

# **PyJuggler**

## **Getting Started Guide**

**Patrick Hartling**

---

# PyJuggler: Getting Started Guide

by Patrick Hartling

Copyright © 2003–2005 Patrick Hartling

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 the Invariant Sections being Appendix A, *GNU Free Documentation License*, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in Appendix A, *GNU Free Documentation License*.

---

---

---

---

# Table of Contents

What is PyJuggler? .....	vi
1. Getting the Software .....	1
Additional Software .....	1
Downloading .....	1
Extracting .....	2
2. Configuring the Environment .....	3
3. Running Example Applications .....	4
simpleGL .....	4
contextApp .....	4
PyAppLoader .....	5
PyExtApp .....	5
4. Writing Code .....	6
Context-Specific Data ( <code>vrj::GlContextData&lt;T&gt;</code> ) .....	6
Clustered Application Data ( <code>cluster::UserData&lt;T&gt;</code> ) .....	8
A. GNU Free Documentation License .....	10
PREAMBLE .....	10
APPLICABILITY AND DEFINITIONS .....	10
VERBATIM COPYING .....	11
COPYING IN QUANTITY .....	11
MODIFICATIONS .....	12
COMBINING DOCUMENTS .....	13
COLLECTIONS OF DOCUMENTS .....	14
AGGREGATION WITH INDEPENDENT WORKS .....	14
TRANSLATION .....	14
TERMINATION .....	14
FUTURE REVISIONS OF THIS LICENSE .....	15
ADDENDUM: How to use this License for your documents .....	15

---

## List of Examples

4.1. C++ Use of <code>vpr::Interval</code> .....	6
4.2. Python Use of <code>PyJuggler.vpr.Interval</code> .....	6
4.3. Using <code>PyJuggler.vrj.GlContextData</code> .....	7
4.4. Sharing Application Data with <code>PyJuggler.cluster.UserData</code> .....	8

---

# What is PyJuggler?

PyJuggler is a collection of Python bindings for the VR Juggler C++ application development framework. With PyJuggler, VR Juggler application objects can be written in Python and loaded into the VR Juggler C++ microkernel. Python application objects can even be intermingled with C++ application objects to allow for multi-language switching between virtual worlds.

The Python bindings that make up PyJuggler are written as C++ modules that the Python interpreter can load at runtime just as it would any other Python module. There are several Python modules, each corresponding to one of the software modules that comprise the larger Juggler framework. For example, the module `PyJuggler.gadget` contains bindings for the Gadgeteer C++ library.

---

# Chapter 1. Getting the Software

Because PyJuggler follows the well-known layered architecture for software, it is built on top of other tools. These must be downloaded and installed prior to installing PyJuggler to ensure that dependencies are satisfied correctly.

## Additional Software

The following is a list of software that is required to utilize PyJuggler, regardless of the type of VR Juggler application that will be written. In other words, these software packages are *required* to use PyJuggler.

- Python [<http://www.python.org/>]: The dynamically typed object-oriented scripting language. The version of Python required depends on the requirements of Boost.Python (see the next bullet item), but as of this writing, Python 2.2 or newer is necessary.
- Boost.Python [<http://www.boost.org/libs/python/doc/>]: The high-level glue between C++ and Python code. PyJuggler requires Boost.Python v2 or newer. The version of Boost.Python used should come from the Boost same version against which VR Juggler was compiled. As of this writing, that is Boost 1.31.0.
- VR Juggler [<http://www.vrjuggler.org/>]: The cross-platform, cross-VR system virtual platform for which PyJuggler provides Python bindings. Without VR Juggler, PyJuggler has no purpose. Releases of PyJuggler come shortly after VR Juggler releases. Hence, the latest version of PyJuggler requires the latest version of VR Juggler (2.0 or newer).
- PyGMTL [<http://ggt.sourceforge.net/>]: Python bindings for the Generic Math Template Library (GMTL). These are required for proper acquisition of GMTL-based data from VR Juggler classes. For example, the transformation matrix for a `gadget.PositionInterface` object will be returned as a `gmtl.Matrix44f` object. PyGMTL is developed independently of VR Juggler and PyJuggler, but the version of PyGMTL used must be the same as the version of GMTL that comes with a VR Juggler release.

