IF Authoring System Developer's Guide

version 0.99, 2005-11-27

Peer Schaefer (peer@wolldingwacht.de)

The Interactive Fiction Authoring System Developer's Guide (version 0.99, 2005-11-27), which is for programmers who want to create an authoring system for Interactive Fiction.

The most recent version of this document is—without guarantee—available at http://www.wolldingwacht.de/if/if-auth-dev-guide.html. In the same place this document should available in alternative file-formats (PDF, Postscript PS, Info, HTML, plain ASCII-text, DVI, and Texinfo-source). If you have problems getting these files please contact me via peer@wolldingwacht.de or—if that fails—via peerschaefer@gmx.net or peer.schaefer@hamburg.de.

Table of Contents

1 Preface

This document is not for authors of Interactive Fiction who want to *use* an authoring system to create Interactive Fiction; this document is for programmers who want to *create* an authoring system for Interactive Fiction—that is a system which allows authors of Interactive Fiction (a.k.a. "Text Adventures") to write Interactive Fiction. In other words: this document is about writing a system like Inform[1] or TADS[2] (so, we're on a pretty abstract meta-level here). I will try to lay down the most important aspects that the developer of an authoring-system should consider, and the most important pitfalls she should avoid.

So, what qualifies the author of this document to write something substantial about the matter? Well, maybe not very much. Long time ago I wrote a *very* simple C compiler for the Amiga 500 and a cross-platform assembler for the i386 PC (generating code for the C-64, C-Plus/4 and Atari VCS 2600), and in the course of these projects I learned a little bit about compiler building. Or at least I think so. And I am playing Interactive Fiction for some years now. Some time in the past—before I learned that Inform and TADS are out there—I tried to write my own authoring system for Interactive Fiction. It took me about three or four months to realise that I had completely failed—and one month more to find enough courage to finally give up. In fact, I only just finished the user-interface and had no clue how to build a parser. In the end I learned much about the difficulties you step into if you persue such bold goal. Later I made myself familiar with Inform and TADS and learned a little bit by studying their inner functionality, with a focus on Inform. Currently I'm writing silly tiny adventures in Inform, but my small sparetime and my bad english prevent me from finishing anything worth publishing.

This document provides an advice and some extensive guidelines. The advice tries to prevent you from wasting your time. And the guidelines try to help you doing the job right. The target group of this document is propably very small, but maybe someone finds this document useful or instrumental in saving him much time.

Thanks to Roger Firth, Andrew Plotkin, Samwyse and Dan Shiovitz for their kind feedback and suggestions. I have taken freely from their replies. All errors and opinions remain—of course—my own. Thanks to Graham Nelson for Inform and Michael J. Roberts for TADS. And of course thanks to Will Crowther, Don Woods, Dave Lebling, Tim Anderson, Mark Blank and all the friendly girls and boys at rec.arts.int-fiction[3].

I am always happy about feedback, both cheers and criticism (including corrections of my bad english) or just saying hello, so please feel free to contact me via peer@wolldingwacht.de or—if that fails—via peerschaefer@gmx.net or peer.schaefer@hamburg.de.

---

[1] Inform: The Inform fiction compiler by Graham Nelson:
http://www.inform-fiction.org/
http://www.ifarchive.org/indexes/if-archiveXinfocomXcompilersXinform6.html
ftp://ftp.ifarchive.org/if-archive/infocom/compilers/inform6/

[2] TADS: The Text Adventure Development System by Michael J. Roberts:
http://www.tads.org/
http://www.ifarchive.org/indexes/if-archiveXprogrammingXtads2.html
http://www.ifarchive.org/indexes/if-archiveXprogrammingXtads3.html
ftp://ftp.ifarchive.org/if-archive/programming/tads2/
ftp://ftp.ifarchive.org/if-archive/programming/tads3/

[3] The newsgroup about writing (authoring) Interactive Fiction:
rec.arts.int-fiction
http://groups.google.de/groups?hl=en&lr=&ie=UTF-8&group=rec.arts.int-fiction

