Contents

Chapter 1

Introduction

You should thoroughly read the EULA following this page; it contains important legal information on what you are allowed to do with the Civilization Call To Power 2 sources.

1.1 SOURCE CODE SOFTWARE END USER LICENSE AGREEMENT

SOURCE CODE SOFTWARE END USER LICENSE AGREEMENT

IMPORTANT - READ CAREFULLY:USE OF THE ***CALL TO POWER II***SOURCE CODE IS SUBJECT TO THE SOURCE CODE SOFTWARE END USER LICENSE AGREEMENT TERMS SET FORTH BELOW. **CALL TO POWER II SOURCE CODE**INCLUDES THE SOFTWARE INCLUDED WITH THIS AGREEMENT, THE ASSOCIATED MEDIA, ANY PRINTED MATERIALS, AND ANY ON-LINE OR ELECTRONIC DOCUMENTATION, AND ANY AND ALL COPIES OF SUCH SOFTWARE AND MATERIALS. BY INSTALLING, AND/OR USING THE CALL TO POWER II SOURCE CODE, YOU ACCEPT THE TERMS OF THIS LICENSE WITH ACTIVISION PUBLISHING, INC. (**ACTIVISION**).

LIMITED USE LICENSE. Activision grants you the non-exclusive, non-transferable, limited right and license to install and use one copy of the Call to Power II Source Code solely and exclusively for your personal, non-commercial use. All rights not specifically granted under this Agreement are reserved by Activision and, as applicable, Activisions licensors. The Call to Power II Source Code is licensed, not sold. Your license confers no title or ownership in the Call to Power II Source Code and should not be construed as a sale of any rights in the Call to Power II Source Code.

LICENSE CONDITIONS.

You agree not to sell, rent, lease, license, distribute or otherwise transfer the Call to Power II Source Code, or any copies of the Call To Power II Source Code, without the express prior written consent of Activision.

You agree not to make copies of the Call to Power II Source Code or any part thereof, except for back up or archival purposes, or make copies of the materials accompanying the Call to Power II Source Code.

You agree that, as a condition to your using the Call to Power II Source Code you will not use or allow third parties to use the Call to Power II Source Code and/or the New Game Materials created by you for any commercial purposes, including but not limited to selling, renting, leasing, licensing, distributing, or otherwise transferring the ownership of such New Game Materials, whether on a stand alone basis or packaged in combination with the New Game Materials created by others, through any and all distribution channels, including, without limitation, retail sales and on-line electronic distribution. You agree not to solicit, initiate or encourage any proposal or offer from any person or entity to create any New Game Materials for commercial distribution. You agree to promptly inform Activision in writing of any instances of your receipt of any such proposal or offer.

If you decide to make available the use of the New Game Materials created by you to other gamers, you agree to do so solely without charge.

New Game Materials may be created only if such New Game Materials can be used exclusively in combination with the retail version of *Call to Power II*. New Game Materials may not be designed to be used as a stand-alone product.

New Game Materials must not contain any illegal, obscene or defamatory materials, materials that infringe rights of privacy and publicity of third parties or (without appropriate irrevocable licenses granted specifically for that purpose) any trademarks, copyright-protected works or other properties of third parties.

New Game Materials must contain prominent identification at least in any on-line description and with reasonable duration on the opening screen: (a) the name and E-mail address of the New Game Materials creator(s) and (b) the words THIS MATERIAL IS NOT MADE OR SUPPORTED BY ACTIVISION.

You agree not to export or re-export the Call to Power II Source Code or New Game Materials or any copy or adaptation thereof in violation of any applicable laws or regulations.

OWNERSHIP. All title, ownership rights and intellectual property rights in and to the Call to Power II Source Code and any and all copies thereof are owned by Activision or its licensors. The Call to Power II Source Code is protected by the copyright laws of the United States, international copyright treaties and conventions and other laws. The Call to Power II Source Code may contain certain licensed materials and, in that event, Activisions licensors may protect their rights in the event of any violation of this Agreement. You agree not to remove, disable or circumvent any proprietary notices or labels contained on or within the Call to Power II Source Code.

NO WARRANTIES.THERE ARE NO WARRANTIES, WHETHER ORAL OR WRITTEN, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, AND NO OTHER REPRESENTATIONS OR CLAIMS OF ANY KIND SHALL BE BINDING ON OR OBLIGATE ACTIVISION. THE CALL TO POWER II SOURCE CODE IS PROVIDED TO YOU AS IS.

