIST" (which means this tile does not yet exist), or the tile's properties.

4.4.1. TileSet Properties

- **Tile Image** shows the name of the TileSet's image file (without extension). This file name does not need to match the name of the TileSet file. This name cannot be changed right now, and it should not be necessary to change it.

- **Max Tile** defines which tile is the last tile in the tileset. Press Enter to edit this value, then press Enter again to accept the changes, or press Esc to cancel. If this value is changed to a lower value, all tiles whose number is greater than the new value will be deleted from the TileSet without asking for confirmation. If you change the value to a higher value, new tile slots will be inserted at the end of the TileSet.
- **Edge Tile** defines the tile that is to be used by the Map Editor to fill the edges of the map. Press Enter to bring up the Tile Selector.

4.4.2. Tile Properties

- **Current Tile** shows the number of the currently selected tile. Press Enter to bring up the Tile Selector.
- **NOT IN LIST**: If the current tile does not exists, only this text is shown. Select this text and press Enter to create a new tile in this slot.
- **ID** stores the position of this tile in the TileSet. If you change this number from ID_{old} to ID_{new}, where $ID_{old} \neq ID_{new}$ and $ID_{new} \leq MaxTile$, the following happens:

  1. The old tile at position ID_{new} is replaced by the current tile.
  2. The current tile's number is changed from ID_{old} to ID_{new}.
  3. The tile at position ID_{old} is removed.

- **Collision** defines the collision type of this tile. Pressing Enter allows you to select one of the supported collision types. The collision types and their icons are currently hard-coded.

  The collision type number 15 has a special meaning. Tiles of this type can be destroyed by firing the Neural Stunner at them. If they are hit, they are removed from the map (i.e. they are replaced by tile 0).

  Note: This special behavior of collision type 15 is actually something that should rather be converted into a TileType, since it has nothing to do with the actual collision handling. The same applies to the platform blocking types 23 and 24. These collision types are likely to be removed in future versions of the engine.

- **InFront** defines whether a tile is drawn in front of the actors (1) or behind them (0).

- **TileType** defines the type of this tile. In theory, any value from 0 to 127 can be used, but since all behaviors are hard-coded in the engine, setting the tile type to an undefined value will not have an effect at all.

  As of version 0.43.0, the tile type is selected from a list of icons, much like the collision type. The same icons are used to display tile type information in the Map Editor and the game itself.

- **ToggleGrp** defines the toggle group of the tiles. Values from 0 to 255 can be used. The toggle group information is used by the game to toggle larger objects comprised of individual tiles, such as bridges and doors. If this value is 0, the game will only toggle a single tile. If the tile is not 0, the game will toggle all tiles to the right and below this tile that have the same toggle group. Ideally, the tiles with the same toggle group should be arranged in a perfect rectangle.

  You should assign different toggle groups to the different object types, so that the game will not accidentally toggle parts of a bridge when a nearby bridge is toggled (and vice versa). Which numbers you assign to which object type is completely up to you.

- **ToggleTile** defines which tile will be used to replace this tile when this tile is toggled. This will either store the ID if the new tile, or -1 if the tile cannot be toggled (i.e. the tile is replaced by itself).

- **ToggleDelay** defines the delay (in game ticks) before this tile is toggled automatically. This basically creates an animated tile (see below), except this will actually replace the tile in the level, thus changing the collision type and tile type values. A value of 0 means that the tile will not toggle automatically, which is the default behavior. You can set a delay of up to 65535 ticks, which would be 18 minutes and 12.25 seconds (60 ticks = 1 second at a frame rate of 60 FPS).

- **AnimDelay** defines the delay during the tile animation. Values from 0 to 255 are allowed, where 0 means that the tile is not animated. The delay is measured in game ticks (see above). To create an animated tile, you need to define a delay here and then add the desired animation frames to the Image setting.

- **Images** shows which cells from the TileSet image are used to display this tile. Press Enter to open the Animation Editor (see figure 4.4).

Except for **AnimDelay** and **Images**, all the tile properties will only have an effect when the tile is placed in the LevelMap layer.