2 The Advice

So, you want to write an Interactive Fiction authoring system? Ok, here's my advice: don't do it. At least, think twice about it. What's wrong with Inform or TADS? In what respect would your system be better? And is your point important enough to justify the effort? And after you've thought twice, think twice again. Inform and TADS are around and should suit your needs. Both may have some minor or major flaws in design and implementation. They are not perfect. But they are pretty good and almost bug-free. They have been intensively tested. They are well documented. They are very flexible. They have been ported to many different platforms. They are widely spread, well discussed and well known, so there is a good chance that some guy at rec.arts.int-fiction[1] can help you with your programming problem. Many add-ons or contributions are available for both. Even if your brand new system would be twice as good as Inform or TADS, both Inform and TADS would still be ten times better documented, a hundred times better discussed and a thousand times better tested. So it's not only a matter of quality. Your system should provide some really new and cool features, or it would be totally pointless.

And if you have thought twice and then twice again, and you are still willing to create your own Interactive Fiction authoring system, I have another advice for you: Make sure that you have a huge amount of sparetime. You *can not* underestimate the time this project will take. If you have a job and a girlfriend[2], cancel both (no, not really—that was a joke).

And one more advice: don't think that anybody will pay for your product. No exact numbers are known, but there are probably only a few tenthousand people worldwide who *play* Interactive Fiction, and there are only a hundred people or so worldwide who *write* Interactive Fiction. Since Inform and TADS are free and nobody gets money for writing Interactive Fiction,[3] nobody will pay for an authoring tool.

I would also encourage reading the article "So you want to write a text adventuring authoring system..." by Alan Conroy in the XYZZY magazine issue #14[4]. It is perhaps not strikingly brilliant but it is—as far as I know—the only report about developing an IF authoring tool that was written by someone who really *did* it (and wasn't just talking about it, like myself).

---

[1] The newsgroup about writing (authoring) Interactive Fiction:
rec.arts.int-fiction
http://groups.google.de/groups?hl=en&lr=&ie=UTF-8&group=rec.arts.int-fiction

[2] Or a boyfriend.

[3] Peter Nepstad may be the great exception nowadays, since his "1893 - A Worlds Fair Mystery" scored some economic success. For details take a look at http://www.illuminatedlantern.com/1893. By the way: Peter Nepstad used the TADS authoring tool by Michael J. Roberts for creating "1893".

[4] "So you want to write a text adventuring authoring system...", by Alan Conroy, in XYZZY magazine issue #14:
http://www.xyzzynews.com/xyzzy.14i.html
http://www.ifarchive.org/if-archive/magazines/XYZZYnews/XYZZY14.PDF

3 The Guidelines

3.1 Set up your goals

Think about what you want. What's wrong with Inform or TADS? What's missing or what's unsatisfying? Are the flaws important enough to write a complete new system? Maybe you could contribute to an existing system, either by providing a library add-on or by providing a patch to the compiler. Be careful and watch your motives. If you think that Inform or TADS are simply not elegant enough, and if that's the only point, than you are both right and wrong: right regarding the lack of perfect elegance in these systems, but wrong in thinking that this is a good reason to create a new system. It's not.

Instead make a list of all the reasons you have to write your own system. Put the list away for two weeks, then pull it out and read it carefully. Are you still convinced by your own reasons?

3.2 Learn what's around

Make yourself familiar with

- Inform and TADS,
- Glulx[1],
- Glk[2].

Read the manuals of Inform and TADS and learn each language. Write at least one small game in each language, e.g. a simple three-rooms-game with a small puzzle. Take a look at the many different implementations of the sample game *Cloak Of Darkness*[3]. Read the documentation for Inform and TADS, especially the technical parts and the chapters about the library and the world-model. Take a look at the sourcecode of the compiler and of the library-files.

3.3 Design a language

Design a language for writing Interactive Fiction. There are—at least—two very different paths you can take:

a. A language that is oriented towards the more "traditional" programming languages like C, C++, Java, Perl, Pascal, Python, BASIC or something similar. This is the path Inform and TADS have taken, both of them beeing more or less C/C++-like.

b. A "language" that has a totally different approach. Possible examples could be:
   - A graphical, mouse-driven language of the kind "click-together-your-adventure-without-typing-a-single-word", e.g. by creating a graphically displayed "web of rules" with the mouse.
   - A "language" that uses a matrix-oriented database that you can fill with data and informations (like an Excel sheet or a form).

