

**GGZC**

*The Poor Man's Interactive Fiction Player*

Bob Newell

Futrezo Software Solutions

Release 0.9991

October 22, 2012

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>What is GGZC?</b>	<b>3</b>
2.1	The Poor Man's Interactive Fiction Player . . . . .	3
2.2	GGZC vs. Gargoyle . . . . .	3
2.3	Why Bother with GGZC? . . . . .	3
<b>3</b>	<b>Prerequisites</b>	<b>4</b>
3.1	Requirements for Running GGZC . . . . .	4
3.2	Requirements for Building GGZC . . . . .	5
<b>4</b>	<b>Using GGZC</b>	<b>5</b>
4.1	Basic Use . . . . .	5
4.2	Game History . . . . .	6
4.3	A Little Organization . . . . .	6
4.4	Save Files, Password Protection, and Errors . . . . .	6
4.4.1	Save Files . . . . .	6
4.4.2	Password Protection . . . . .	7
4.5	Errors and Backups . . . . .	7
<b>5</b>	<b>Building GGZC</b>	<b>7</b>
5.1	Building the Interpreters . . . . .	7
5.2	Assembling the Package . . . . .	8
5.3	Password Protection of GGZC . . . . .	8
<b>6</b>	<b>Disclaimer</b>	<b>9</b>
6.1	Ownership Claims and Disclaims . . . . .	9
6.2	License . . . . .	9
6.3	No Warranty Given; No Liabilities Accepted . . . . .	9
<b>7</b>	<b>Contact Information</b>	<b>10</b>
<b>8</b>	<b>Change Log</b>	<b>10</b>

# 1 Introduction

This current effort should be considered a preliminary manual for GGZC, and likely always will be. GGZC is all about minimalism and that is the perfect excuse for failing to provide adequate, detailed documentation.

## 2 What is GGZC?

### 2.1 The Poor Man's Interactive Fiction Player

GGZC is accurately sub-titled “The Poor Man's Interactive Fiction Player.” Inspired by the high-end *Gargoyle* player, GGZC is a poor cousin which nevertheless serves specific purposes, as will be seen.

### 2.2 GGZC vs. Gargoyle

GGZC incorporates 12 different interpreters for over 12 different types of interactive fiction “story files” or games (some interpreters handle multiple file types). With one notable exception, all of these interpreters are configured to run in either “console” or “term” mode, using character based output, fixed width fonts, and a bare minimum of text styling (bold, colors, and reversed text are pretty much the limit of this repertoire). In contrast, Gargoyle uses attractive proportional fonts and can render various font sizes and styles, as well as graphics, sound, etc., running in a window.

GGZC will only run on Linux systems, whereas Gargoyle runs on Linux, Windows, Mac, and probably more.

### 2.3 Why Bother with GGZC?

So, given that GGZC does less, does it less well, and is limited to Linux, why even bother?

If you're not a Linux user, if you're not interested in the peculiar charm that minimalism offers, if you're not satisfied with a simple console-style display, go with Gargoyle. We're the first ones to say that Gargoyle is better in just about every way, from quality of coding to features to appearance to portability and breadth and depth.

Nevertheless, we did put some interesting things into GGZC.

- While GGZC accepts a game file name on the command line, it is not required. GGZC allows the user to traverse directories looking for a file of interest.
- GGZC allows the user to open archives (`zip`, `tar`, `bz2`, `gz`, and combinations thereof) to look for and select game files *inside* the archive. After a session is completed, any new

save game files will be incorporated into the archive for later use. (This functionality relies on recursive calls to GGZC and as such is klunky in its current implementation.)