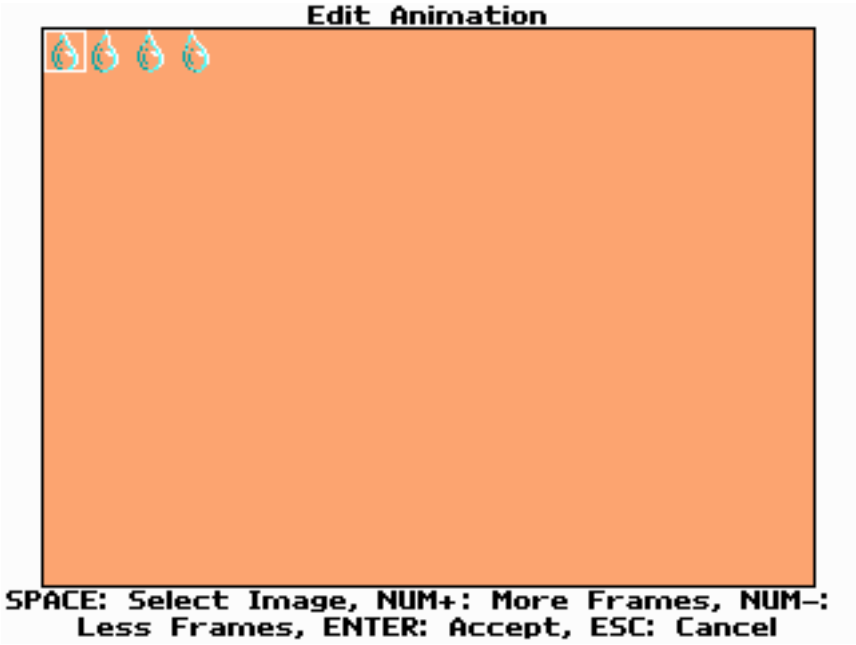

Figure 4.4.: Animation Editor

4.4.3. The Tile/Image Selector

The image selector is shown when you try to change the collision type or tile type values, or when you change an animation frame in the Animation Editor (see below). You can use the PgUp/PgDn and the arrow keys to navigate to the desired tile and press Enter to select it. Pressing Esc will abort and take you back to the previous menu.

You can also use the mouse to select a tile. Simply move the mouse cursor over the desired tile and click the left mouse button. You can use the mouse wheel to scroll.

The Tile Selector is shown when you select the "Edge Tile", "Current Tile" or "Toggle Tile" options. It uses the same controls as the image selector, but with one addition: you can hit the Tab key to toggle collision and tile type drawing.

4.4.4. The Animation Editor

The Animation Editor allows you to select the image cell(s) from the TileSet image that will be used for drawing the tile. You can even create a sequence of images that will result in an animated tile. You can use the following keys to edit the images.

- **Cursor keys:** Select an animation frame.
- **Space**: Opens the Image Selector and lets you select a new image cell for the current animation frame.
- **Num +**/**Num -**: Adds/removes an animation frame at the end of the animation. You can create animations of up to 255 frames.
- **Enter**: Accept all changes to the animation sequence and close the animation editor.
- **Esc**: Discard all changes to the animation sequence and close the animation editor.

Keep in mind that the animation delay value applies to every single frame of the animation. If you want to show a certain frame for a longer period, you need to add multiple frames using the same image cell to the image sequence.

4.5. Developer's Notes

4.5.1. Configuration And Special Features

You can define whether the TileSet graphics are to be loaded into video RAM before editing the TileSet. In the section `EDITORS` in `game.ini`, set the entry `preload pics` to 1 to enable this, or set it to 0 to disable this option. This entry will affect the Map Editor as well as the TileSet Editor.

4.5.2. Unimplemented Features

The following features are still on the "To Do" list:

- fullscreen mode
- internal menus for loading and saving files (required for fullscreen mode)

- full mouse support for the menus

5. Text Viewer

5.1. Basics

If you have read the help texts in **Commander Keen: Goodbye, Galaxy!**, **Bio Menace** or **Wolfenstein 3-D**, you should be able to handle the KE**EN**GINE Text Viewer. The Text Viewer is included as a developer's tool to allow you to view any text file without having to gain access to the text in-game, to make it easier to check for errors.

5.2. Getting Started

The Text Viewer is part of the engine and is included in the main executable (named `keengine.exe` by default). So if you obtained a game based on KE**EN**GINE, you automatically have access to the tool required to view any of the text files.