---

[1] Glulx: The Glulx virtual machine (standard) by Andrew Plotkin:
http://www.eblong.com/zarf/glulx/
http://www.ifarchive.org/indexes/if-archiveXprogrammingXglulx.html
ftp://ftp.ifarchive.org/if-archive/programming/glulx/

[2] Glk: Andrew Plotkin's Glk API, which provides a portable interface for text adventure systems:
http://www.eblong.com/zarf/glk/
http://www.ifarchive.org/indexes/if-archiveXprogrammingXglk.html
ftp://ftp.ifarchive.org/if-archive/programming/glk/

[3] The *Cloak Of Darkness* sample adventure, written in many different languages:
http://www.firthworks.com/roger/cloak/

- A HTML-like descriptive language (maybe an XML[4] variant) that allows you to design an adventure in much the same way you would design a document. There are already two authoring systems that follow that path:
  - AAS (Advanced Authoring System) by Roddy Ramieson (http://www.aas-ta.com);
  - Aiee! (An Interactive Environment Engine) by Mark Hughes (http://kuoi.asui.uidaho.edu/~kamikaze/Aiee/).

  Both are written in Java, which is pretty appropriate for systems using XML and allows "online playing" with your webbrowser.
- A language that implements a kind of "ladder-logic"[5].

If you take option (b), you are very much on your own. Such radically new approaches have the advantage that your system doesn't have to compete directly with Inform or TADS, because it's totally different and has other design-goals and a different target group resp. audience. But it also has at least two drawbacks: first, there are no or at least very few prototypes or examples you can orient yourself; and secound, there is—from my point of view—nothing as flexible as a "traditional" programming language; every other approach will almost sure be more rigid and more restricted, resulting in somehow "limited" games. On the other hand, a symbolic approach or a graphical interface would be less intimidating to non-programmers; that could open up a complete new "market" and would make the authoring of Interactive Fiction possible for a completely new group of people.

If you take option (a), your language should provide some very nice and cool "gimmicks" that are real improvements compared to Inform and TADS. It seems that Inform and TADS have probably saturated the market for C/C++-like languages for Interactive Fiction; it's hard to imagine another one in the same image having much to offer, so your "gimmicks" better have to be really cool. Possible "gimmicks" would be e.g.:

- linking in code written in another language (e.g. C/C++, Inform, TADS, Lisp, Pascal, etc.);
- a radically different world-model (e.g. a 3D simulationist (VR) approach where the author must provide X-Y-Z-coordinates and physical informations for each object);
- an Artificial Intelligence (AI) facility for NPCs (but in the end that could also be accomplished by writing a library-contribution for Inform or TADS and doesn't require a complete new system).

Your best bet for success is probably to find something that neither Inform nor TADS does very well, and make it so easy that everyone will slap their heads and say, "Why didn't we think of that?".

In the process of language-design itself you should choose an existing language as a prototype. Start from that point and "modify" the language whenever and wherever required, so that it in the end suits your needs. As hinted above, I would recommend C/C++ as a starting point, because it's a generic all-purpose language, powerful, widely used and almost a standard for programming. Other fair choices would be Java, Pascal or a standardized object-oriented variant of BASIC. I would discourage scripting languages (e.g. Perl, Python) because they were designed for different purposes and wouldn't fit very well. I would also discourage "ancient", seldom used or special purpose languages like Lisp,

[4] XML, the Extensible Markup Language:
http://www.w3.org/XML/

[5] "Ladder-logic":
http://en.wikipedia.org/wiki/Ladder_logic
http://www.google.com/search?hl=en&q=ladder+logic&btnG=Google+Search

Fortran, Forth, Logo, etc. unless you have a very good reason, because they lack some of the more powerful constructs and concepts, or they are difficult to learn because they are so different from what people nowadays are used to and comfortable with. Almost anybody with some programming experience knows C/C++, so learning a C/C++-like language will be very easy for most people; using Lisp, Fortran or Logo will force 90% of your audience to learn a completely new language from scratch, which wouldn't promote the success of your system. On the other hand, a complete new approach—e.g. the use of Lisp or Logo—could result in complete new solutions to old problems. Maybe that justifies the trouble and effort necessary for learning a new language (or re-learning an "ancient" one), but the surplus has better to be large—or your approach will fail. Or maybe your "strange" choice drives off 90% of your adience, but the remaining 10% could be people that were not attracted by all the other C/C++-like languages (e.g. newbies oder professionals with different preferences or just another attitude).

