MathEngine Karma™ Installation Guide

MathEngine Karma Collision. Developer Guide.

Overview

MathEngine Karma includes the Karma Dynamics and Collision software packages. This software - that includes libraries, partial library source code, headers, documentation, demo executables, and example source code - is available for PlayStation2, Windows, and Linux. A beta version is available for Xbox.

The evaluation software can be downloaded in the following formats:

File	Description
`metoolkit-1.1.0_ps2_s_static.tar.gz`	A tar archive for `Playstation2.` Single precision
`metoolkit-1.1.0_win32_s_msvcrt.zip`	A zip file for Win32. Single precision. Linked against MSVCRT
`metoolkit-1.1.0_win32_s_libcmt.zip`	A zip file for Win32. Single precision. Linked against LIBCMT
`metoolkit-1.1.0_linux_s_static.tar.gz`	A single precision tar archive for Linux

Double precision builds, win32 builds linked against LIBC, and Xbox builds are also available.

Note that, depending on which version you have downloaded, the version number 1.1.0 may be different.

To install Karma, unzip / untar the relevant archive in your chosen location.

Downloading Karma

You can download MathEngine Karma directly from the MathEngine Developers Website when you have registered at http://www.mathengine.com/ and activated your account.

Prerequisites

Target Platforms

Supported runtime platforms are PlayStation2, Win2000, Win98, WinNT, Red Hat Linux 6.x, and Xbox.

Development Platforms

Supported development platforms are:

- x86
  - Win32 (Cygwin, MS Visual C++ 6.0)
  - Linux (Red Hat 6.x)
- PS2
  - Linux (Sony ToolChain 2.14)
  - CodeWarrior
  - SN Systems (using make or Developer Studio)

The supported Win32 operating systems are WindowsNT, Windows98 and Windows2000. A Makefile mechanism (with `autoconf`) is provided for Linux and Windows. Visual C++ 6.0 workspace and project files are provided for Developer Studio environments (Win32 and SN Systems).

Installation Directory Structure

After unpacking, the directory structure on your disk will look like this:

```
\---metoolkit                                   Top-level of distribution
    |
    |   index.html                              Documentation link
    |   LICENSE.TXT                             Read this before installing
    |   README.TXT                              Read this before installing
    |
    |
    +---3rdParty                                Third-party software
    |   |
    |   |   COPYING.TXT                         QHull copyright
    |   |   README_Qhull.TXT                    QHull acknowledgement
    |   |   README_RenderWare.TXT               RenderWare acknowledgement
    |   \---glut                                OpenGL Utility Toolkit API
    |
    +---bin                                     Executable files.
    |   +---examples                            Example programs
    |   |
    |   +---GreaseMonkey                        GreaseMonkey Demo: README,
    |   |                                         executable, scripts, etc
    |   +---tutorials                           Tutorials executables
    |   |
    |   +---Resources                           BMP,.dll,.dtd and .ME files
    |   |
    |   \---Tools
    |         SceneLoader.exe
    |
    +---doc                                     Documentation Set:
    |                                             index.htm start-up page
    |                                             Release Notes
    |                                             CHANGES_MCD.TXT
    |                                             CHANGES_MDT.TXT
    +---include                                   Header files
    |
    +---lib.chk                                 Check Libraries
    |
    +---lib.rel                                 Release Libraries
    |
    \---src                                     Source files
        |
        |   README.TXT
        |   makefile                            Top-level makefile
        |   MeToolkits.dsw                      Top-level workspace for
        |                                         Visual C++
        |
        +---components                          Component Toolkits
        |
        +---examples                            Example programs:
        |                                         .c, .dsp, makefile
        \---tutorials                           Tutorial programs:
                                                  .c, .dsp, makefile
```

Building the Examples

Using Visual C++

To build the source code and samples provided using Visual C++:

1. Open `metoolkit\src\MeTookits.dsw`
2. Click **Build -> Batch Build**.
3. In the Batch Build dialog box, make sure that all components are selected.
4. Click **Rebuild All** in the dialog box.

The target paths provided are distinct from those in which the shipped libraries reside, so compiling the source provided will not override your installation.

When creating a new Microsoft VC++ project, the following additional libraries may need to be added:

- `libcmt.lib` (/MT) or `msvcrt.lib` (/MD)
- `oldnames.lib`
- `kernel32.lib`
- `user32.lib`
- `netapi32.lib`
- `advapi32.lib`
- `gdi32.lib`
- `comdlg32.lib`
- `comctl32.lib`
- `wsock32.lib`

To do this, go to **Project** then **Settings**, select the **Link** tab, then select **Input** in the **Category** list. You can now add the following libraries to the **Object/libraries modules** text box.