To run the Text Viewer, you need to run the main executable with the parameter `textview`.

5.3. Viewing Text Files

Select the entry "View Text File" from the main menu and open the desired text file in the following dialogue. If the text file was opened correctly, you can read the text just like you would in the game.

However, there are some aspects in which the Text Viewer differs from the Text Viewer in the game:

1. The images are not kept in memory. Instead, the images are loaded again each time a text is loaded. So you do not need to restart the program if one of the image files was changed.

2. You can use the F5 key to re-load the entire text file. If possible, the Text Viewer will stay at the same page number after re-loading the text file.

This allows you to view the text file in the Text Viewer while editing the raw text file in a text editor of your choice and/or modifying the images in the text file. This should make it easier to create story and help texts for your game.

5.4. Developer's Notes

5.4.1. Configuration And Special Features

The Text Viewer uses the same graphics options as the game. You need to change the graphics options in the game if you want to change the resolution of the Text Viewer or switch from fullscreen mode to windowed mode (or vice versa).

Additional settings such as background color and default text color can be changed in `game.ini`. These settings will be used in the game as well as in the Text Viewer tool.

5.4.2. Known Issues

There is one issue related to the text color. If there is no Color-command at the beginning of a page, the text color will depend on whatever was the last text color. Depending on whether you navigate forwards or backwards through the text, the text color may change. This is intentional, as the text display in the original DOS games had the same issue.

6. Type Info Viewer

6.1. About

The Type Info Viewer is a text-based program that allows you to list all methods and fields of any type used in KE**EN**GINE. It will also list the type hierarchy, i.e. which type(s) the current type is based on and which types extend the current type. This was created to provide at least some kind of reference for modders, in case the documentation is not yet written or incomplete. It should help you create new Lua scripts and can also be useful for creating virtual actor classes.

The Type Info Viewer is part of the engine and is included in the main executable (named `keengine.exe` by default). To run the Type Info Viewer, you need to run the main executable with the parameter `typeinfo`.

6.2. Usage

Using the Type Info Viewer is simple. When the program is started, it will show the following prompt:

```
Enter Type Name:
```

This is where you enter the name of the type you want to learn more about. If you do not know the name of the type, you can enter the most basic type `Object` and the program will list all types that are based on the `Object` type. For actor types, you can start with `TLevelObject`, which is the type that all actors are based on.

Enter `exit` to quit the program. `Exit` is a keyword in the BlitzMax language, so there cannot be a type with that name.

Part III.

File Formats & Structures

7. Virtual Actor Classes

7.1. About

As of version 0.43.0, KE**EN**GINE is able to overwrite certain hard-coded values in the actor classes with values read from a `.ini` file. These settings are read from `actors.ini` located in your game's data directory.

This allows you to modify the default settings of the actor class, but also allows you to create virtual actor classes. Creating a virtual actor class is basically the same as modifying the default values of an existing actor class, but it does not actually replace the actor class. Instead, this provides you with a new class that can be assigned to a MapItem and can be used along with the original type.

7.2. Modifying Actor Settings

To modify an existing actor, you need to create a new section in `actors.ini` and name it after the the class you want to modify.

Note: All the actual actor classes start with a "T", like "TPlatform" and "TSnocky".

Disclaimer: You cannot modify every actor type. Only actor instances that are initialized by the `GetActorByName()` function will use the customized settings. All actors that are placed in the level via a MapItem will always use this, but actors that are spawned by other actors during the game (like the player's shots) will usually create new actor objects without calling that function.

7.3. Creating Virtual Actor Classes

To create a new actor class, you need to create a new section in `actors.ini` and name whatever you want to call the new class. Just make sure the the name is not used by any actual actor type.

The first thing you need to define for a virtual actor class is the name of the class it is based on. This is done by setting the `Type` value to the name of the base class. An example would look like this:

```
[MyPlatform]
Type = TPlatform
```

This creates a new virtual actor class called `MyPlatform` that is based on the `TPlatform` actor class.

Note: Virtual actor classes can be based on regular classes or on other virtual actor classes. The only restriction is that the references must not be cyclic (i.e. the virtual class must not be based on itself, neither directly nor indirectly) and any virtual class must eventually link back to an actual actor class.

