Client and Tools Build Guide

BigWorld Technology 2.1. Released 2012.

Software designed and built in Australia by BigWorld.

Level 2, Wentworth Park Grandstand, Wattle St
Glebe NSW 2037, Australia
www.bigworldtech.com

Table of Contents

Chapter 1. BigWorld Technology Client & Tools Build Instructions

1.1. Overview

This document describes how to set up the BigWorld Technology build environment, and build the client and tools.

1.2. Client and Tools Machine Requirements

System requirements for running the client and tools.

1.2.1.1. Client using New Terrain

	Minimum	Recommended
Graphics	Shader model 2.0 • GeForce 6600 128Mb • Radeon 9600 128Mb	Shader model 3.0 • GeForce 7600 • Radeon x1600
CPU	2 GHz	2 GHz
Physical RAM	512 MB	1 GB
Operating System	Windows XP Home 32-bit	Windows 7 32-bit

1.2.1.2. Client using Classic Terrain

	Minimum	Recommended
Graphics	Fixed function • GeForce4 MX • Radeon 7 Series	Shader model 1.1 • GeForce4 Ti • Radeon 9500
CPU	2 GHz	2 GHz
Physical RAM	512 MB	1 GB
Operating System	Windows XP Home 32-bit	Windows 7 32-bit

1.2.1.3. Tools

	Minimum	Recommended
Graphics	Shader Model 3.0 • GeForce 7900 GT • Radeon X1800 XT	Shader Model 3.0+ • GeForce 8800 GTX • Radeon HD 3870
CPU	Athlon 64-bit 3800+ Dual Core Pentium D 820	Athlon 64-bit 4200+ Dual Core Pentium T4200
Physical RAM	2 GB DDR2	2 GB DDR2
Operating System	Windows XP Professional 32-bit	Windows 7 64-bit

1.3. Client and Tools Build Requirements

The minimum requirements for building the client are:

- Visual Studio 2005 or Visual Studio 2008
- DirectX 9 SDK – the February 2010 release was the last release to support Visual Studio 2005
- Platform SDK 6.0 or newer

The minimum requirements for building the tools are:

- Visual Studio 2005 or Visual Studio 2008
- DirectX 9 SDK – the February 2010 release was the last release to support Visual Studio 2005
- Platform SDK 6.0 or newer
- 3ds Max 2008/2009/2010/2011 SDK – required to build the Max exporters
- Maya 2008/2009/2010/2011 SDK – required to build the Maya exporters

1.4. Installing BigWorld Client and Tools

- Check out the BigWorld package from the official Subversion repository.
- If running on a machine without developer tools installed, be sure to install DirectX and Visual C++ run-times which can be found in the BigWorld package:
  - `PackageRoot\vc80-redist\vcredist_x86.exe`
  - `PackageRoot\directx-redist\DXSETUP.exe`

Note

For Vista – If you are changing operating system (for example, going from Windows XP to Vista), and leaving the old build folders in place, then be sure to completely clean all previous output files (.OBJ) and executables from the output folders. You should also remove all previous compiled FX files (.FXO) from the RES path of your project.

1.5. Installing and Configuring Required Software

Note

Some instructions refer to Visual Studio 2005 but are equally applicable to Visual Studio 2008. Be sure to substitute references to `*2005.sln` to `*2008.sln`.

- Install Visual Studio 2005
  - Install Visual Studio 2005 Service Pack 1.
  - Vista only: Install Visual Studio 2005 Service Pack 1 Update for Windows Vista.
- Install Visual Studio 2008
  - Install Visual Studio 2008 Service Pack 1.
  - Install Visual C++ 2008 feature pack

- Install the DirectX SDK.

  Note

  It is recommended to use the February 2010 release as it was the last release to support Visual Studio 2005.

  - Make sure that the DirectX SDK's `include` directory is added to Visual Studio's C++ `includes` search path (see below).
  - Make sure that the DirectX SDK's `library` directory is added to Visual Studio's C++ `libraries` search path (see below).
  - If you have installed MSVC++ after installing DirectX, then make sure that the DirectX folders are listed before the other default folders in the `include` and `link` lists, to give them preference during compiling and linking.
- Install the Windows Platform SDK version 6.0 or later, and make sure this has been added to Visual Studio's C++ `includes` and `libraries` paths at a higher priority than the standard Visual Studio paths.
- BigWorld's source code ships with support for *SpeedTree*, *Umbra* and other third-party libraries turned off by default. In order to turn support on in BigWorld's game client and tools, see the Third-Party Integrations document.
- To add a directory to Visual Studio's C++ `includes` and `libraries` search paths, follow the steps below:
  - Select menu item *Tools→Options*.
  - In the *Options* dialog, select *Projects and Solutions* (or *Projects*, in earlier versions) list item, then *VC++ Directories*.
  - In the *Show Directories For* drop-down list box, select *Include files* or *Library files*.
  - Add the required directories to the list.
- Install the 3ds Max SDK.