With those packages installed, one or more of the following will be needed in order to render graphics from the VR Juggler application object:

- PyOpenGL [<http://pyopengl.sourceforge.net/>]: Python bindings for the OpenGL [<http://www.opengl.org/>] graphics API. OpenGL-based VR Juggler applications written in Python use PyOpenGL to make calls into the natively compiled OpenGL libraries on the local system. We have tested with PyOpenGL 2.0.
- PyOSG [<http://sourceforge.net/projects/pyosg>]: Python bindings for the Open Scene Graph [<http://www.openscenegraph.org/>] (OSG). OSG-based VR Juggler applications written in Python use PyOSG to make calls to the natively compiled OSG libraries on the local system. If direct OpenGL calls must also be made, then PyOpenGL will be needed (see above). We have tested with PyOSG 0.4.5.

## Downloading

PyJuggler is distributed from the VR Juggler SourceForge.net project site [<http://www.sourceforge.net/projects/vrjuggler/>]. Because PyJuggler is still in the developmental stages, it

is recommended that users keep up to date with the latest version at all times. PyJuggler will always keep up to date with the VR Juggler API, and because it provides a one-to-one mapping of C++ to Python, the API will be identical except for slight differences based on Python syntax.

PyJuggler is an open source software library, and as such, its source code is available. Users are free to download the source, look at it, and modify it (keeping in mind the requirements of the license, of course). For the most part, the PyJuggler source code is pretty dull; the interesting aspect is what developers can do to write VR Juggler application objects in Python. As such, the source is provided primarily so that people can compile it for their local hardware and operating system. Some pre-compiled binary versions will exist, but the requirements of those binaries may not satisfy every single user's needs.

## Extracting

PyJuggler comes in a compressed archive format. For UNIX-based environments, we use the well-known **tar**(1) format for the archive and either **gzip**(1) (.gz file extension) or **bzip2**(1) (.bz2 file extension) for the compression. Because pre-compiled versions of PyJuggler are quite large, we normally use the BZIP2 format because it gives a better compression ratio than GZIP. To extract the contents of a BZIP2-compressed TAR file, use the following if GNU **tar**(1) is not available:

```
% bzip2 -cd pyjuggler.tar.bz2 | tar -xvf -
```

If GNU **tar**(1) is installed (it may be installed as **gtar** on some systems), use the following simpler command:

```
% tar -xjvf pyjuggler.tar.bz2
```

For Win32 environments, we use PKZIP compression (files have a .zip extension). The typical way to extract such an archive is to use the WinZip [<http://www.winzip.com/>] software. We will not show its use here since it offers a fairly straightforward GUI.

---

# Chapter 2. Configuring the Environment

PyJuggler needs all the environment variables required for execution of normal C++ VR Juggler applications. Additionally, the PyJuggler `lib` directory must be included in the shared library (DLL) search path so that the library `libpyjutil.so` (or `pyjutil.dll` or `libpyjutil.dylib`) can be found at module load time. This is normally accomplished by modifying the environment variable `LD_LIBRARY_PATH`, `PATH`, or `DYLD_LIBRARY_PATH`, depending on the operating system. If PyJuggler is installed so that its `lib` directory is already in the shared library search path, then the library search path does not have to be changed.

With the addition of Python into the mix, it may be necessary to set the `PYTHONPATH` environment variable. This environment variable tells Python where to look for modules outside of its default path. In other words, it *augments* the Python module search path to use the directories listed therein.