7.4. Settings

Documenting all the fields of all the actor classes would be a fair amount of work, and there is also the danger of the developers forgetting to add new additions to this documentation. To give modders an idea of what data is actually read from the file, KE**EN**GINE now has the `/dumpini` parameter. If that parameter is used, KE**EN**GINE will create a new `actors.ini` file in the user directory, which contains all the data from your game's `actors.ini` file(s), as well as every other entry KE**EN**GINE tried to read from the file(s) but could not find. In that case, the default values for those entries are written to the exported file.

7.4.1. Sprite

The following fields define the actor object's sprite:

- **PicW** defines the width of each frame in the image
- **PicH** defines the height of each frame in the image
- **PicFrames** defines the total number of frames to load from the image.
- **PicFile** defines the file name of the sprite image (without the .png extension).

7.4.2. Hitbox

Each actor's hitbox is defined by the following fields:

- **BlockX** defines *half* the width of the actor's hitbox. This stores only half the width to make the collision detection slightly faster.
- **BlockY** defines the total height of the actor's hitbox.

8. MapItems

8.1. About MapItems

Map items are used to define the actors and the other types of information available for editing the Actor/Info layer of a map in the Map Editor. Except for the destination coordinates, WorldMap information and message numbers, only the information defined in the MapItems can be placed in a level.

8.2. Structure Of The MapItems

MapItems store the following information:

Type	Name	Default
Int	Number	—
Int	ImageIndex	Number
Int	RawData	Number
Byte	Skill	11111111_2
Int	Direction	0
String	Class	""
String	Info	"Map Item <Number>"

Table 8.1.: Structure Of The MapItems

8.2.1. Number

This stores the number of the MapItem (i.e. its index in the table of MapItems). The game will use this value to find the correct MapItem in the table of MapItems when a level is loaded.

The number is limited to a maximum value of 16777215_{10} ($FFFFFF_{16}$) due to the way the information is stored in the map's Info layer (see section 8.2.3). It is

rather unlikely that you will ever get anywhere near that limit anyway. Games like **Commander Keen 4-6** didn't use more than 128 different values to store actor objects.

8.2.2. ImageIndex

The ImageIndex defines which cell from the image will be used to display the MapItem in the Map Editor. This value is the same as the MapItem's number by default. You should not need to change it, unless you want two MapItems to use the same image in the Map Editor.

8.2.3. RawData

RawData defines the value that will actually be placed in the map's Info layer. This is the value that the game will read from the map. By default, this is the MapItem's number. If the current MapItem is supposed to represent an actor object, you should not modify this value.

If you want to use a MapItem to place information in the Info layer instead of representing an actor object, you need to enter that information as the MapItem's RawData value. The highest (leftmost) byte of the RawData value defines the type of the information, so this leaves "only" the three lower bytes for the actual information. That is why the MapItem's number is not allowed to exceed the limits mentioned above.

Byte	**Info Type**
00_{16}	MapItem
01_{16}	Coordinates
02_{16}	WorldMap Stuff
03_{16}	Messages
05_{16}	High scores

Table 8.2.: Info Types

8.2.4. Skill

The skill value indicates at which difficulty levels the game will create an actor object from the MapItem. The information is encoded as bit flags in a byte

value, which means the MapItem format supports up to eight different skill levels.

The lowest (rightmost) bit indicates if the object will be spawned at the easiest difficulty setting. The higher bits represent the higher difficulty levels. So the default skill setting indicates that the object will be spawned at any possible difficulty level.

If you want to make an object (like an extra life, for example) only appear on the easiest difficulty level, you need to set the skill value to 00000001_2. If you want to make an object (like a strong enemy) only appear at the third difficulty level or above, you need to set the skill value to 11111100_2.

Keep in mind that the skill setting is used for every actor class, including the player classes. This allows you to create different starting locations for each difficulty level. If you decide to do that, you need to make sure there is no overlap in the skill bits of each starting location. Otherwise, the game will try to place more than one player in the level. In that case, only the last player object read from the map will actually be placed in the level.

8.2.5. Direction

The direction value allows you to define the initial direction for the actor. You can use any of the values listed in table 8.3.