  The 3ds Max visual and animation exporters make use of the `3dsMax_sdk/samples/modifiers/morpher/resource.h` header file in order to expose the morpher modifier. A small edit must be made in order for the exporters to compile without errors. Since this file forms part of the 3dsMaxSDK, it do not form part of the BigWorld distributable. The required edit is as follows:

  - Comment the following line in `maxsdk/samples/modifiers/morpher/resource.h` *(line 5)*:

    From: `005 #define IDS_LIBDESCRIPTION 1`

    To: `005 //#define IDS_LIBDESCRIPTION 1`
  - Make sure that in the VC++ options, the DirectX SDK folders are listed higher up than the 3ds Max ones.

1.6. Compiling the Client

- In Visual Studio, load `bigworld/src/client/bwclient2005.sln`.
- Set the `bwclient` project as the startup project.
- Build the `bwclient` project.
- Set the *Solution Configuration* to `Release`.

- To test the client, run `bigworld/bin/bwclient.exe`.

  If you are running the game from within Visual Studio, then set the working folder to `bigworld/bin/client`.

> **Note**
>
> The client can also be compiled as a Python extension module. For more information on how to do this, please refer to the section on *Interactive debugging* in the “Script interactive debugging”.

1.7. Compiling the Tools

- Compile Model Editor
  - Open Visual Studio 2005
  - Load `bigworld/src/tools/modeleditor/modeleditor2005.sln`.
  - Select the *Editor Hybrid Win32* solution configuration.
  - Select *Build All* (`SHIFT+CTRL+B`).
  - The executable `modeleditor.exe` is created in `bigworld\tools\modeleditor`.
- Compile World Editor
  - Open Visual Studio 2005
  - Load `bigworld/src/tools/worldeditor/worldeditor2005.sln`.
  - Select the *Editor Hybrid Win32* solution configuration.
  - Select *Build All* (`SHIFT+CTRL+B`).
  - The executable `worldeditor.exe` is created in `bigworld\tools\worldeditor`.
- Compile Particle Editor
  - Open Visual Studio 2005.
  - Load `bigworld/src/tools/particle_editor/particle_editor2005.sln`.
  - Select the *Editor Hybrid Win32* solution configuration (only Editor configurations are supported).
  - Select `Build All` (`SHIFT+CTRL+B`).
  - The executable `particleeditor.exe` is created in `bigworld\tools\particleeditor`.
- Compile the Animation Exporter
  - Open Visual Studio 2005
  - Load `bigworld/src/tools/animationexporter/animationexporter.sln` or `bigworld/src/tools/animationexporter/animationexporter_2005.sln`.
  - Select the appropriate solution configuration depending on which version of Max you are building for, i.e. *Max2011 Release*.
  - Make sure the maxsdk `include` and `library` folders are set up properly for the configuration.

  - Select `Build All` (`SHIFT+CTRL+B`).
- Compile the Visual Exporter
  - Open Visual Studio 2005
  - Load `bigworld/src/tools/visualexporter/visualexporter.sln` or `bigworld/src/tools/visualexporter/visualexporter_2005.sln`
  - Set `visualexporter` as the startup project.
  - Select the appropriate solution configuration depending on which version of Max you are building for, i.e. *Max2011 Release*.
  - Make sure the maxsdk `include` and `library` folders are set up properly for the configuration.
  - Select `Build All` (`SHIFT+CTRL+B`).
- Compile the Maya Visual Exporter
  - Open Visual Studio 2005
  - Load `bigworld/src/tools/mayavisualexporter/mayavisualexporter.sln` or `bigworld/src/tools/mayavisualexporter/mayavisualexporter_2005.sln`.
  - Set `visualexporter` as the startup project.
  - Select the appropriate solution configuration depending on which version of Maya you are building for, i.e. *Maya 2011 Release*.
  - Make sure the Maya `include` and `library` folders are set up properly for the configuration.
  - Select `Build All` (`SHIFT+CTRL+B`).
- Compile NavGen
  - Open Visual Studio 2005
  - Load `bigworld/src/tools/navgen/navgen.sln`.
  - Select the *Hybrid Win32* solution configuration.
  - Select `Build All` (`SHIFT+CTRL+B`).
  - The executable `navgen.exe` is created in `bigworld/src/tools/misc`.
- Compile the collada converter
  - Open Visual Studio 2008

Note

The collada converter can only be built with Visual Studio 2008

  - Load `bigworld/src/tools/dae2bigworld/dae2bigworld2008.sln`.
  - Select the *Release* solution configuration.
  - Select `Build All` (`SHIFT+CTRL+B`).