LIMITATION ON DAMAGES. IN NO EVENT WILL ACTIVISION BE LIABLE FOR SPECIAL, INCIDETAL OR CONSEQUENTIAL DAMAGES RESULTING FROM POSSESSION, USE OR MALFUNCTION OF THE PROGRAM, INCLUDING DAMAGES TO PROPERTY, LOSS OF GOODWILL, COMPUTER FAILURE OR MALFUNCTION AND, TO THE EXTENT PERMITTED BY LAW, DAMAGES FOR PERSONAL INJURIES, EVEN IF ACTIVISION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. ACTIVISIONS LIABILITY SHALL NOT EXCEED THE ACTUAL PRICE PAID FOR THE LICENSE TO USE THIS PROGRAM. SOME STATES/COUNTRIES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS AND/OR THE EXCLUSION OR LIMITAION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITAIONS AND/OR EXCLUSION OR LIMITAION OF LIABILITY MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY HAVE OTHER RIGHTS THAT VARY FROM JURISDICTION TO JURISDICTION.

TERMINATION. Without prejudice to any other rights of Activision, this Agreement will terminate automatically if you fail to comply with its terms and conditions. In such event, you must destroy all copies of the Call to Power II Source Code and all of its component parts.

U.S. GOVERNMENT RESTRICTED RIGHTS. The Call to Power II Source Code and documentation have been developed entirely at private expense and are provided as Commercial Computer Software or restricted computer software. Use, duplication or disclosure by the U.S. Government or a U.S. Government subcontractor is subject to the restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clauses in DFARS 252.227-7013 or as set forth in subparagraph (c)(1) and (2) of the Commercial Computer Software Restricted Rights clauses at FAR 52.227-19, as applicable. The Contractor/Manufacturer is Activision, Inc., 3100 Ocean Park Boulevard, Santa Monica, California 90405.

INJUNCTION. Because Activision would be irreparably damaged if the terms of this Agreement were not specifically enforced, you agree that Activision shall be entitled, without bond, other security or proof of damages, to appropriate equitable remedies with respect to breaches of this Agreement, in addition to such other remedies as Activision may otherwise have under applicable laws.

INDEMNITY. You agree to indemnify, defend and hold Activision, its partners, licensors, affiliates, contractors, officers, directors, employees and agents harmless from all damages, losses and expenses arising directly or indirectly from your acts and omissions to act in using the Product pursuant to the terms of this Agreement

MISCELLANEOUS. This Agreement represents the complete agreement concerning this license between the parties and supersedes all prior agreements and representations between them. It may be amended only by a writing executed by both parties. If any provision of this Agreement is held to be unenforceable for any reason, such provision shall be reformed only to the extent necessary to make it enforceable and the remaining provisions of this Agreement shall not be affected. This Agreement shall be construed under California law as such law is applied to agreements between California residents entered into and to be performed within California, except as governed by federal law and you consent to the exclusive jurisdiction of the state and federal courts in Los Angeles, California.

If you have any questions concerning this license, you may contact Activision at 3100 Ocean Park Boulevard, Santa Monica, California 90405, USA, (310) 255-2000, Attn. Business and Legal Affairs, legal@activision.com.

Chapter 2

Setting up your build environment

This chapter deals with setting up your build environment, i.e. take the necessary step to get ready to build CtP2 from the source.
If you have already setup your build environment, you can continue with the next chapter.

2.1 Windows

2.1.1 System requirements

Your system is supposed to meet the following system requirements:

- A PC with a Pentium II 266 MHz processor or higher
  (or comparable compatible processor (AMD/Cyrix/IBM/Via))
- At least 64 MB RAM
- Plenty of space on your hard disc (a few GByte, if you keep backup copies)
- Windows 98, 98 SE, ME, NT, 2000, XP or later

2.1.2 Install Civilization Call to Power II

Provided you haven't already installed Windows and Civilization Call to Power II, you need to install them now.

For Call to Power II, choose a full setup (e.g. to *C:\Program_Files\Activision\CtP2*).

Refer to the documentation bundled with your software on how to do this.

2.1.3 Install the CtP2 1.11 patch

Download and install the CtP2 v1.11 patch from here.

2.1.4 Backup your CtP2 directory

Create a copy of your CtP2 installation.

2.1.5 Install MS Visual C++ 6.0

Install MS Visual C++ 6.0, if you did not do so already.

Refer to the documentation bundled with your software on how to do this. Afterwards, download Visual Studio 6.0 Service Pack 6 and follow the installation instructions on the same link.

Note: Instead of MS Visual C++ 6.0, you may also install the Visual C++ 6.0 Introductory Edition, MS Visual C++ 6.0 Authorenedition or any other localized version of the Introductory Edition. You still find it shipped with some C++-Books, and you may succeed upgrading them with Visual C++ 6.0 SP6.