Value	Direction
0	LEFT
1	RIGHT
2	UP
3	DOWN
4	UP & LEFT
5	UP & RIGHT
6	DOWN & LEFT
7	DOWN & RIGHT

Table 8.3.: Directions

8.2.6. Class

This stores the name of the actor class. You can use actual classes (hard-coded in the engine) as well as "virtual" actor classes (see chapter 7).

8.2.7. Info

This allows you to define an info string for this MapItem. This text will be displayed in the Map Editor when you move the mouse cursor over this MapItem. Enter a short description of the MapItems in here to make editing maps a little easier.

You should enter a short description containing the actor class, skill and direction settings, as the Map Editor will not display the individual settings for each MapItem.

8.3. Editing The MapItems

8.3.1. Changing The Data

The MapItem data is stored in a `.ini` file (currently `game.ini`, but that may change in future versions of the engine), so they are easy to edit.

First, the maximum amount of MapItems has to be defined in the `.ini` file, so the engine knows how many MapItems need to be read from the file. You do this by writing a section of the following structure:

```
[MAP ITEMS]
max item = 64
```

This allows you to define up to 64 MapItems (numbered 1 to 64).

The individual MapItems are then declared like this:

```
[MAP ITEM 1]
info = Keen (facing right; skills 2&3)
image index = 1
class = TKeen
skill = %110
direction = 1

[MAP ITEM 18]
info = Coordinates
data = $01000000
```

```
[MAP ITEM 19]
info = Map Entrance
data = $0200000E
```

The engine will automatically use the default values for any option that is not declared in the `.ini` file, like the "ImageIndex" values in the examples above.

Notice that you are able to write numbers in binary or hexadecimal format (as seen in the examples above) instead of having to convert all numbers into decimal format. To store a binary number, the value must start with the percent symbol `%`. A hexadecimal value needs to start with the dollar symbol `$`. These symbols must be placed directly at the beginning of the number with no spaces, otherwise the number will be set to 0.

8.3.2. Changing The Graphics

The Map Editor will use the file `Enemies.png` located in your `Graphics\Enemies` subdirectory. You can use any image manipulation software to edit this image, as long as the transparency stays intact.

If you wish to add new cells to the image, you should keep the image width intact and only increase the image height to add new cells at the bottom of the image. This will make sure that the existing MapItems are not drawn using the wrong parts of the image.

9. Text Format

9.1. About The Text Format

Even though this is talking about a "Text Format", this format stores more than just plain text. The text format is used by the help and story texts in the game and can display text in different colors, along with images.

If you have been working with the text format used by **Commander Keen 4-6** or **Wolfenstein 3-D**, you should have no problems getting used to the format used by KE**EN**GINE. The syntax is mostly identical.

9.2. Syntax

The text format recognizes certain commands that all begin with the ^ symbol. The commands have the following structure:

Command	**Name**	**Effect**
^P	Page	Marks the beginning of a new page
^E	End	Marks the end of the text file
^G<y>,<x>,<n>	Graphic	Inserts the image <n> at position (<x>,<y>)
^C<x>	Color	Sets the text color to <x>
^/	Comment	Turns the line into a comment

Table 9.1.: Text Commands

Except for the Color command, each command must be located at the beginning of a line to have an effect. If a line starts with a Page or End command, the rest of the line will be treated as a comment and thus ignored by the Text Viewer. The same applies to the comment command.

9.2.1. The Page Command

For a text file to be read properly by the Text Viewer, the text file needs to contain at least one page. That means the first line of every text file should contain a Page command. You could also have comments in the first line, but you cannot have any other text or empty lines in the text file before the first Page command.

The rest of the line containing the Page command will be ignored. This allows you to type `^PAGE` instead of just `^P`.

9.2.2. The End Command

Unlike the **Commander Keen** text format, the End command is not actually required in KE**EN**GINE's text files. However, it is recommended to put an End command at the end of the last page of your text file.

Everything after the End command is ignored.

9.2.3. The Graphic Command

This draws an image file at the coordinates given by the `<x>` and `<y>` arguments on the current page. The actual values of `<x>` and `<y>` must be decimal numbers and the entire command must not contain any spaces. Everything following after the last comma of the Graphic command will be treated as the `<n>` argument, so be sure not to put any text in the same line as the Graphic command.