- GGZC will open password-protected zip archives (if you have the password, of course) and will rebuild them with the same password or a new password.
- Wherever possible (and it's possible in most cases) the game interpreters are built with `libncursesw` to allow for Unicode input and output. While this is not well-tested (actually none of GGZC is well-tested) the point is to allow for playing non-English games. (And, in fact, the `fizmo` interpreter is provided instead of `frotz` for this reason.) Further to this point, again when possible interpreters are built with a version of `glkterm` built with `libncursesw`. (Only `fizmo` and `frob` don't meet this specification, but `frob` is built with the `ncursesw` library.)
- GGZC is built as a single, complete executable file, with all scripting and game file interpreters built in. GGZC can itself be built in password protected form (you need the source package to do so).
- GGZC is eminently suitable to put on a thumb drive, particularly a bootable thumb drive; we have such an installation currently on a 2GB drive booting to Puppy Linux 4.31 (at present writing).
- GGZC will run without X-windows in a console (except for DOS games). This could be of interest if you put a very minimal console-only Linux system on a very old computer (we had an ancient Gateway Handbook, for instance, that ran console-only Linux).

## 3 Prerequisites

### 3.1 Requirements for Running GGZC

GGZC must be run on a “relatively” modern Linux system. While we haven't gathered all the exact details, you will need the following (which we find is found on every system we've tried, but we can't make guarantees).

- The `bash` shell.
- `libc6`.
- `libsdl`.
- `libncursesw` (not just `libncurses`).
- If you wish to run DOS games, you'll need DOSBOX and X-Windows, even though that is a bit counter-intuitive for a minimalist effort. (You might, with some hacking, be able to run with DOSEMU in a console.)

- An important requirement that’s easy to miss: you must have full rights in the directory in which you run your game file as well as full rights to the game file or containing archive as well. You won’t be able to open an archive, for instance, in a directory in which you don’t have write access.

## 3.2 Requirements for Building GGZC

Building GGZC has a lot more in the way of requirements. In addition to the run-time requirements, you’ll need a full development environment. Here are some of the things necessary.

- C and C++ build environments.
- Library header files (the “dev” packages) for at least the following, probably more:
  - `libsdl`.
  - `libsnd`.
  - `libncursesw`.
  - `libxml1`.
- A  $\text{\LaTeX}$  installation to build the documentation.

## 4 Using GGZC

### 4.1 Basic Use

Using GGZC, by design, is the ultimate in simplicity. After you’ve unzipped the distribution file `ggzc.zip` then at the command line, in the same directory, in a console or term, run

```
./ggzc
```

and that’s all there is to it. You will be presented with a simple text mode menu showing either the current directory or the “last used” directory, subdirectories, recognized archive files, and recognized game files. If you select a subdirectory or parent directory, a new menu appears with the entries in that location. If you select an archive, GGZC attempts to open it, and if successful, shows a menu with the archive entries. If you select a game file, GGZC tries to play the game file with the correct interpreter.

(GGZC’s concept of the “last used” directory is the last directory in which a game was actually played. To save time in directory navigation, GGZC returns to this directory at its next invocation.)

You can run up and down directories all you wish, open archives and even nested archives, and move around as much as you like, but in the end the idea is to select a game file and play it!

If you wish, you can select a game file directly on the GGZC command line, for instance

```
./ggzc jigsaw.z8
```

will start the game *Jigsaw* with no further interaction needed (as long as `jigsaw.z8` is present in the current directory, of course).

If you change your mind, typing `q` at any menu display will exit GGZC.

## 4.2 Game History

Typing `h` at any menu display will show the ten most recently played games, with the most recent at the top and the least recent at the bottom (duplicates are removed). At this menu any one of these recent games can be selected for immediate play. This can be a time-saver if you wish to play the same game over again, or you'd like to find a recently played game but you don't remember the exact name or directory location. Note that game file names entered on the command line are not saved to history.

## 4.3 A Little Organization

You might want to put a menu item in your system for GGZC. You also may wish to copy the `ggzc` executable file to somewhere on your command path, such as possibly `/usr/local/bin`. However, we recommend that you create a directory containing all your interactive fiction files. You might wish to organize it as one zip file per game title, or one subdirectory per game title. It's up to you. (Again, since playing from archives is a little kludgy at the moment, this isn't that highly recommended.) Then put `ggzc` at the top level of the interactive fiction directory.