Using make

All the source code of this release, including the examples and tutorials, can be built in any UNIX-like environment with a sh-compatible shell by using the standard GNU-style build procedure. For Win32 Cygwin is recommended (see below for instructions for obtaining Cygwin). The makefiles depend on a central set of definitions and rules contained within the directory `src/components/build/makerules`.

Prior to compilation, the core set of rules must be processed with a GNU autoconf configure script (generated using GNU autoconf), to generate an appropriate central makefile. The makefiles throughout the tree will accomplish this task as needed, so no overt act is required to create this file.

Note: GNU make v3.77 is required .

Before building, export the PLATFORM environment variable and set it to the target platform for which you want to build. The types currently recognized are ps2, win32, and linux. Run the following command to set the PLATFORM environment variable:

```
export PLATFORM=win32
```

Here is the sequence of commands to do a top-level build of all source code in the `src` directory, given that the PLATFORM variable has already been set:

```
cd src
make
```

Similarly you can build only the examples or tutorials or a single component by going to that subdirectory and executing the following commands (e.g. for tutorials):

```
cd src/tutorials
make release
```

The makefiles support the following build rules:

- `release`. This builds libraries and executables with no symbols and no error/debug output.
- `debug`. This builds debug-able libraries and executables. Both symbols and error/debug output are enabled.
- `check_release`. This builds a release version (no symbols) but with error/debug output enabled.
- `clean`. This removes all intermediate compilation files and executables, but leaves behind temporary files used by the build process itself, such as dependencies.
- `clobber`. Like `clean,` but more thorough, attempting to return the source tree to its original state.
- Resulting example and tutorial executables are put in `examples/bin.rel` and `tutorials/bin.rel` respectively for release versions. `Debug` versions will be generated into `examples/bin.dbg` and `tutorials/bin.dbg`. `Check_release` builds will generate files into `examples/bin.chk` and `tutorials/bin.chk`, respectively.

The component libraries will be compiled into the directories `components/lib.rel`, `components/lib.dbg`, and `components/lib.chk`.

Using make: Platform-dependent notes

Win32

You can download the full suite of GNU tools for Win32 from:

http://sourceware.Cygnus.com/cygwin/

This includes all the relevant tools. We recommend building from the bash shell prompt.

Microsoft Visual Studio (`cl.exe` and `link.exe`) is required to build from within the Cygnus CYGWIN environment. The PATH, INCLUDE and LIB variables must be set up to locate the MS tools (possibly using `[installed location]\DevStudio\VC98\bin\vcvars32.bat`).

Linux

MathEngine currently supports the RedHat version 6 or later Linux distributions. To build and run the examples, you need an OpenGL compatible implementation, for example any recent version (3 or later) of Mesa. A 3D-accelerated driver and X windows implementation, for example XFree86 4, is recommended. MathEngine may support other Linux distributions for paid developers only. Please contact sales@mathengine.com for additional information.

Sony PlayStation2: Building from Linux

The libraries supplied with Karma are built using the Sony toolchain together with standard tools that can be found in recent Linux distributions. The version of the toolchain used to build the libraries is specified in the README_ps2libs.txt in the metoolkits/src folder.

It is possible to build the supplied source and examples using any of the three supported toolchains: the Sony compiler, Codewarrior from Metrowerks, and SN Systems.

Sony PlayStation2: Building from Win32, using SN Systems

The Sony PlayStation2 toolchain, recompiled for Win32 use, is included with the SN System tools. If you are not using the supplied DSPs you will need to download and install the GNU tools for Win32 as indicated in *Win32* on page 8. Karma's build system uses the default Sony PlayStation2 GNU linker; this produces libraries that are entirely compatible with those produced by the ProDG linker.

To build using the SN compiler on Playstation 2.

Option 1: Use Make

If you have Cygwin installed, you should be able to build the example programs and library source using the supplied makefiles. For example, the makefile for the Ballman example can be found in

…..\metoolkit\src\examples\BallMan;

Using a bash shell type:

export PLATFORM=ps2

then

make debug

or

make release

The executable binary or library will be placed in the appropriate ../bin.dbg or ../bin.rel directory

Option 2: MS Developer Studio

If you have installed the SN Visual Studio Integration plug-in you will have to create the .DSP from the makefile.