The need for `PYTHONPATH` depends on where PyJuggler is installed. If PyJuggler is installed where Python modules are installed by default, then Python will not need any help to find the PyJuggler modules. However, a pre-compiled version of PyJuggler can be installed anywhere (which is good for people who do not have administrative rights on their local system and thus cannot perform system-wide installations). For example, on a UNIX-based system, PyJuggler could be unpacked into `$HOME/pyjuggler`. If PyJuggler was compiled against Python 2.2, `PYTHONPATH` would be set as follows (assuming the use of either `cs`h or `tc`sh as the user's shell):

```
% setenv PYTHONPATH $HOME/pyjuggler/lib/python2.2/site-packages
```

For more information about Python, Python modules, and environment variables utilized by the Python interpreter, refer to the on-line Python documentation [<http://www.python.org/doc/>].

## Tip

PyOpenGL and PyGMTL may have to be handled similarly to PyJuggler as far as `PYTHONPATH` is concerned. It may be convenient to install all three in the same place so that only one directory must be specified in `PYTHONPATH`. Of course, one or both of PyOpenGL or PyG-MTL may be installed in the default Python module directory, so this may not be a concern at all.

On Mac OS X, problems have been encountered with the omniORB [<http://omniorb.sourceforge.net/>]-based plug-ins that are loaded by the JCCL Config Manager and by the VR Juggler Performance Mediator. Loading of these plug-ins can be disabled by setting the environment variables `NO_RTRC_PLUGIN` and `NO_PERF_PLUGIN` respectively. They may be set to any value. Setting these environment variables will prevent a crash inside the omniORB libraries due to static initialization problems.

## Note

According to users of omniORBpy, setting `DYLD_BIND_AT_LAUNCH` should be sufficient to fix the static initialization problems within the omniORB C++ libraries when they are loaded into the Python interpreter application space on Mac OS X. Thus far, this has not proven to fix the problem when running a PyJuggler Python application, but there may be something else going wrong. For now, setting the two environment variables described above gets around the omniORB problems within the scope of PyJuggler.

---

# Chapter 3. Running Example Applications

Executing the example PyJuggler applications works very similarly to C++ VR Juggler applications. This is done on purpose to make it easier for people with C++ VR Juggler experience to get started with PyJuggler. Of course, how any given application is written is entirely up to the author; these are simply examples.

## simpleGL

In the PyJuggler installation directory, a very simple “pure Python” application can be found under `share/pyjuggler/examples/python/simpleGL`. The directory contains a single file `simpleGL.py`. To execute this application, the following will work in a UNIX-based environment:

```
% ./simpleGL.py standalone.jconf
```

For a DOS/Win32 environment, the command is similar:

```
> python simpleGL.py standalone.jconf
```

### Note

For VR Juggler 2.0 Alpha 4 and beyond, the full path to the configuration files does not have to be specified, so it is omitted in the above examples. The file `standalone.jconf` can be found in the directory `$VJ_BASE_DIR/share/vrjuggler/data/configFiles`.

This opens a single graphics window showing the default VR Juggler simulator environment and a gray box. Clicking button 1 on the mouse causes the box to be grabbed so that it moves wherever the simulated user's hand moves. Releasing button 1 causes the box to return to its original location. To exit the application, press the ESC key.

## contextApp

This pure-Python application demonstrates the use of context-specific data through PyJuggler. The application can be found under the directory `share/pyjuggler/examples/python/contextApp`. It contains a single file `contextApp.py`. To execute this application, the following will work in a UNIX-based environment:

```
% ./contextApp.py standalone.jconf
```

For a DOS/Win32 environment, the command is similar:

```
> python contextApp.py standalone.jconf
```

This opens a single graphics window showing the default VR Juggler simulator environment and a gray box attached to the wand. To exit the application, press the ESC key. To see the multi-context support in action, run the application as follows:

```
% ./contextApp.py sim.base.jconf sim.wand.mixin.jconf sim.c6displays.mixin.jconf
```

## Caution

The above will open *nine* OpenGL windows, which can be very taxing on the graphics hardware. In some cases, it may even crash the computer. The multi-context behavior can be demonstrated through less abusive measures by using `sim.c6viewports.mixin.jconf` instead of `sim.c6displays.mixin.jconf`.