If you intend to use .NET, you may need to do some porting work (keeping backwards compatibility to Visual Studio 6.0).

2.1.6 DirectX SDK Setup

There are two possible and tested DirectX setups for now.
You can choose between the DirectX SDK 7.0 and DirectX SDK 9.0b. If you choose the first one, you have the same SDK versions that CtP2 was compiled with. This means that additionally you'll have to install the Direct Media SDK 6.0 as well.

If you choose to use DirectX SDK 9.0b, you don't need the Direct Media SDK 6.0, but have to compile the DirectShow BaseClasses shipped with the DirectX SDK yourself.

2.1.6.1 Alternative 1: Using DirectX SDK 7.0

At the time of writing, DirectX SDK 7.0 has become unavailable. If you have a link, please place a post in the CtP2-Source Code Project forum.

Install the DirectX 7.0 SDK. Then, download the Direct Media 6.0 SDK and follow the installation instructions.

Launch the Visual C++ IDE. Select menu *Tools*, then submenu *options*. Within the upcoming dialog, select tab *Directories*. Select the include directories path, and make sure, these directories are at the top of the path:

```
C:\DXSDK\Include
C:\DXMedia\Include
C:\DXMedia\Classes\Base
```

Then, select the lib directories path, and make sure, these directories are at the top of the path:

```
C:\DXSDK\Lib
C:\DXMedia\Lib
```

Note: If you installed the SDKs to somewhere different, of course the paths C:\DXSDK and C:\DXMedia must be replaced with the locations of your installations.

Note: The download link to the DXMedia SDK may become inactive soon, the SDKs are not listed at the official site at DirectX download site.

2.1.6.2 Alternative 2: Using the latest DirectX SDK

At time of writing, the latest DirectX SDK can be downloaded here. If the link doesn't work, download the latest DirectX SDK from here. Then, follow the instructions on the bottom of the link to install the SDK. Make sure you select "Headers and Libraries" and the C++ Samples as well.

Launch the Visual C++ IDE. Select menu *Tools*, then submenu *options*. Within the upcoming dialog, select tab *Directories*. Select the include directories path, and make sure, these directories are at the top of the path:

```
C:\DXSDK\Include
```

Then, select the lib directories path, and make sure, these directories are at the top of the path:

```
C:\DXSDK\Lib
```

Afterwards, open the DirectShow BaseClasses workspace by loading the file *C:\DXSDK\Samples\C++\DirectShow\BaseClasses\baseclasses.dsw*.

Build the release version (strmbase.lib):
Open *Build - Set active configuration* and select *BaseClasses - Win32 Release*. Then run *Build - strmbase.lib*.

Build the debug version (strmbased.lib):
Open *Build - Set active configuration* and select *BaseClasses - Win32 Debug*. Then run *Build - strmbased.lib*.

Again, select menu *Tools*, then submenu *options*. Within the upcoming dialog, select tab *Directories*. Select the include directories path, and make sure, these directories are at the top of the path in this order:

```
C:\DXSDK\Samples\C++\DirectShow\BaseClasses
C:\DXSDK\Include
```

Then, select the lib directories path, and make sure, these directories are at the top of the path, in that order:

```
C:\DXSDK\Samples\C++\DirectShow\BaseClasses\Debug
C:\DXSDK\Samples\C++\DirectShow\BaseClasses\Release
C:\DXSDK\Lib
```

2.1.7 Optional: Install Simple DirectMedia Layer libraries

If you want to build CtP2 against Simple DirectMedia Layer, you also need to install the SDL libraries mentioned below.

2.1.7.1 Install SDL 1.2.7

Download SDL-devel-1.2.7-VC6.zip and unpack it to *C:\libs*. If the download doesn't work, proceed so with the latest version from http://www.libsdl.org/download-1.2.php.

2.1.7.2 Install SDL_image 1.2.3

Download SDL_image-devel-1.2.3-VC6.zip and unpack it to *C:\libs*. If the download doesn't work, proceed so with the latest version from http://www.libsdl.org/projects/SDL_image/.

2.1.7.3 Install SDL_mixer 1.2.5a

Download SDL_mixer-devel-1.2.5a-VC6.zip and unpack it to *C:\libs*. If the download doesn't work, proceed so with the latest version from http://www.libsdl.org/projects/SDL_mixer/.

