Getting Started with Swarm

http://www.santafe.edu/projects/swarm/

Swarm Development Group

Published 10 February 1999
Release 1.4.1

Abstract 1: Overview

Learning to program with Swarm will require reading example applications, studying the technical reference material and sometimes even getting down to the level of reviewing the header files in libraries. It does currently require a good hands-on knowledge of object-oriented programming and software development processes in general.

Swarm is not yet a 'shrink-wrapped' simulation toolkit. There are many of those kind of products on the market. However, with these packages, the ease of use comes at a price - you are locked into the that vendor's particular modelling paradigm. Swarm was intended to embrace many different types of modelling - consequently, it can be more difficult for a novice user - but more powerful in the long-run.

Revision History

1999-01-26 installmeta.sgml *alex*
(booktitlelogo): Scale GRAPHIC by 98%, so all recto-mode elements fit on one page in print backend.
1999-01-25 installmeta.sgml *alex*
Move bibliodata entity outside of BOOKBIBLIO - so legalnotice links work.
1999-01-12 installmeta.sgml *alex*
Include the newly-defined {doc,swarm}-legalnotice entities from global.ent. Move CORPAUTHOR inside BOOKBIBLIO.
1998-10-31 install01.sgml *alex*
Changed URL to Paul Johnson's Swarm FAQ to new location.
1998-10-13 installbook.tex.in *mgd*
Use top_dossrcdir instead of top_srcdir.
1998-10-08 installbook.tex.in *mgd*

Include tex/macros.tex.
1998-10-05 installmeta.sgml *mgd*
Make more conservative. Remove statements of intent.
1998-10-05 install01.sgml *mgd*
Update library version facts, compatibility, etc. Add URLs for both versions and general package information.
1998-07-16 install02.sgml *alex*
Fixed ID of ARTICLE to ‘INSTALLBOOK’ rather than ‘INSTALL’.
1998-07-16 install01.sgml *alex*
(SIMPLESECT): Updated ULINK http and ftp locations for download of Tcl/Tk to ‘scriptics.com’. ‘smli.com’ is now out-of-date. Also changed link to BLT to ftp location, rather than the webpage. (ARTICLE): Added missing ID.
1998-06-24 install01.sgml *alex*
Updated installation instruction to reflect new ‘configure’ process.
1998-06-16 Makefile.am *mgd*
(GENERATED_SGML): Use $(NAME)revhistory.sgml instead of installrevhistory.sgml. (EXTRA_DIST): Use installbook.ent instead of install.ent.
1998-06-16 installmeta.sgml *mgd*
Use installbookrevhistory.sgml instead of installrevhistory.sgml. Scale graphic to 100%.
1998-06-15 install01.sgml *alex*
Updated ‘Supported Systems’ to make mention of Windows NT. Updated ‘Prerequisite Libraries’ to mention libraries required as of Swarm 1.1/1.2.
1998-06-14 install.ent *alex*
Removed install00.sgml from list of ENTITIES.
1998-06-14 installbook.sgml *alex*
Added new JPEG notation class and notation.
1998-06-14 Makefile.am *alex*
(SGML): Removed install00.sgml.
1998-06-14 installcont.sgml *alex*
Removed include of install00.sgml.
1998-06-14 install02.sgml *alex*
Removed links to ’reference book’ elements - made all linkends be FORMALPARAs.
1998-06-14 install01.sgml *alex*
Commented-out LINK to reference book ’debug’ section. Moved original PREFACE material in install00.sgml into new SIDEBAR.
1998-06-14 install00.sgml *alex*
Removed.
1998-06-14 Makefile.am *mgd*
Include $(top_srcdir)/Makefile.common. (EXTRA_DIST): New variable.
1998-06-11 installbook.sgml *mgd*
Include versions and figures entities.
1998-06-11 installmeta.sgml *mgd*