If you have chosen a language, customize and modify it with your goal —a language for implementing Interactive Fiction—in mind. Here are some suggestions:

- Interactive Fiction is all about text. Make (dynamic) strings a native data/variable-type of your language.
- Interactive Fiction is all about objects. Make the creation, movement and destruction/removement of objects as simple as possible. Make the creation and inheritance of classes as simple as possible. Make the management of the object tree as simple as possible. Design special commands, e.g. a special loop construct that loops over all child-objects (see Informs `objectloop`).
- Interactive Fiction is all about "states" like on/off. Make two-state 1-bit data (`boolean`) a native data/variable-type.
- Create a simple and transparent syntax for actions or processes that are typical for Interactive Fiction. For example, it's typical for Interactive Fiction to write some text to the screen and then prompt for a new player input. In Inform there is a special syntax for this: you can simply write the text in quotes and that's it. A simple quoted string (without command or function-call) means "write this string, append a new line/line feed, and then return the value `TRUE` to the parser". That's a damn clever feature.
- Make the language slim. Strip off everything that's not needed in Interactive Fiction. For example, most mathematical functions or file-handling routines are not needed. That will ease your pain and will reduce the number of bugs (in my opinion the number of bugs grows exponentially with the growth of the code `;-)`.
- Make all basic operations part of the language itself, don't provide them as libraries. For example, in standard C all I/O-functions are not part of the language itself but part of the standard library `STDIO.H`. That's ok for an all-purpose-language like C/C++, but not ok for a specialized language like yours. Make life simple for you and for all programmers that will later use your language.

I would strongly recommend to keep all I/O-functions Glk[6]-compliant. Glk is almost a standard nowadays (or should be), it is well designed, it keeps you from making mistakes or forgetting something, it is free and it makes it easier to write interpreters for your system or to port it to other machines and operating systems.

[6] Glk: Andrew Plotkin's Glk API, which provides a portable interface for text adventure systems:
http://www.eblong.com/zarf/glk/
http://www.ifarchive.org/indexes/if-archiveXprogrammingXglk.html
ftp://ftp.ifarchive.org/if-archive/programming/glk/

3.4 Develop a compiler

Develop a compiler for the language that you have designed according to Section 3.3 [Design a language], page 3.

Write the compiler in a widely used and standardized language that is available for many plaforms, at least for Windows, MacOS-X, and Linux/Unix. Write portable, good structured, well documented code. I suggest plain standard C (make sure that it can be compiled with gcc[7]). Using C would also allow you to use the Bison[8] tool, which can ease the pains of compiler-building by automatically writing a code-parser in plain portable C for you that parses your self-designed source language. Other fair choices are C++, Java or Perl. I would discourage the use of any language that is not available for free (like Delphi) or not standardized (like BASIC).

The compiler should generate platform-independent code. Glulx runtime-code would be a good choice, because it's specifically designed for Interactive Fiction, well documented, and free (and interpreters are available for several platforms). Another fair choice would be runtime-code for the Java-VM[9] or for the T3 Virtual Machine[10]. Other possible choices would include Java sourcecode, or Z-CODE[11] (V5, V6 or V8). C/C++-sourcecode as output might be acceptable, but keep in mind that many players of text adventures haven't used a compiler in their lives (at least make sure that the code compiles with gcc), and distributing compiled binaries would make the game platform-dependend. Poor choices would be assembler/machine code, BASIC, Fortran, Forth, and other languages that are not portable, not standardized, seldom used, less powerful, out of date or not available for free. And there is *really* no point in designing a new runtime-format, unless you have a *very* good reason.

3.5 Develop a library

Write some library files in your language (see Section 3.3 [Design a language], page 3) for use with your compiler (see Section 3.4 [Develop a compiler], page 6) that can simply by included in the author's sourcecode via `#include` or a similar command. Write clean, nicely formatted, well commented code. The library should provide two things:

- a parser,
- a world-model.