2.1.7.4 Add the SDL libraries to your Visual Studio library paths

Launch the Visual C++ IDE. Select menu *Tools*, then submenu *options*. Within the dialog showing up, select tab *Directories*. Select the include directories path, and make sure, these

directories are at the top of the path:

```
C:\libs\SDL-1.2.7\include
C:\libs\SDL_image-1.2.3\include
C:\libs\SDL_mixer-1.2.5\include
```

Finally, select the lib directories path, and make sure, these directories are at the top of the path:

```
C:\libs\SDL-1.2.7\lib
C:\libs\SDL_image-1.2.3\lib
C:\libs\SDL_mixer-1.2.5\lib
```

2.1.8 Unpacking the source

If not already done so, download the source from apolyton.net.
Install or unzip the source to a directory of your choice (e.g. C:\ctp2source).

2.1.9 Setup your environment variables

If you run Windows 98, Windows98 SE or Windows ME, edit your autoexec.bat to contain

```
SET CDKDIR=C:\ctp2source\bin
SET PATH=%PATH%;%CDKDIR%
```

You must make sure that the CDKDIR variable points to the subdirectory bin, where some programs like *byacc.exe* and *flex.exe* exist.

If you run Windows NT, 2000, XP or later, edit your environment variables to contain those paths.

2.1.10 Create a tmp directory

Then, create a directory called 'tmp' at the root of each harddrive, i.e.:

```
C:\tmp
D:\tmp
...
```

You need at least a 'tmp' dir at the drive your CDKDIR is located at and possibly for the Visual C++ 6.0 installation drive.

2.1.11 Reboot

Finally, you'll have to reboot the system.

2.1.12 Obtain latest ALL patch from CtP2 Source Code Project

Go to the PROJECT: Altered source files thread, and look for the latest post with a .zip file containing the string all, e.g. 2004.08.01.CtP2.All.zip. Unpack that file into your ctp2 source code directory, e.g. *C:\ctp2source*.

2.1.13 Optional: Get up to date until latest post

Beginning from the post where you found the latest ALL patch, descend to the latest post and perform the following steps in post order:

1. Download posted files
2. Foreach posted file, do the following:
3. If posted file is a .zip or .rar archive, and there are no instructions, unzip it to your ctp2 source code directory.
   Perhaps you get warnings by your zip-Software regarding files to be overwritten. This is ok, so let your packer replace your local files by the newer ones in the archives you downloaded.
4. If the previous didn't apply, follow the instructions of the post where you got the file from to apply the changes to your ctp2 source code directory.

2.1.14 Make a backup of the clean source directory

Backup the sources to another directory. It will help you playing with the sources, later.

Chapter 3

Building Civilization Call To Power 2

This chapter contains information, how you start the build process.

3.1 Windows

3.1.1 Launch Microsoft Visual C++ IDE

If you haven't done so already, launch MS VC++ IDE.

3.1.2 Opening the workspace

Choose *File - Open* and start with the installation directory of the CtP2 source code (e.g.: *C:\ctp2source*). Click through the directories to open *C:\ctp2source\ctp2_code\ctp\civctp.dsw*.

Now you have the choice between building the DirectX version and the SDL version. If you haven't installed the SDL libraries, follow the next section, else continue with the SDL section.

3.1.3 Compiling the DirectX version

3.1.3.1 Set the active configuration

Open *Build - Set active configuration* and choose *ctp2 - Win32 Debug*.

3.1.3.2 Start the build

Run *Build - CivCTP_dbg.exe*. Have yourself a break while the source compiles. This lasts about 20 - 30 minutes, but will vary depending on your system.

3.1.3.3 Copy CivCTP_dbg.exe to your CtP2 installation

Go to *C:\ctp2source\ctp2_code\ctp* and copy *CivCTP_dbg.exe* to the *ctp2_program\ctp* subdirectory of your CtP2 installation (e.g. into *C:\Program_Files\Activision\CtP2\ctp2_program\ctp*).

3.1.3.4 Copy anet2d.dll to your CtP2 installation

Repeat the previous step for the file *anet2d.dll*.

3.1.3.5 Copy the ctp2_data and ctp2_program directories

Copy the ctp2_data directory and the ctp2_program directory recursively to your CtP2 installation path (e.g. into *C:\Program_Files\Activision\CtP2\ctp2_program\ctp*). If you are prompted for overwriting files, respond with "Yes for All" to make sure you get the newest version.

3.1.3.6 Run the executable CivCTP_dbg.exe

You can now run your self build Civilization Call to Power 2. If you get some assertion errors, ignore them first and see how things are going.
You can then start diving into the code and fixing bugs (or assertions by assuring the condition within the Assert() won't happen) or adding new features.

