mathengine

MathEngine Karma™ Installation Guide

May 2002.

Accompanying Karma Version 1.2.1.

MathEngine Karma Installation Guide.

Karma Collision uses Qhull (c) to construct convex models from points. Qhull is a software package used for computing the convex hull and related geometrical structures. It is available free of charge from the University of Minnesota Geometry Center.

Installation 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 demo source code - is available for PlayStation®2, GameCube, Xbox and Windows.

The evaluation software can be downloaded in the following single precision formats:

File	Description
`Karma-1.2.1_ps2_static.tar.gz`	PlayStation®2 compressed tar archive.
`Karma-1.2.1_xbox_libcmt.zip`	Xbox built against LIBCMT.
`Karma-1.2.1_xbox_libc.zip`	Xbox built against LIBC.
`Karma-1.2.1_win32_msvcrt.zip`	Win32 built against MSVCRT.
`Karma-1.2.1_win32_libcmt.zip`	Win32 built against LIBCMT.
`Karma-1.2.1_win32_libc.zip`	Win32 built against LIBC.

The following Karma builds are available to licensed customers:

- Beta version of Karma 1.2.1 (`Karma-1.2.1_ngc_beta.zip`) for Nintendo GameCube.
- Double precision builds.
- Linux build.

Please contact support@mathengine.com for details.

Note that, depending on which version you have downloaded, the version number 1.2.1 may be different.

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

Downloading Karma

To download Karma from the MathEngine website:

- register at http://www.mathengine.com/
- your account will be activated within 2 working days by MathEngine.

Prerequisites

Target Platforms

Supported runtime platforms are PlayStation®2, GameCube, Win2000, Win98, and Xbox .

Development Platforms

Supported development platforms are:

- x86
  - Win32 (Cygwin, MS Visual C++ 6.0)
  - Xbox (Cygwin, MS Xbox SDK)
  - Linux
- PlayStation®2 (Compatible with the Sony libraries 2.2.0, and 2.4.3)
  - Linux (ee-gcc 2.91, 2.95 and 2.96)
  - CodeWarrior for PlayStation®2 R3.01. Project files are provided.
  - SN Systems version 2.2 (using make or Developer Studio).

Note: CodeWarrior requires a patch to be compatible with ee-gcc 2.96. Karma has been tested with Codewarrior and ee-gcc 2.91. Karma should be compatible with ee-gcc 2.95 but has not been tested. Karma does not support ee-gcc 2.96.

Note: SN is not compatible with ee-gcc 2.96. The prebuilt libraries provided with Karma are built using ee-gcc 2.91. Customers will be able to use Karma with ee-gcc 2.91 or 2.95.

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

Installation Directory Structure

After unpacking, the Karma directory structure (the Win32 libc build is given) 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
    |   |
    |   \---glut                       OpenGL Utility Toolkit API
    |
    +---bin                            Executable files
    |   +---Demos                      Example exe programs
    |   |
    |   +---TechDemos                  Technical demonstration exe programs
    |   |
    |   +---GreaseMonkey               GreaseMonkey Demo: README,
    |   |                                executable, scripts, etc
    |   |
    |   \---Resources                  .bmp .dll .dtd and .ka files
    |
    +---doc                            Documentation. index.html is the start-up
    |                                    page.This includes the:
    |                                      Userguide
    |                                      Installation Guide
    |                                      changes and readme text files
    |                                      Doxygen files
    |                                      MS Help chm files
    |
    +---include                          Header files
    |
    +---lib.chk                          Check Libraries
    |
    +---lib.rel                          Release Libraries
    |
    \---src                              Source files
        |
        |   README_build.TXT
        |   makefile                     Top-level makefile
        |   MeToolkits.dsw               Top-level workspace for Visual C++
        |
        +---components                   The Karma source for the provided
        |                                  dynamics, collision and other code.
        |
        +---Demos                        Example programs .c .dsp and makefile
        |
        \---TechDemos                    Technical demos .c, .dsp, makefile
```

Building the Demos

Using Visual C++

Note: Visual C++ 6.0, Visual C++ 6.0 service pack 5 and visual C++ processor pack 5 are required.

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

- Open `metoolkit\src\MeTookits.dsw`
- Click **Build -> Batch Build**.
- In the Batch Build dialog box, make sure that all components are selected.
- 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) instead of LIBC.
- kernel32.lib
- user32.lib
- advapi32.lib
- gdi32.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

Note: Visual C++ 6.0, Visual C++ 6.0 service pack 5 and visual C++ processor pack 5 are required.

All the source code of this release, including the demos and technical demos, 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 v3.79 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 PlayStation®2, 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 demos or tech demos or a single component by going to that subdirectory and executing the following commands (e.g. for tech demos):

```
cd src/techdemo
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 demo and tech demo executables are put in `demos/bin.rel` and `techdemos/bin.rel` respectively for release versions. `Debug` versions will be generated into `demos/bin.dbg` and `techdemos/bin.dbg`. `Check_release` builds will generate files into `demos/bin.chk` and `techdemos/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:

www.cygwin.com

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`).

Sony PlayStation®2: 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_ps2.txt in the metoolkits/src folder.

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

Sony PlayStation®2: Building from Win32, using SN Systems

The Sony PlayStation®2 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 PlayStation®2 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 demo programs and library source using the supplied makefiles. For example, the makefile for the Ballman example can be found in

…..\metoolkit\src\demos\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 plugin you will have to create the .DSP from the makefile.

In MSDev:

- Create a new PlayStation®2 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:

    libMeApp.a

    libMeAssetDB.a libMeAssetDBXMLIO.a libMeXML.a libMeAssetFactory.a

    libMst.a libMeViewer2.a

    libMdt.a libMdtKea.a libMdtBcl.a

    libMcdPrimitives.a libMcdConvex.a libMcdConvexHull.a libMcdCommon.a libMcdFrame.a

    libMcdPrimitives.a libMcdConvex.a libMcdConvexHull.a libMcdCommon.a libMcdFrame.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. Some of the libraries are listed twice. This is needed because of circular dependencies in those libraries.

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 demos use the project file in

    metoolkit\src\demos\demos.mcp

To build the tech demos use the project file in

    metoolkit\src\techdemos\techdemos.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

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

to

    metoolkit\include\MeCWPS2.lcf

Note: CodeWarrior requires a patch to be compatible with ee-gcc 2.96.

Creating a new project

Note: the following is an example only, hence some of the version numbers may be slighly dated.

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 2.2.0 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 2.2.0 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 PlayStation®2 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.