Scale graphic to 75%.
1998-06-11 install00.sgml, install01.sgml, install02.sgml, installmeta.sgml *mgd*
Update all IDs to SWARM.module.SGML.type.
1998-06-10 install.ent *alex*
Made all SYSTEM identifiers PUBLIC (listed in catalog now).
1998-06-10 Makefile.am *alex*
(GENERATED_SGML): Added for revhistory. (noinst_DATA): Added. (SGML, SGML_FILES): Separated out generated/non-gen files.
1998-06-10 install.ent *alex*
Created installrevhistory.sgml entity.
1998-06-10 installmeta.sgml *alex*
Replaced hardcoded revhistory with the new entity installrevhistory.sgml.
1998-06-08 installmeta.sgml *alex*
Changed corpauthor to be inserted from Global public identifiers.
1998-06-07 installbook.sgml *mgd*
Use public identifiers.
1998-05-22 installbook.tex.in *mgd*
New file.
1998-05-22 Makefile.am *mgd*
New file.
1998-05-22 installbook.sgml *mgd*
Make ID of book be "installbook".
1998-05-22 Makefile *mgd*
Remove.
1998-05-21 *mgd*
Begin revision log.

Table of Contents

Unless you are using a Windows or a Unix system with pre-built package manager support (Debian GNU/Linux 2.0, Redhat 5.1, Solaris 2.5), installing Swarm will take some time: various external libraries need to be acquired, compiled, and installed, and then Swarm itself needs to be compiled. Please report any problems during installation back so that we can try to fix them in the future.

If you are using a Unix system with binary package manager support, please read the manual appropriate to the manager. Respectively, these are 'dpkg', 'rpm', and 'pkgadd'.

Swarm uses quite a few libraries and is intended to run under all major flavors of Unix. This presents the unpleasant but largely unavoidable side-effect of making Swarm hard to install. Ideally, you should get the sys-admin for your site to do the job. In any event, the new online *Swarm FAQ* where we've tried to compile some of the common obstacles to installation, may be useful.

- **Online FAQ *(http://lark.cc.ukans.edu/~pauljohn/SwarmFaq/SwarmOnlineFaq.html)* .** Paul Johnson's active Swarm FAQ. *Share your wisdom!*

Supported Systems

- **Unix.** Our intention is that Swarm will run on any modern Unix system. Ideally, Swarm itself should be 100% portable. Swarm is currently known to be running on SunOS 4.1.3, Solaris, libc5- and glibc-based GNU/Linux systems, HPUX 9 and 11, IRIX 5.3 and 6.4, and MachTen/68k. Significant potential obstacles to getting Swarm running on a new Unix platform are 1) poor support for GCC Objective C and 2) portability problems with libffi or ffcall.
- **Windows NT.** We have also ported Swarm to work on Windows NT. The following discussion largely pertains to installing Swarm under Linux/Unix systems. In the case of Windows NT, the installation of the 'Prerequisite Programs' is replaced by installing the Cygnus Win32 environment, and a Swarm "Install Shield" has been created to install Swarm on Windows NT (also replacing the job of installing the 'Prerequisite Libraries'). Please see the Swarm release page *(http://www.santafe.edu/projects/swarm/release.html)* , for more details.

See the Platform News *(http://www.santafe.edu/projects/swarm/platforms.html)* web page for the most up-to-date information.

Prerequisite Programs

We assume you already have Unix and X Windows on your system: the rest of this software is freely available. Expect to spend some time compiling and installing these packages. URLs on this page are to the home distribution site: as a convenience, the Swarm ftp site ftp://ftp.santafe.edu/pub/swarm/

(ftp://ftp.santafe.edu/pub/swarm/needed-software/) contains copies of the necessary packages in needed-software/ *(ftp://ftp.santafe.edu/pub/swarm/needed-software/)* .

- **gcc 2.7.x / gcc 2.8.x / egcs 1.x.** gcc is the FSF's C, C++, and Objective C compiler. You need gcc 2.7.2 or greater installed on your system, including the Objective C support, libobjc.a. The latest version is available from ftp://ftp.gnu.org/pub/gnu/ *(ftp://ftp.gnu.org/pub/gnu)* . Note: Swarm will not work with versions of gcc before 2.7.2.
- **GNU make.** Make is used to automate building programs: every OS ships their own version of make with some random set of features. The Swarm makefiles currently use features that are not supported by all makes: GNU make is the only one guaranteed to work. Linux systems probably use GNU make already. The latest version of GNU make is available from GNU *(ftp://ftp.gnu.org/pub/gnu/)* .
- **gdb.** gdb is the GNU debugger: Swarm doesn't require this, but you will probably want gdb on your system to debug programs. gdb is available on GNU *(ftp://ftp.gnu.org/pub/gnu/)* .

