MG Framework User Guide

Modest Game Framework 1.0.0.0

Table of Contents

1 Introduction

MG means Modest Game hinting that this framework is a rather simple game engine. The main focus is instead ease of use and allowing a quick game project startup. It may prove the most useful when seen as an example implementation of different features, which can inspire to similar solutions in other projects. Copy with pride!

1.1 License

MG Framework may be used, modified and distributed in any possible way (including inside commercial pre-compiled applications) as long as these conditions are met:

1. When distributing an application based on MG Framework source the following information string (or similar, with framework name, version and author) must be included in the documentation and/or reproduced by the binary:
   "<this application> is based on MG Framework <version> by Martin Gyllensten."
2. The example application(s) distributed together with MG Framework may not be re-distributed in any way.

Notice that listed SW dependencies (which are not part of but used by MG Framework) have their own licenses and may not be re-distributed in any way not conforming to their respective license.

1.2 Support

MG Framework is not supported per say, but comments are welcome.

2 Installation

2.1 Building an MG Framework application

When building your application (or the distributed example application) all the lib files must be installed correctly, as well as the SDL header files. Good hints are given by Visual Studio if you test build and it doesn't work. Dont expect the Visual Studio project to build ok out of the box as it is depending on having SDL installed a certain way.
Visual Studio 2008 is used to build the distributed example application.

2.2 Running an MG Framework application

If all DLLs are present in the same directory as the EXE there should be no issues getting the example application to run under Windows XP or Windows 7 (including 64 bit).

3 SW architecture

3.1 Classes

3.1.1 MGFramework

Main framework class. This is the class to be inherited from to create an MG Framework based application. When doing so, you are forced to implement three methods; init(...), draw() and handleGameLogics(). This class works as the backbone of the framework as well as a container for graphics functionality and utility functions.

3.1.2 MGMap

MGMap is the class that collects all tiles with their properties, as well as scrolling and any functionality applicable to the tiles such as loading them or saving them (not implemented yet).

3.1.3 MGPeriodicEvent

This class implements a way to create periodic triggers and events in a very simple way. It can be used where game logics are updates to for example have an NPC change walking direction every ten seconds or a message pop up every hour.

3.1.4 MGWindow

The window implementation is quite simple with few options. Width, height, color depth, fullscreen/windowed and window title.

4 Feature descriptions

In this chapter the main features of MG Framework are described. Some very detailed, some not so detailed.

4.1 Debug console

Logs are printed to the console if activated in the example application. The console can also be used to enter debug commands. To do so, hit the ESC button and a prompt (mg>) will be shown in the console. At this point the game loop is not executing anymore. You can give multiple commands after eachother and when you want to resume game loop execution you give the command exit. Below follows a description of implemented commands.

Command	**Description**
help	Displays some help information about available console commands.
quit	Quits the program.
exit	Exits the debug console and resumes game loop execution.
minimap	Toggles the minimap enabled / disabled.
debug / logging	Toggles debug printouts on / off
fps30 / fps60 / fps90 / fps200	Sets desired FPS.

4.2 Map

The map logics are contained inside the MGMap class. Drawing it is the responsibility of the class inheriting from MGFramework and an example can be found in the example application. Map logics include tile to coordinate conversions, scrolling, tile properties and map size.

4.3 Minimap

The minimap feature is included with enable/disable flag only in the framework. The actual implementation is in the example application outside of the framework.

4.4 Scrolling

Scrolling the map can be done with the mouse as in a lot of RTS games. Click and hold down the right button while moving the cursor. The scroll x and y are used as offsets when drawing the scene, as will as when deciding what parts of the scene should be drawn. This is not part of MG Framework but of the class inheriting it (see example application). In the framework the implementation includes event handling in handleEvents as well as underlaying logics in a number of methods and members inside the MGMap class.

4.5 Periodic events

Periodic events (MGPeriodicEvent class) can be used to have things happen with a given period, for example in the handleGameLogics function. The accuracy of the periodic event is given by the smallest game loop time frame, typically the FPS.

4.6 Frames per second

FPS can be configured (desired FPS) and presented (calculated FPS) by calling framework methods. You can also get the number of milli seconds the application sleeps between each frame, indicating if your system has a lot of power left given the current FPS or not..

5 Releases

5.1 MG Framework 1.0.0.0

5.1.1 Release notes

MG Framework 1.0.0.0 is the first version released to the public

5.1.2 Example application(s)

5.1.3 SW dependencies

DLL (LIB)	**Description**	**Version integrated in MG Framework release**
Zlib1.dll	zlib data compression library	1.2.5.0
SDL_ttf.dll (SDL_ttf.lib)	SDL_TTF library	2.0.11.0
SDL.dll (SDL.lib)	SDL	1.2.14.0
libfreetype-6.dll	Libfreetype font engine	2.3.5.2742