## PyAppLoader

PyAppLoader is a C++ application that demonstrates how to load Python modules containing VR Juggler application objects into the VR Juggler microkernel for execution. It can be found in `share/pyjuggler/examples/cxx/PyAppLoader` under the base PyJuggler installation directory.

## PyExtApp

PyExtApp is another C++ that shows how Python code can be called from within a C++ VR Juggler application object. The Python function called is quite trivial because the purpose of the application is to show what of the Python/C API [<http://www.python.org/doc/current/api/api.html>] must be utilized in order to make things work. The source code can be found in `share/pyjuggler/examples/cxx/PyExtApp` under the base PyJuggler installation directory.

---

# Chapter 4. Writing Code

In creating PyJuggler and PyGMTL, a conscious effort has been made to keep the Python class interfaces the same, or nearly the same, as the backend C++ class. Reading the VR Juggler *Programmer's Guide* should be more than sufficient for programmers to learn how to use PyJuggler. The most notable syntactic difference from C++ to Python will be the lack of the `::` and `->` operators. This difference is illustrated in the following two code blocks. While this example is overly simple, it illustrates the point that the Python interface to a given C++ classes uses the same basic interface.

## Example 4.1. C++ Use of `vpr::Interval`

```
#include <iostream>
#include <vpr/Util/Interval.h>

void func()
{
    vpr::Interval i = vpr::Interval::now();
    std::cout << i.sec() << " seconds" << std::endl;
}
```

## Example 4.2. Python Use of `PyJuggler.vpr.Interval`

```
import PyJuggler.vpr as vpr

def func():
    i = vpr.Interval.now()
    print i.sec(), "seconds"
```

With that in mind, there are some important parts of the Juggler C++ interface that do not map well to Python. For example, the context-specific data handler `vrj::GlContextData<T>` relies on language-level support for templates (more accurately, generic types) and the higher level concept of a smart pointer. Python does not have generic types; indeed, it is an untyped language. In some cases, Python language features have been exploited to make the interfaces more Python-esque so that experienced Python programmers can take advantage of the Python features they know and love. The remainder of this chapter will explain the specific instances where the PyJuggler classes differ from the C++ Juggler classes.

## Context-Specific Data (`vrj::GlContextData<T>`)

Context-specific data is an important aspect of utilizing the VR Juggler OpenGL Draw Manager. As such, it must be available for authors of VR Juggler application objects who write their application objects in Python. We will not get into the details of context-specific data here and instead refer readers to the relevant parts of the VR Juggler *Programmer's Guide*.

In a C++ application object, we would define a type instantiation of the generic type `vrj::GlContextData<T>`. As stated above, Python does not have generic types, so we cannot define our own instantiations of `vrj::GlContextData<T>` in a Python application object. We can,

however, take advantage of the dynamic nature of Python to mimic the semantics of a C++ instantiation of `vrj::GlContextData<T>`. The result is an easy-to-use Python class that allows dynamic context-specific type construction, conveniently named `PyJuggler.vrj.GlContextData`.

For each piece of context-specific data that is required by the application object, an instance of `PyJuggler.vrj.GlContextData` (referred to henceforth as `vrj.GlContextData`) must exist as a data member in the application object. The `vrj.GlContextData` instances can then have any number of attributes added to them dynamically. This usage is shown in Example 4.3, “Using `PyJuggler.vrj.GlContextData`”.