Both things are detailed below.

3.5.1 Develop a parser

The parser fetches the textual input from the player and translates it into a computer-readable form. For example, the parser should translate a player's command like `"OPEN THE WOODEN DOOR WITH THE GOLDEN KEY"` into a data-structure that contains the following elements: the internal ID for the `OPEN`-action, the internal ID of the `WOODEN DOOR` object and the internal ID of the `GOLDEN KEY` object.

---

[7] gcc: The GNU C/C++ compiler:
http://www.gnu.org/software/gcc/
http://www.delorie.com/djgpp/

[8] The GNU Bison parser generator:
http://www.gnu.org/software/bison/

[9] The Java(TM) Virtual Machine Specification:
http://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html

[10] The Specification of the T3 Virtual Machine:
http://www.tads.org/t3spec/intro.htm

[11] Z-CODE:
http://www.ifarchive.org/indexes/if-archiveXinfocomXinterpretersXspecification.html
ftp://ftp.ifarchive.org/if-archive/infocom/interpreters/specification

The parser should also handle placeholders like `"ALL"` or `"IT"` or `"EVERYTHING"`, so that commands like `"TAKE IT"` or `"TAKE EVERYTHING FROM THE TABLE EXCEPT THE BRASS LANTERN"` are solved and the parser passes the right IDs of the right objects to the game. Maybe the parser has to colaborate with the world model (see Section 3.5.2 [Develop a world model], page 7 below) to handle this, or the entire task has to be left to the world model; but these are intricate questions you have to make up by yourself.

See p. 1 and chapter IV of the *Inform Designer's Manual*, 4th edition[12].

3.5.2 Develop a world model

The world-model should provide some basic rules, for example that you can only see something when there is light, that some things (e.g. a bag) can contain other things, that you can't see or touch things that are inside other things (except when the `container` is `transparent` or `open`) etc.

The world-model should provide some default-behavior for all standard actions. For example, commands like `TAKE` or `DROP` should by default work as you might expect and should handle concepts like containment (inclusion inside of other things) and the "scope" (what you can see, touch and reach).

See p. 1 and chapter III of the *Inform Designer's Manual*, 4th edition.

3.5.3 Think about customizing

Keep in mind that the biggest problem that an author of Interactive Fiction is confronted with is *customizing the game*, which means overriding, changing, and redesigning library behavior. I'm not just talking about "hacking the library" here. As Andrew Plotkin once said on rec.arts.int-fiction, every line of code in a piece of Interactive Fiction exists to customize the library—even the simplest object description is a customization of the default "You see nothing special"-response. So your library should provide many entry points on many levels to allow the author to customize the behavior of the library from within in the code of his game, without having to modify the library files themselves. Freedom is good, and options are good. The author should have the power to override pretty much everything, as and when he feels like it. Of course, not overwhelming the author with a forest of infinite detail is also good. Thus, the library should provide factory-equipped default values for (almost) everything, so that the author only has to mess around with that what is special about the particular player input, line of grammar, verb, action, room, object or NPC.

A typical entry point should be a variable that is defined in and provided by the library (parser or world-model), which contains a pointer to a function/routine. The default value of the variable should be 0 (`NULL`), signaling the library that the default-behavior should apply. But if the author provides his own function/routine and lets the entry-point-variable point at it, the library calls that routine. If the routine returns 0 as a return value, the default-behavior still applies, but if the routine returns some other value the library knows that the routine has handled the matter, and the default-behavior is suppressed.

The library should provide entry points at as many stages and levels as possible (parsing level, grammar level, scope determination, action-resolving, action-verification, object reaction, post-action, etc.). Examples for good entry points can be found in the *Inform*

[12] The *Inform Designer's Manual*, 4th edition, by Graham Nelson:
ftp://ftp.ifarchive.org/if-archive/infocom/compilers/inform6/manuals/designers_manual_4.pdf
http://www.ifarchive.org/if-archive/infocom/compilers/inform6/manuals/designers_manual_4.pdf

Designer's Manual at pages 412ff. and 431ff., and in the appropriate sections of the TADS Manuals[13]. But—of course—there is always room for more...

3.6 Write a game or two

Write a piece of Interactive Fiction in your language, using your compiler and your library files.