  - The executable `dae2bigworld.exe` is created in `bigworld/src/tools/collada_converter`.

- To test the tools, run the executables in the appropriate folder under `bigworld/tools`.

Chapter 2. Additional Information and Troubleshooting

2.1. Build Configurations

2.1.1.1. Client Build Configuration Summary

The client comes with the following main build configurations:

- *Debug*
  - optimisation disabled (/Od)
  - linked against the debug CRT (/MDd)
  - includes full debug information (/ZI)
  - defines `_DEBUG`
  - warning level 3 (/W3), warnings treated as errors (/WX)
- *Hybrid*
  - full optimisation (/OX), favor fast code (/Ot)
  - linked against the CRT with (/MD)
  - includes some debug information (/Zi)
  - defines `NDEBUG` and `_HYBRID`
  - warning level 3 (/W3), warnings treated as errors (/WX)
- *Release*
  - full optimisation (/OX), favor fast code (/Ot)
  - linked against the CRT with (/MD)
  - includes some debug information (/Zi)
  - defines `NDEBUG` and `_RELEASE`
  - warning level 3 (/W3), warnings treated as errors (/WX)
- *Consumer Release*
  - full optimisation (/OX), favor fast code (/Ot)
  - linked against the CRT with (/MD)
  - no debug information
  - defines `NDEBUG`, `_RELEASE` and `CONSUMER_CLIENT`
  - warning level 3 (/W3), warnings treated as errors (/WX)
  - debug and any remaining developer options are disabled. Most of this is set in `cstdmf/config.hpp`.

2.1.1.2. Tools Build Configuration Summary

The tools' build configurations are similar to the client, except they are prefixed with `Editor_`. There are no "release" or "consumer release" build configurations for the tools and "hybrid" should be used instead.

The tools come with the following main build configurations:

- *Editor_Debug*
  - optimisation disabled (/Od)
  - linked against the debug CRT with (/MDd)
  - includes full debug information (/ZI)
  - defines `_DEBUG`
  - warning level 3 (/W3), warnings treated as errors (/WX)
- *Editor_Hybrid*
  - full optimisation (/OX), favor fast code (/Ot)
  - linked against the CRT with (/MD)
  - includes some debug information (/Zi)
  - defines `NDEBUG` and `_HYBRID`
  - warning level 3 (/W3), warnings treated as errors (/WX)

In addition there are some build configurations with `_Eval` and `_Indie` postfixed on their names which are for evaluation and indie BigWorld customers.

2.1.2. Property Sheets

The project configurations use property sheets (`.vsprops` files) to set the common options for each build configuration for each project. They can be found in `bigworld/src/build/vsprops`.

2.1.3. Precompiled Headers

The projects use precompiled headers (.pch files) to increase build speed.

To duplicate this in a new project:

1. Create files called `pch.cpp` and `pch.hpp` in the project directory.
2. Set the project property *Project Properties -> C/C++ -> Precompiled Headers -> Create/Use Precompiled Header* to "Use Precompiled Header (/Yu)".
3. Set the `pch.cpp` property *Properties -> C/C++ -> Precompiled Headers -> Create/Use Precompiled Header* to "Create Precompiled Header (/Yc)".

2.2. Libraries

2.2.1. The C Runtime Library

The client and tools must be linked against the appropriate C Runtime Library (CRT).

BigWorld is compiled dynamically against the CRT, using the `/MDd` option for debug, and `/MD` for release. This decreases the executable size, because it does not include the CRT, but it depends on the user having

MSVCP80.dll or MSVCP90.dll installed correctly, depending on which version of Visual Studio is being used.

This requires running vcredist.exe during installation on end user's computers to ensure the correct DLLs are installed.

BigWorld does not support compiling statically against the CRT, (using the /MTd option for debug and, and /MT for release). This is because care must be taken to ensure that all of the dependent libraries included are also compiled statically against the CRT and BigWorld uses third-party libraries which do not work well with the static CRT. Static linking can also cause memory problems, if for example, CRT allocations are made in the program and then deleted in a library, this is not supported by static linking.

References: C Run-Time Libraries and MSDN Run-Time Library Flags.