### Example 4.3. Using `PyJuggler.vrj.GlContextData`

```

1 from OpenGL.GL import *
  import PyJuggler.vrj as vrj

  class MyApp(vrj.GlApp):
5     def __init__(self):
        vrj.GlApp.__init__(self)

        # Allocate the context-specific objects.
        self.cube_display_list = vrj.GlContextData()
10        self.texture_obj      = vrj.GlContextData()

        def contextInit(self):
            # Add an attribute named 'id' that stores a display list ID
            # for this context.
15            self.cube_display_list.id = glGenLists(1)
            glGenLists(self.cube_display_list.id, GL_COMPILE)
            # Define the OpenGL commands for the display list...
            glEndList()

20            # Add an attribute named 'id' that stores a texture object ID
            # for this context.
            self.texture_obj.id = glGenTextures(1)
            glBindTexture(GL_TEXTURE_2D, self.texture_obj.id)

25        def draw(self):
            # Requests the display list ID for this context.
            glCallList(self.cube_display_list.id)

30            # Set up texture coordinates...
            # Render the texture using the texture object ID for this context.
            glBindTexture(GL_TEXTURE_2D, self.texture_obj.id)

```

Using instances of `vrj.GlContextData` has the same basic restrictions as in C++. Reads and writes to the attributes of the `vrj.GlContextData` instances can only occur when an OpenGL context is active. That is, context-specific data attribute access can occur only in the application object overrides of `contextInit()`, `contextPreDraw()`, `draw()`, `contextPostDraw()`, and `contextClose()`. This usage is shown in more detail in the example Python application `share/pyjuggler/examples/contextApp/contextApp.py`.

The use of context-specific data from a Python application object is implemented as a regular C++ class exposed to Python via Boost.Python. The special aspect is that this class makes use of a context-specific dictionary. We take advantage of the dynamic nature of Python to graft attributes on to objects at run time. Accesses (sets and gets) are intercepted and re-routed to the context-specific dictionary behind the scenes (in C++, no less).

This implementation preserves the features and the semantic behavior of using `vrj::GlContextData<T>` directly in C++, and it does it in a way that is largely transparent to Python programmers. In a sense, it makes for a good balance between the features of C++ and Python simultaneously. The big down side is the restricted attribute access that must be correlated with an active OpenGL context. Of course, this same limitation exists when using `vrj::GlContextData<T>` directly in C++.

## Clustered Application Data (`cluster::UserData<T>`)

The cluster feature of VR Juggler includes support for sharing user-defined data types across the nodes of the cluster. This is achieved through the use of the generic type `cluster::UserData<T>` where `T` must be a subclass of `vpr::SerializableObject`. Programmers must implement overrides of the pure virtual methods `vpr::SerializableObject::readObject()` and `vpr::SerializableObject::writeObject()`, which serialize and de-serialize the object respectively. In Python terms, this corresponds directly to pickling and unpickling an object. This correlation provides the foundation for the access to `cluster::UserData<T>` through `PyJuggler`<sup>1</sup>.

Clustered application data is implemented using the class `PyJuggler.cluster.UserData`, henceforth referred to as `cluster.UserData`. This class takes advantage of Python language features to provide a more powerful, flexible data sharing mechanism that is easier to use than its C++ counterpart. An instance of `cluster.UserData` can share any Python data structure that can be pickled. Since most Python types can be pickled automatically, there is no need for application programmers to write any code for serializing and de-serializing their data structures. Furthermore, the dynamic nature of Python types allows the shared data structures to change at run time. Multiple instances of `cluster.UserData` can be used to share multiple data structures, or the multiple data structures can be aggregated into a single Python object that is shared via a single `cluster.UserData` instance.

The use of `cluster.UserData` is shown in Example 4.4, “Sharing Application Data with `PyJuggler.cluster.UserData`”. Just as with `cluster::UserData<T>`, the clustered data must be initialized in the application object's `init()` method using a GUID. After the `cluster.UserData` object is initialized, the Python object to be shared must be registered through the method `cluster.UserData.setPickleObject()`. All nodes in the cluster must perform these actions.