In our installations, we create `/usr/local/games/if`. Then `ggzc` goes in that directory, and we have one subdirectory for each game title or group of titles, for instance, `jigsaw`, `edifice`, `infocom`, `plant`, `tzero`, etc. (If you *really* need to save space use zip files instead of subdirectories, although this isn't the best idea at present.) We then add a menu entry in the "games" section of our X-window menu system, although it's easy enough to go to a console or term, change to the interactive fiction directory, and run `./ggzc`.

## 4.4 Save Files, Password Protection, and Errors

### 4.4.1 Save Files

If you play a game file that is contained within an archive, and create some save game files, at the end of your session GGZC will try to rebuild the archive, incorporating the save files within it, so that they are available for your next session with that game. It's up to you to name save files so that they don't overwrite one another; GGZC does not provide warnings if that is about to happen.

At any GGZC menu display, typing `s` will give a list of save game files in that directory which have a suffix of `.sav`. You can name save game files any way you like, of course, but they will only show up in the listing if named according to the `.sav` convention.

#### 4.4.2 Password Protection

Presently, GGZC supports password protection for zip files (only), and it supports it in a somewhat peculiar manner. If a zip file is password protected, you are prompted for the password when GGZC tries to open it. If you get the password right, everything proceeds as usual. When your session is done, GGZC tries to rebuild the zip file to incorporate save files. (By the way, GGZC is dumb about this; it tries to rebuild the archive even if you didn't create new save files at all.) Upon starting the rebuild, GGZC will ask you for a password to protect the rebuilt archive. You can give the same password as the original archive password, if you so desire, or give an entirely different password or even none at all. It's up to you. GGZC does *not* save the original password at any time.

### 4.5 Errors and Backups

GGZC tries to control its reactions to errors with error traps and tests. For instance, it won't try to overwrite a protected zip file for which you didn't have the original password, and takes similar precautions in other potentially risky situations. But there is always inherent danger, especially when rebuilding archives. While obviously we find that this works correctly nearly all the time, the operative word is *nearly* and there are never any guarantees against data loss. You are advised and counseled to make backup copies of anything that you don't want to lose, or some day you will lose it.

## 5 Building GGZC

To build GGZC itself, you need the source package `makeggzc.zip` which should be available here:

<http://www.bobnewell.net/filez/makeggzc.zip> .

Unzip the package in a convenient place, thereby creating the directory `makeggzc`.

### 5.1 Building the Interpreters

Building the GGZC interpreters package is not for the faint of heart or the debutante. If you're not familiar with the process of compiling packages on a Linux system, this is not for you. The information below will speak to the knowledgeable person and isn't intended to be a beginner's tutorial.

A complete build of GGZC requires first building the interpreters from source code. Go into the subdirectory `interpreters` and run

```
./build.sh .
```

This will build the `glktermw` library, all of the interpreters, and will compile the `LATEX` documentation into a PDF manual. Watch the output carefully and be sure that everything gets built without error (though there will be many, many warning messages).

Errors are almost always due to missing packages (usually library header files). It will be fairly clear what package is missing from the error messages. Use your package manager to install the required “dev” package(s) and try again.

There is sometimes a problem with Linux library headers. This will be obvious if it occurs, and you may have to search around your system for things like `stddef.h` and others, and copy them to `/usr/include`. Again, if this happens, you’ll see it easily.

If all goes well, in the end you’ll have a new `terps.zip` file in the `ggzc.dir` subdirectory, with all the interpreters packaged together.

We learned to our chagrin to build the distribution version of GGZC on a Linux system that is likely to be at least as old as the target system(s). This avoids unfortunate library compatibility problems. New libs will most likely be backwards compatible, but old libs won’t have the right symbols at runtime if you’ve compiled on a different, newer system.

## 5.2 Assembling the Package

In the `makeggzc` directory simply run

```
./gmake.sh
```

and you’re done. The executable `ggzc` will be built and packaged with this documentation file and put into `ggzc.zip` just above the level of the `makeggzc` directory. A new `makeggzc.zip` will also be put there, potentially overwriting your downloaded version if you’re not careful!