The image file `help_image_<n>.png` located in your `Graphics\Help` subdirectory will be used, where `<n>` can be a number or any string of characters allowed in a file name. Inserting an image will adjust the left and right text margins accordingly, so text and images will never overlap.

9.2.4. The Color Command

As mentioned before, the Color command can be used at the beginning of a line as well as anywhere else in a line. The only requirement is that there must be a space before the Color command if it is not at the beginning of a line.

The Color command consists of exactly three characters. The first two characters are `^C` and the third character is the hexadecimal value of the desired color index.

Any characters following the Color command will be drawn as text, using the new text color.

Changing the text color is permanent and will carry over to all following lines and pages. It is recommended that you put a Color command at the beginning of every page just to make sure the text will be displayed using the correct color.

9.3. Sample Page

Figures 9.1 and 9.2 show the text for one page and the result as displayed by the Text Viewer. Keep in mind that this is merely one single page and not an entire text file.

```
^P
^G10,230,dopefishlives
^G160,10,k1n9duk3
^CFEINBINDEN VON BILDERN
^CE
Mit der GRAPHIC-Anweisung "^G<Y>,<X>,<N>" können Grafiken
eingebunden werden.  <N> steht dabei für die laufende Nummer der
Grafik, <X> und <Y> für die Position, an der das Bild dargestellt
werden soll.

So bewirkt "^G0,220,10", dass die Bilddatei "help_image_10.png"
geladen und an der Position (220,0) - also in der rechten oberen
Ecke des Bildschirms - dargestellt wird.

Wie bereits zu erkennen ist, wird der für den Text verfügbare
Platz automatisch von den eingefügten Bildern begrenzt.  Dies
erleichtert das Erstellen und Gestalten der Textdateien
zusätzlich.
^P
```

Figure 9.1.: Sample Page – The Text

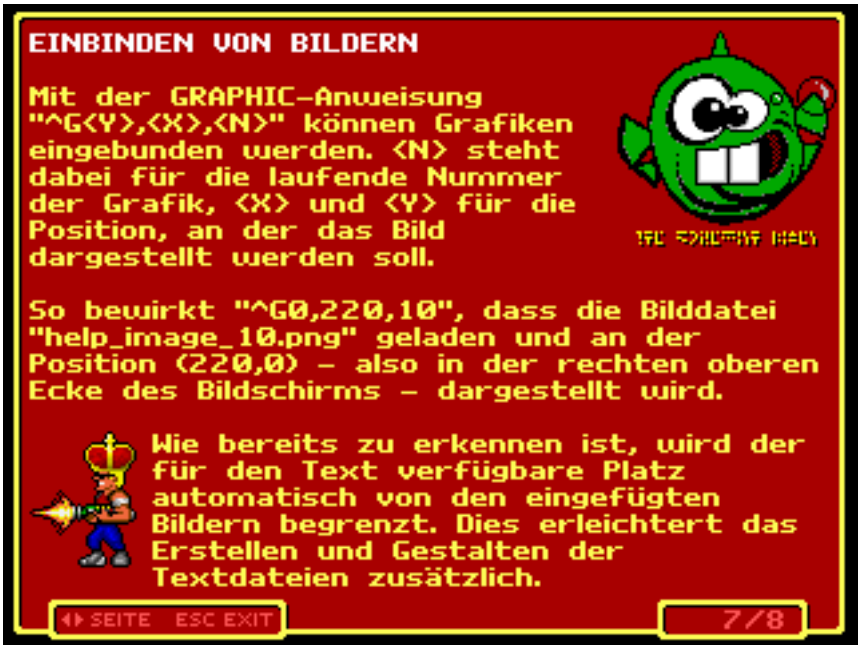

Figure 9.2.: Sample Page – The Result

Appendix

A. FAQ

Q *Who came up with this stupid name for the engine?*
A That was me! — K1n9_Duk3

Q *Where can I get this BlitzMax-thingy?*
A `http://blitzbasic.com` would be worth a look.

Q *Do I really have to pay for BlitzMax?*
A Nope. BlitzMax is free software now.

List of Tables

List of Figures