Maybe it's a good idea to start with a small and simple game (two or three rooms with two simple puzzles, some furniture and at least one nice gimmick, e.g. a machine that does something). During the process of writing the game you will discover bugs in your compiler and in the library files. Correct them. But you will also notice some things about your language that are inconvinient or missing. Improve your language (and—in consequence—your compiler and/or library files).

Then write a large game (or let somebody write a large game and keep in touch with him closely). Some game that requires several hours to play through. For example, Inform was published together with the superb *Curses*[14], and that was crucial for the success of Inform because *Curses* proved that Inform can really produce a really large game of some quality. Writing your large game you will propably run into difficulties. Improve your language, your compiler, your library files. Maybe you will run into serious difficulties. Well, take a deep breath, go ahead and redesign your system—parser, world-model, common library, language syntax—, and try again.

3.7 Write manuals

Write detailed and accurate specifications (reference manuals) for your language (see Section 3.3 [Design a language], page 3), your compiler (see Section 3.4 [Develop a compiler], page 6) and your library-files (see Section 3.5 [Develop a library], page 6).

Write an easy-to-read tutorial that introduces a new programmer with some basic programming knowledge to your language, the compiler and the library files.

Provide the reference manuals and the tutorial at least in PDF, Postscript (PS) and HTML-format. Maybe it's a good idea to use the Texinfo[15]-standard that can easily produce nicely formated PDF, PS and HTML output from one document-source. Using DocBook[16] is also an option.

3.8 Beta test your work

Don't publish your work now. Beta test it first. Inform two or three competent girls or boys from rec.arts.int-fiction about your project and ask them for a first beta-test. If they are interested, give them a FTP or HTML download location where your work is available for download. Let them test your compiler, your library-files and your manuals. Listen to them. Take them serious. Take their advices. It's much, much work, it's boring and it's a nuisance. But it's neccesary.

---

[13] The TADS Manuals, by Michael J. Roberts and Eric Eve:
http://www.ifarchive.org/indexes/if-archiveXprogrammingXtads2Xmanuals.html
http://www.ifarchive.org/indexes/if-archiveXprogrammingXtads3Xmanuals.html

[14] *Curses*, by Graham Nelson.
http://www.ifarchive.org/indexes/if-archiveXgamesXzcodeXcurses.z5
ftp://ftp.ifarchive.org/if-archive/games/zcode/curses.z5

[15] Texinfo: The GNU Texinfo system for creating documentations:
http://texinfo.org/
http://www.gnu.org/software/texinfo/

[16] DocBook: an XML/SGML vocabulary particularly well suited to books and papers about computer hardware and software:
http://www.oasis-open.org/docbook/

3.9 Publish and monitor

Upload your system to the IF-archive[17] and announce it at rec.arts.int-fiction. Don't fall into any illusions; half of the work is still ahead you.

Prepare for a flame-war. The use of computer languages or development systems is a religious question, and there are many priests out there. Don't care about unpolite, unfriendly or unspecific criticism ("Your system sucks!"). But do care about criticism that points at specific aspects or makes specific suggestions (these should be the majority, since rec.arts.int-fiction is usually a friendly place with intelligent and helpful people).

- Make a list of all bugs. Monitor rec.arts.int-fiction and your mailbox for several weeks and update the bug-list constantly.
- Make a list of all improvements that (a) you by yourself think would be good or (b) were suggested in the public discussion of your work. Wait some time and update the wish-list constantly.
- Take the bug-list and the wish-list and make a ranking for each list, with the most urgent/important bugs/wishes first. Work your way through both ranking-lists, starting at the top with the most urgent/important bugs/wishes. Don't forget to adjust the manuals if you change something in your authoring system.
- During your way through the bug/wish-list you should publish every three or four weeks or so a new version of your work, by uploading it to the IF-archive and announcing it at rec.arts.int-fiction. Don't publish too often–give your beta-testers enough time to test your suite, to discuss it, and to respond.

3.10 Check for success

Wait a few months. Monitor rec.arts.int-fiction. Monitor your mailbox. Does your authoring system have any audience? Is there a circle of at least a few users that use your system? If yes, keep on working on your system.