Prerequisite Libraries

Swarm uses a variety of freely available software libraries. All of these libraries need to be configured and installed independently of Swarm. When you configure Swarm itself, take a look at the output of "configure –help" to see what "–with-*dir" options are available for locating the installations of these libraries.

- **XPM Library *(http://www.inria.fr/koala/lehors/xpm.html)* .** The XPM library adds pixmap (coloured bitmap) support to X11. XPM is a common X library, many systems already have it installed. A reasonably modern version is needed: we've used version 3.4f. If you get complaints about "Object" being multiply defined, your version of libXpm is too old. The library is available from ftp://ftp.x.org/contrib/libraries/xpm-3.4h.tar.gz *(ftp://ftp.x.org/contrib/libraries/xpm-3.4h.tar.gz)* .
- **Tcl/Tk *(http://www.scriptics.com)* .** Swarm works with Tcl/Tk versions 7.4/4.0, 7.5/4.1, 7.6/4.2, and 8.0p2 *(ftp://ftp.scriptics.com/pub/tcl/tcl8_0/tcl8.0p2.tar.gz)* /8.0p2 *(ftp://ftp.scriptics.com/pub/tcl/tcl8_0/tk8.0p2.tar.gz)* . It may work with later versions, but it hasn't been tested.
- **BLT *(http://www.tcltk.com/blt/index.html)* .** BLT is an add-on to Tk that provides more widgets. BLT 2.4f *(http://ftp.tcltk.com/pub/blt/BLT2.4f.tar.gz)* is the current version. Another version, BLT 8.0-unoff (SFI) *(ftp://ftp.santafe.edu/pub/swarm/needed-software/blt8.0-unoff-swarm-0.tar.gz)* is needed for use with Windows.
- **ffcall *(http://clisp.cons.org/~haible/packages-ffcall.html)* / libffi *(http://www.cygnus.com/~green/libffi.html)* .** Either ffcall or libffi can be used in Swarm to provide the underlying support for message probing. Both libraries provide a portable, high level programming interface to various platform calling conventions. This allows a programmer to call any function specified by a call interface description at run time. ffi stands for Foreign Function Interface.

A foreign function interface is the popular name for the interface that allows code written in one language to call code written in another language.

The latest version of ffcall is 1.3.1 *(ftp://ftp.santafe.edu/pub/gnu/ffcall-1.3.1.tar.gz)* The latest version of libffi is: 1.20 *(ftp://ftp.cygnus.com:/pub/green/libffi-1.20.tar.gz)*

- **libpng *(http://www.cdrom.com/pub/png)* .** Swarm requires support for pixmap images - ‘png’ provides that support.

The current version is 1.0.2 *(ftp://swrinde.nde.swri.edu/pub/png/src/libpng-1.0.2.tar.gz)* **zlib *(http://www.cdrom.com/pub/infozip/zlib)* .** png requires data compression which zlib, a general data compression library, provides. The current version is 1.1.3 *(ftp://ftp.cdrom.com/pub/infozip/zlib/zlib.tar.gz)* .

Configuring and Installing Swarm

Once you have all of the above software installed, the next step is to configure Swarm. First unpack the Swarm .tar.gz file into a convenient place `(via gzip -dc swarm-xx.yy.zz.tar.gz | tar xf -)`.

As of the Swarm 1.2 release, the compilation and installation of Swarm is handled by a GNU **configure** script. This removes a large burden from the user, as **configure** has some intelligence which automatically sets many common options.

You first need to read the file `INSTALL` with the release of Swarm, in the top-level source directory, to determine the appropriate and recent options to give to the **configure** script (this is analogous to what you needed to do when editing the *INCDIR macros in the obsolete `Makefile.conf` in earlier releases).

Compiling Swarm Libraries – (not required for binary distributions)

You’ve done the hard part, now type **make**, as in earlier releases from the top of the source directory, and watch the fun! If running **make** in the root Swarm directory does nothing, you probably aren’t running GNU Make.