### Example 4.4. Sharing Application Data with `PyJuggler.cluster.UserData`

```

1 import gmtl
  from PyJuggler import *

  class ClusterApp(vrj.GlApp):
5     def __init__(self):
        self.user_data = cluster.UserData()
        self.nav_matrix = gmtl.Matrix44f()

        def init(self):
10         self.user_data.init(vpr.GUID("2bbad1af-27c4-11d9-bd0d-00045a86e9cd"))
            self.user_data.setPickleObject(self.nav_matrix)

        def preFrame(self):
            if self.user_data.isLocal():

```

<sup>1</sup>It is assumed that the reader is already familiar with the use of `cluster::UserData<T>` in C++. Those readers who are not familiar with this class should refer to the appropriate chapter of the VR Juggler *Programmer's Guide* for details.

```

15         nav_delta = gmtl.Matrix44f()
           # Calculate navigation change...
           gmtl.postMult(self.nav_matrix, nav_delta)

           def draw(self):
20             glPushMatrix()
               glMultMatrixf(self.nav_matrix.getData())
               # Render the scene...
               glPopMatrix()

```

Once the shared data is registered with the `cluster.UserData` instance, only the application data host node should be allowed to write to the shared data. If any other node writes to the shared data, the changes will be lost the next time the shared data is unpickled. The application data host node can be determined using the method `cluster.UserData.isLocal()`, just as in the C++ counterpart. Any node can read from the shared data. Note that, unlike the of `cluster::UserData<T>` in C++, the shared data can be accessed directly. Indeed, it *must* be accessed directly as there is no smart pointer semantics associated with `cluster.UserData`. While this makes shared application data use easier in Python than in C++, it is also more efficient because there is no extra level of indirection.

As noted above, most shared data structures will not require any special code to allow pickling and unpickling. Refer to the Python object pickling documentation [<http://docs.python.org/lib/module-pickle.html>]. More specifically, refer to the documentation on what can be pickled and unpickled [<http://docs.python.org/lib/node65.html>].

## Note

As of this writing, the only PyJuggler classes that can be pickled are `PyJuggler.vpr.Interval` and `PyJuggler.vpr.GUIID`. If any other PyJuggler class instances are stored in an object shared across the cluster via `cluster.UserData`, pickling will fail. All data types in PyGMTL can be pickled.

The backend C++ implementation of `cluster.UserData` utilizes `cluster::UserData<T>` and a subclass of `vpr::SerializableObject` called `pyj::PickleObject`. `pyj::PickleObject` makes use of the `pickle` module from C++ through `Boost.Python`. Every instance of `pyj::PickleObject` holds a local `boost::python::object` instance that is the Python object registered through `cluster.UserData.setPickleObject()`. In `pyj::PickleObject::readObject()`, the `boost::python::object` instance is pickled using `pickle.dumps()`. In `pyj::PickleObject::writeObject()`, the received pickled object is unpickled using `pickle.loads()`. The full dictionary of the freshly unpickled `boost::python::object` instance is copied into the local `boost::python::object` instance. This allows dynamic structural changes to be propagated from the application data host node to the rest of the cluster nodes.

The current backend C++ implementation for `cluster.UserData` makes use of the basic, slow Python pickling protocol. This protocol pickles objects into strings. It is not known at the time of this writing if the binary pickling support would allow for heterogeneous communication that is supported by VR Juggler's clustering capabilities. In other words, it is not known if the binary pickling protocol would allow for cross-version and/or cross-platform data transfer.

---

# Appendix A. GNU Free Documentation License

Version 1.2, November 2002

## FSF Copyright note

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## 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 noncommercially. Secondly, 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.

## 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 Sec-

tions 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.

## 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.

## 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.

## 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:

### GNU FDL Modification Conditions

- 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.

## 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".

## 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.

## 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.

## 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.

## 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.

## 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.

## 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:

### **Sample Invariant Sections list**

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:

### **Sample Invariant Sections list**

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.