2.2.2. Debugging Dependencies

Here are some tools that can be used to help debug library dependency issues.

2.2.2.1. Dumpbin

To get a dump of the dependencies for an executable, you can use the DUMPBIN command line utility that comes with Visual Studio.

For example:

```
>dumpbin /DEPENDENTS "<path_to_client>bwclient.exe"
Microsoft (R) COFF/PE Dumper Version 9.00.30729.01
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file <path_to_client>bwclient.exe

File Type: EXECUTABLE IMAGE

Image has the following dependencies:

  WINMM.dll
  WS2_32.dll
  USER32.dll
  d3dx9_42.dll
  PSAPI.DLL
  KERNEL32.dll
  GDI32.dll
  VERSION.dll
  voip.dll
  python26.dll
  MSVCP90.dll
  MSVCR90.dll
  WININET.dll
  IMM32.dll
  gdiplus.dll
  umbra.dll
  ADVAPI32.dll
  ole32.dll
  SHELL32.dll
  DINPUT8.dll
  d3d9.dll
  SHLWAPI.dll
  fmodex.dll
  fmod_event_net.dll
  OLEAUT32.dll
```

```
Summary

     98000 .data
     10000 .idata
    24E000 .rdata
      4000 .rsrc
    F9C000 .text
      B000 .tls
```

2.2.2.2. Dependency Walker

Another helpful utility is *Dependency Walker* which will give the same information as DUMPBIN, as well as checking the dependencies of the dependencies. See http://www.dependencywalker.com/.

2.3. Common Build Errors

2.3.1. Sharing Violation

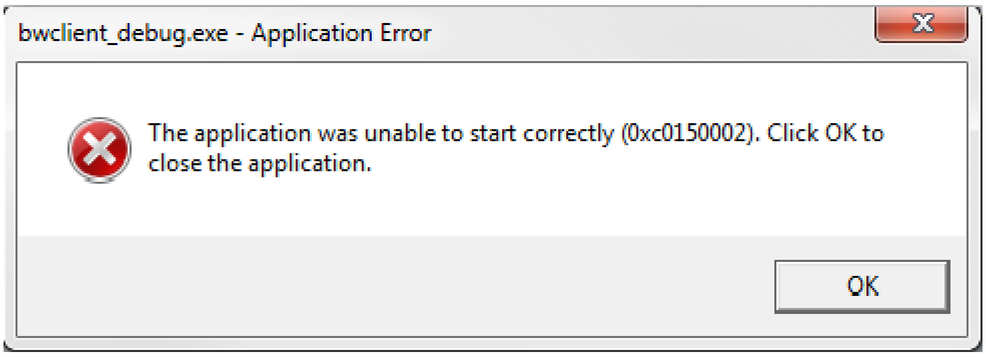
A build can fail if an instance of the client or tools is running while BigWorld builds, due to the Windows file system not allowing files to be modified while they are being used. You might receive an error like the following:

```
1>------ Build started: Project: bwclient, Configuration: Hybrid Win32 ------
1>Compiling...
1>script_bigworld.cpp
1>Performing Pre-Link Event...
1>..\lib\third_party\llmozlib2\llmozlib_dll\llmozlib_dll
\release_2008\llmozlib_dll.dll
1>Sharing violation
1>Project : error PRJ0019: A tool returned an error code from "Performing Pre-
Link Event..."
1>Project : warning PRJ0018 : The following environment variables were not
 found:
1>$(GFXSDK)
1>Build log was saved at "file://d:\bw_2current\bigworld\src\client\obj
\Win32\Hybrid_2008\BuildLog.htm"
1>bwclient - 1 error(s), 0 warning(s)
========== Build: 0 succeeded, 1 failed, 34 up-to-date, 0 skipped ==========
```

To fix this, go into the *Task Manager* and stop any BigWorld client or tools processes that are running, then build.

2.3.2. Application Error

If it's your first time trying to run the debug version of the client and you get an error similar to:


bwclient_debug.exe - Application Error

The application was unable to start correctly (0xc0150002). Click OK to close the application.

```
"The application was unable to start correctly..."
```

Additionally if you run it with the Visual Studio debugger you might see that it got up to loading the Python DLL in the output window before stopping.

This typically happens if you have a copy of `python26_d.dll` which was compiled with an incompatible version of Visual Studio. For example, if you have a copy of the DLL which was compiled with Visual Studio 2005, but the client has been compiled with Visual Studio 2008. To fix this, try compiling the Python DLL with your version of Visual Studio, see “Building for the Client and Content Creation Tools”.