If nobody uses your system, if you are the only one who writes postings to rec.arts.int-fiction about your system, you should stop working *now*. You have wasted your time. Sorry, bad luck. Maybe your system is not good enough. Maybe your system has some fundamental flaws. Maybe your system is pretty good but doesn't provide any substantial improvements in comparison to Inform or TADS. Well, never mind. Hell, there is more to life than programming an Interactive Fiction authoring system. Keep your mood up and think about all the programming experience that you have made. Go fishing. Get a girlfriend[18]. And, of course—don't loose your interest in Interactive Fiction.

3.11 Feed back

Send me an e-mail and tell me about your experiences and what you think about these guidelines. Correct my bad english. Thank you.

---

[17] IF-archive: The central repository for software related to Interactive Fiction:
`ftp://ftp.if-archive.org/`
`http://www.ifarchive.org/`

[18] Or a boyfriend (I'm repeating).

4 Some final notes

4.1 Conclusion

I think we can close with some conclusions:

- *Avoid delusions.* Writing a system that is greater, better, more elegant and more powerful than any existing system is propably a little bit over the top (in other words: overkill).
- *Find a niche. Create a gimmick.* Instead, make your system special. Make something nobody has. A very special user-interface. A cool new object-model. I don't know. Something that kicks.
- *Standards matter.* Use existing standards (for example Glk or Glulx) whenever and wherever possible.
- *Portability matters.* Use portable languages, libraries and techniques. Don't lock yourself to a specific platform. Portability to MacOS-X, Unix/Linux and Windows is the subsistence level; "generic" programming (portability to almost anything that adheres to a minimum set of standards) is preferable.
- In general: *Don't invent the wheel a second time.* Use existing libraries, formats, languages etc. whenever and wherever possible. Think about using Glk, Glulx, and such stuff.
- *Don't underestimate it.* Make sure you have enough resources (that is, enough spare-time, IF-experience, and programming-skills).

Reading this guide one might get the impression that the author wanted to prevent any new authoring systems from emerging. You couldn't be more wrong. In fact, I would very like to see some new stuff coming up. But I hate wasting time.

4.2 Credits

Thanks to Roger Firth, Andrew Plotkin, Samwyse and Dan Shiovitz for their kind feedback and suggestions. I have taken freely from their replies. All errors and opinions remain—of course—my own. Thanks to Graham Nelson for Inform and Michael J. Roberts for TADS. And of course thanks to Will Crowther, Don Woods, Dave Lebling, Tim Anderson, Mark Blank and all the friendly girls and boys at rec.arts.int-fiction.

4.3 Getting in touch

The most recent version of this document is—without guarantee—available at `http://www.wolldingwacht.de/if/if-auth-dev-guide.html`. In the same place this document should available in alternative file-formats (PDF, Postscript PS, Info, HTML, plain ASCII-text, DVI, and Texinfo-source). If you have problems getting these files please contact me via `peer@wolldingwacht.de` or—if that fails—via `peerschaefer@gmx.net` or `peer.schaefer@hamburg.de`.

Appendix A Copying This Manual

A.1 GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero

Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

   B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

   C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

   D. Preserve all the copyright notices of the Document.

   E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.1.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendix B Table of links

1. Inform: The Inform fiction compiler by Graham Nelson:
   http://www.inform-fiction.org/
   http://www.ifarchive.org/indexes/if-archiveXinfocomXcompilersXinform6.html

   ftp://ftp.ifarchive.org/if-archive/infocom/compilers/inform6/

2. TADS: The Text Adventure Development System by Michael J. Roberts:
   http://www.tads.org/
   http://www.ifarchive.org/indexes/if-archiveXprogrammingXtads2.html
   http://www.ifarchive.org/indexes/if-archiveXprogrammingXtads3.html
   ftp://ftp.ifarchive.org/if-archive/programming/tads2/
   ftp://ftp.ifarchive.org/if-archive/programming/tads3/

3. rec.arts.int-fiction: The newsgroup about writing (authoring) Interactive Fiction:
   rec.arts.int-fiction
   http://groups.google.de/groups?hl=en&lr=&ie=UTF-8&group=rec.arts.int-fiction