The executable `ggzc` contains both the runtime scripts and the interpreters package. It is assembled with a modified version of the excellent `makeself` package.

## 5.3 Password Protection of GGZC

There is an option in the `gmake.sh` script to password protect the GGZC executable itself. This is documented in the script; edit `gmake.sh` and you’ll see how it’s done rather easily. A protected version of `ggzc` is about 20% larger than an unprotected version.

Why would you want a protected version of GGZC? We’ll only say that it would be for the same reason that you would want protected zip file versions of the game files themselves. If you’re following our meaning here, all well and good; if you’re not following it, even better.



## 6 Disclaimer

### 6.1 Ownership Claims and Disclaims

Portions of GGZC acquired from others, such as `makeself`, the interpreters, etc., are the property of their respective owners and are used under licenses specified by the owners.

Original portions (those portions created by us, including but not limited to the GGZC scripts) are Copyright (C) 2007-2012 Futrezo Software Solutions, a division of Mr. Fred Investments.

### 6.2 License

We grant you an unlimited, no-cost, perpetual, non-exclusive license to use GGZC in any legal manner you wish; however, we do not give up any of our rights in GGZC, and you must respect the licensing terms and rights of the owners of those portions of GGZC not created by us.

In practice, this means you can modify, give away, sell, or do anything you want with GGZC as long as you don't break any laws or violate the license terms of the owners of those portions of GGZC that we didn't create; and you can't take GGZC away from us.

### 6.3 No Warranty Given; No Liabilities Accepted

We make no claims that GGZC represents good software engineering practice in any way; in fact, we claim the reverse, that the portions of GGZC developed by us are hackish to say the least.

YOU ARE ADVISED THAT GGZC IS OFFERED AS IS, WITH NO WARRANTY OF ANY KIND, INCLUDING WARRANTIES OF FITNESS FOR INTENDED PURPOSE.

YOU USE GGZC AT YOUR OWN RISK AND THE AUTHORS ACCEPT NO LIABILITY OF ANY KIND, WHETHER DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, OR OTHERWISE, UNDER ANY LEGAL THEORY WHATSOEVER, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. YOU MUST EVALUATE THE SAFETY AND USABILITY OF GGZC IN YOUR OWN ENVIRONMENT AND MUST ACCEPT ANY AND ALL RISKS.

YOU ARE WARNED THAT GGZC CAN AND WILL CAUSE DATA LOSS AND YOU ARE RESPONSIBLE FOR BACKING UP YOUR DATA PRIOR TO USING GGZC AND YOU AGREE TO ASSUME RESPONSIBILITY AND LIABILITY FOR ANY AND ALL RISKS OF DATA LOSS AND/OR ACTUAL DATA LOSS, CORRUPTION, INAVAILABILITY, ETC.

IF YOU DO NOT ACCEPT THESE TERMS AND CONDITIONS YOU MAY NOT USE GGZC. YOUR USE OF GGZC CONSTITUTES YOUR FULL ACCEPTANCE OF THESE

TERMS AND CONDITIONS.

## 7 Contact Information

If you find GGZC useful, we'd certainly like to hear from you. Likewise, if you find bugs or have suggestions for improvement, we'd also like to hear from you, though strictly speaking we can't make any promises about fixing bugs or developing new features. You can reach us at `ggzc@bobnewell.net` .

If you hate GGZC, you can tell us that too, although it would be more constructive to also tell us *why* you hate it or don't find it useful.

If you feel we've used any code, package, or other material (which is your property) in an unallowed or improper manner, please let us know at once and we'll work with you to resolve the situation as appropriate.

If you want to tell us that GGZC represents poor practice, poor software engineering, or the like, you can do so, but we already know that.

If you want to make some sort of fantastic job offer, bring it on!

## 8 Change Log

The GGZC change log is no longer maintained within the `ggzc.sh` script, and has been moved to the file `changelog.txt`.