In MSDev:

Create a new PS2 EE project in normal way locating it in the same directory as the source files

Add the source files to the project

(Project Settings)

Add an additional include path to the ME include directory

Add PS2 to the preprocessor definitions (also _MECHECK for debug build)

Add an additional library path to the ME lib.rel/ps2_single or lib.chk/ps2_single directory

Add a selection of MathEngine libs, such as the following:

libMst.a

libMeApp.a

libMcdPrimitives.a

libMcdFrame.a

libMdt.a

libMdtBcl.a

libMdtKea.a

libMeViewer2.a

libMeMessage.a

libMeGlobals.a

Note that these libs may change - you will need to check this. You may need to change the order to resolve the link dependencies.

When building the SN compiler will probably output a number of warnings. You will probably not need to worry about these.

If you have problems, you should check that you have entered the correct paths because the compiler may not make it obvious that there is a problem with this.

To build using the Codewarrior compiler on Playstation 2.

Setup for the released projects

To build the libraries use the project file in metoolkit\src\components\components.mcp

To build the examples use the project file in metoolkit\src\examples\examples.mcp

To build the tutorials use the project file in metoolkit\src\tutorials\tutorials.mcp

The following must be done once, or whenever the relevant paths are changed:

* After selecting the 'Edit->Preferences...' menu entry, ensure that in 'General->Source Trees' there are two definitions, one for 'SCE', the top directory of the Sony toolchain tree, and one for 'MEPS2', the top directory of Karma for PS2 tree.

You will also need to copy the

CodeWarrior\Stationery\PlayStation2 - 2.0.0\c++\LinkSegment_PS2.lcf

file to

metoolkit\include\MeCWPS2.lcf

Creating a new project

There is an empty project (template) that contains all of the following in src/cwstationary/ We include the following information in the event that you want to add this to an existing project.

The following must be done for every target in every project you create:

* Add to the target 'Settings' 'Target->File Mappings' mappings for the extensions ".o" and ".a" as "Lib Import MIPS".

* Add to the target 'Settings' 'Target->Access Paths' the 'User Paths' for the Sony toolchain, for example, for version 1.6.6 with the 2.1.3 libraries:

{SCE}ee/gcc/lib/gcc-lib/ee/2.9-ee-991111-01

{SCE}ee/gcc/lib/gcc-lib/ee/2.9-ee-991111

{SCE}ee/gcc/ee/lib

{SCE}ee/lib

They don't need to be recursively scanned.

* Add to the target 'Settings' 'Target->Access Paths' the 'System Paths' for the Sony toolchain, for example, for version 1.6.6 with the 2.1.3 libraries:

{SCE}ee/gcc/lib/gcc-lib/ee/2.9-ee-991111-01/include

{SCE}ee/gcc/ee/include

{SCE}ee/include

They don't need to be recursively scanned.

* Add to the target 'Settings' 'Target->Access Paths' the 'User Paths' for Karma for PS2:

{MEPS2}include

{MEPS2}lib.rel/ps2<- for release builds

{MEPS2}lib.chk/ps2<- for checked or debug builds

They don't need to be recursively scanned.

* Add to the target 'Settings' 'Target->Access Paths' the 'System Paths' for Karma for PS2:

{MEPS2}include

It does not need to be recursively scanned.

* In the target 'Settings', set 'Target->MIPS Bare Target->Small Data' to zero.

* In the target 'Settings', enable 'Language Settings->C/C++ Language->Enums Always Int'.

* In the target 'Settings', set the 'Language Settings->C/C++ Language->Prefix File' field to "MeCWPS2Prefix.h" (for release builds) or "MeCWPS2PrefixCheck.h" (for checked or debug builds), or to the name of a header file of your own that includes either of them as appropriate.

The following must be done for every project:

* Add to the projects's 'Files' list the runtime libraries in '{MEPS2}lib.rel/ps2' (for a release build) or '{MEPS2}lib.chk/ps2' (for a checked or debug build).

* Add to the project's 'Files' list:

{SCE}ee/lib/crt0.s
{Compiler}PS2 Support/gcc_wrapper.c
{MEPS2}include/MwCWPS2.lcf

All these actions have been incorporated in a sample project file, 'Program.mcp', included with the PS2 version of Karma.

This project file can be copied to an appropriately named directory under the '{Compiler}Stationery' directory and then used as CodeWarrior project stationery.