4. "So you want to write a text adventuring authoring system...", by Alan Conroy, in XYZZY magazine #14:
   http://www.xyzzynews.com/xyzzy.14i.html
   http://www.ifarchive.org/if-archive/magazines/XYZZYnews/XYZZY14.PDF

5. Glulx: The Glulx virtual machine (standard) by Andrew Plotkin:
   http://www.eblong.com/zarf/glulx/
   http://www.ifarchive.org/indexes/if-archiveXprogrammingXglulx.html
   ftp://ftp.ifarchive.org/if-archive/programming/glulx/

6. Glk: Andrew Plotkin's Glk API, which provides a portable interface for text adventure systems:
   http://www.eblong.com/zarf/glk/
   http://www.ifarchive.org/indexes/if-archiveXprogrammingXglk.html
   ftp://ftp.ifarchive.org/if-archive/programming/glk/

7. The Cloak Of Darkness sample adventure:
   http://www.firthworks.com/roger/cloak/

8. XML, the Extensible Markup Language:
   http://www.w3.org/XML/

9. "Ladder-logic":
   http://en.wikipedia.org/wiki/Ladder_logic
   http://www.google.com/search?hl=en&q=ladder+logic&btnG=Google+Search

10. gcc: The GNU C/C++ compiler:
    http://www.gnu.org/software/gcc/
    http://www.delorie.com/djgpp/

11. The GNU Bison parser generator:
http://www.gnu.org/software/bison/

12. The Java(TM) Virtual Machine Specification:
http://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html

13. The Specification of the T3 Virtual Machine:
http://www.tads.org/t3spec/intro.htm

14. Z-CODE:
http://www.ifarchive.org/indexes/if-archiveXinfocomXinterpretersXspecification.html

ftp://ftp.ifarchive.org/if-archive/infocom/interpreters/specification

15. The Inform Designer's Manual, 4th edition, by Graham Nelson:
ftp://ftp.ifarchive.org/if-archive/infocom/compilers/inform6/manuals/designers_manual_4.pdf
http://www.ifarchive.org/if-archive/infocom/compilers/inform6/manuals/designers_manual_4.pdf

16. The TADS Manuals, by Michael J. Roberts and Eric Eve:
http://www.ifarchive.org/indexes/if-archiveXprogrammingXtads2Xmanuals.html
http://www.ifarchive.org/indexes/if-archiveXprogrammingXtads3Xmanuals.html

17. Curses, by Graham Nelson.
http://www.ifarchive.org/indexes/if-archiveXgamesXzcodeXcurses.z5
ftp://ftp.ifarchive.org/if-archive/games/zcode/curses.z5

18. Texinfo: The GNU Texinfo system for creating documentations:
http://texinfo.org/
http://www.gnu.org/software/texinfo/

19. IF-archive: The central repository for software related to Interactive Fiction:
ftp://ftp.if-archive.org/
http://www.ifarchive.org/

Index

This is some kind of general index, so don't wonder if you find persons, concepts, keywords and names all mixed up here.

The index is aimed to be complete regarding topics, not complete regarding references. Therefore, not every occurence of e.g. "TADS" in the text is referenced here, only those occurences that put TADS in the context of developing an authoring system (and e.g. not the mentioning of TADS in the preface).

What stuff I used

Created using GNU Make 3.80, Copyright (C) 2002 Free Software Foundation, Inc.; pdfTEX (Web2C 7.4.5) 3.14159-1.10b, kpathsea version 3.4.5, Copyright (C) 1997-2003 Han The Thanh; Kpathsea is copyright (C) 1997-2003 Free Software Foundation, Inc.; TEX (Web2C 7.4.5) 3.14159, kpathsea version 3.4.5, Copyright (C) 1997-2003 D.E. Knuth; Kpathsea is copyright (C) 1997-2003 Free Software Foundation, Inc.; dvips(k) 5.92b, kpathsea version 3.4.5, Copyright (C) 2001 Radical Eye Software.; makeinfo (GNU texinfo) 4.7, Copyright (C) 2004 Free Software Foundation, Inc.

Created Sun Nov 27 21:13:41 CET 2005 using a Mobile Intel® Pentium® 4 running Ubuntu GNU/Linux® 5.10 (kernel 2.6.12-10-686).