There is now an explicit "installation" step to install Swarm – **make install** so that binaries and libraries can be installed cleanly to a specified location. On many Unix systems this likely to somewhere like `/usr/local/`.

The Makefiles included by user-created libraries, and user-created applications are `Makefile.lib`, and `Makefile.appl`, respectively. These distributed with earlier versions are now generated by **configure** . If you run into problems, the right thing to do is to re-run **configure**, and re-install Swarm, tinkering with these generated Makefiles directly is not recommended.

Compiling and Running Swarm Applications

Swarm applications are distributed separately: you will need to download and unpack applications independently. After the application is unpacked, you will have to change the `SWARMHOME` variable in the Makefile. It is setup, by default to point to "../swarm", as if the Swarm libraries were in a sibling directory called "swarm". Just edit that string to point to where you installed the Swarm libraries. No other configuration should be necessary. For example in the Makefile for your Swarm application, change:

```
SWARMHOME=../swarm
```

to the following format:
`SWARMHOME=<path to where Swarm is installed>`

For example, a typical place a system admin might install Swarm is:
`SWARMHOME=/usr/local/swarm-<version_number>`

All you need to do now, to compile an application is type **make**.

Once the application is compiled, just run it out of its own directory. A control panel and a couple of parameter windows should pop up; press the "Go" button to watch it run. Congratulations!

Post-compile cleanup

After you've tried out the Swarm libraries for awhile you may want to clean up the intermediate .o files, and other generated files, in the original source directory. Once you have done the **make install** you can remove the entire source directory, or if you might want to re-install later with different option, just type **make clean** to clean-up all files generated by the original make. See your the file `INSTALL` in the Swarm distribution, for more details.

Just remember to set `$(SWARMHOME)` to the right directory in your application's makefile. If you have installed Swarm, but for some reason, need to recompile the library and want to start from a completely clean source directory, you can run **make uninstall** and then **make clean**- see `INSTALL` file again here.

For debugging purposes, you should pass the `-g` option to **configure**, to whit: **CFLAGS=-g ./configure <the rest of your options>** and leave the sources in the build or source tree. However, your applications will run faster if you compile with optimization, **CFLAGS=-O3 -g**. This will make debugging inside the Swarm libraries strange, but gives noticeable performance improvement.

Libraries

The main part of Swarm is a set of libraries, one per source directory. Briefly, the directory organization is as follows:

lib, include, bin. Installed libraries, include files, and helper binaries.

src. Swarm library source files.

- **defobj.** Support of Swarm style Objective C programming
- **collections.** Collection library – various objects that "collect" other objects
- **activity.** Schedule execution mechanism – the "process"
- **objectbase.** Base class for simulated objects – the "agents"
- **space.** Spatial environment library (currenly mostly 2D spaces are supported).
- **random.** Random number library.
- **simtools.** Collected tools (non-GUI) useful for developing Swarm simulations
- **simtoolsgui.** Further tools useful for interactive "GUI-oriented" simulations
- **gui.** Graphical Interface library
- **analysis.** Objects that help with data processing

Sample applications

In addition, there is a set of sample Swarm applications that are distributed separately. Applications are the best current roadmap for the Swarm code: much of what is possible with Swarm is demonstrated in the sample applications. The most important applications are:

- **template.** Template simulation. The code itself here is trivial, but provides a nice base for new Swarm programmers to start from.
- **tutorial.** A step-by-step tutorial to using Swarm. The tutorial starts with a simple implementation of a cellular automaton in ordinary "C" and proceeds up to a full-blown GUI-oriented Swarm application in Swarm Objective C.
- **heatbugs.** Heatbugs, our canonical, simple complex system. The code here is thoroughly commented for use as a guide to swarm programming.
- **mousetrap.** Mousetrap is a discrete event simulation of a room full of mousetraps loaded with ping-pong balls. The triggering of one of the traps sets off a chain reaction supposedly similar to fission. It is also thoroughly commented.

- **market.** Market is an Objective-C wrapped piece of legacy code originally written in C. It is a useful example of how to convert legacy code into Swarm; but, it is not very instructive on Swarm programming in general.