3.1.4 Compiling the SDL version

3.1.4.1 Set the active configuration

Open *Build - Set active configuration* and choose *ctp2 - SDL Debug*.

3.1.4.2 Start the build

Run *Build - CivCTP_SDL_dbg.exe*. Have yourself a break while the source compiles. This lasts about 20 - 30 minutes, but will vary depending on your system.

3.1.4.3 Copy CivCTP_SDL_dbg.exe to your CtP2 installation

Open the Windows Explorer and go to the path *C:\ctp2source\ctp2_code\ctp*. Copy *CivCTP_dbg.exe* to the *ctp2_program\ctp* subdirectory of your CtP2 installation (e.g. into *C:\Program_Files\Activision\CtP2\ctp*

3.1.4.4 Copy anet2d.dll to your CtP2 installation

Repeat the previous step for the file *anet2d.dll*.

3.1.4.5 Copy the ctp2_data and ctp2_program directories

Copy the ctp2_data directory and the ctp2_program directory recursively to your CtP2 installation path (e.g. into *C:\Program_Files\Activision\CtP2\ctp2_program\ctp*). If you are prompted for overwriting files, respond with "Yes for All" to make sure you get the newest version.

3.1.4.6 Copy the SDL libraries to your CtP2 installation

Go into C:\libs and copy all .dll files located within the lib subdirectory of each SDL directory to the *ctp2_program\ctp* subdirectory of your CtP2 installation path (e.g. into *C:\Program_Files\Activision\CtP2*

3.1.4.7 Run the executable CivCTP_dbg.exe

You can now run your self build Civilization Call to Power 2. If you get some assertion errors, ignore them first and see how things are going.
You can then start diving into the code and fixing bugs (or assertions by assuring the condition within the Assert() won't happen) or adding new features.

Appendix A

Definitions

On the next pages you will find the preprocessor definitions in the source and how they actually affect a build.

A.1 Preprocessor definitions used in Call to Power 2

A.1.1 Additional #defines

__AUI_USE_DIRECTMEDIA__
Default: defined (*ui/aui_common/auicfg.h*)
Description: Use DirectMedia

__AUI_USE_DIRECTX__
Default: defined (*ui/aui_common/auicfg.h*)
Description: Use DirectX

__BIG_DIRTY_BLITS__
Default: Always undefined
Description: undefined (never used, *gfx/tilesys/tiledmap.cpp*)

__GRIDDED_BLITS__
Default: defined (*gfx/tilesys/tiledmap.cpp*)
Description: ??? (*gfx/tilesys/tiledmap.cpp*)

__GW_USE_EXPORT
Default: default within *GameWatch/gamewatch/gamewatch.dsp*, all other projects using gamewatch should use **__GW_USE_IMPORT**
Description: Use __declspec(dllexport) in gamewatch headers, i.e. export symbols for linking a dll

__GW_USE_IMPORT
Default: default within *ctp/civctp.dsp*, *GameWatch/gwarchive/gwarchive.dsp*, *GameWatch/gwciv/gwciv.dsp*, *GameWatch/gwfile/gwfile.dsp*
Description: Use __declspec(dllimport) in gamewatch headers, i.e. prevent linking symbols by just providing declarations of functions used

__GWARCHIVE_USE_EXPORT
Default: default within *GameWatch/gwarchive/gwarchive.dsp*, all other projects using gwarchive should use **__GWARCHIVE_USE_IMPORT**
Description: Use __declspec(dllexport) in gwarchive headers, i.e. export symbols for linking

__GWARCHIVE_USE_IMPORT
Default: undefined
Description: Use __declspec(dllimport) in gwarchive headers, i.e. prevent duplicate symbols by just providing declarations of functions used

__GWCIV_USE_EXPORT
Default: default within *GameWatch/gwciv/gwciv.dsp*, all other projects using gwciv should use **__GWCIV_USE_IMPORT**
Description: Use __declspec(dllexport) in gwciv headers, i.e. export symbols for linking

__GWCIV_USE_IMPORT
Default: default within *ctp/civctp.dsp*
Description: Use __declspec(dllimport) in gwciv headers, i.e. prevent duplicate symbols by just providing declarations of functions used

__GWFILE_USE_EXPORT
Default: default within *GameWatch/gwfile/gwfile.dsp*, all other projects using gwfile should use **__GWFILE_USE_IMPORT**
Description: Use __declspec(dllexport) in gwfile headers, i.e. export symbols for linking

__GWFILE_USE_IMPORT
Default: undefined
Description: Use __declspec(dllimport) in gwfile headers, i.e. prevent duplicate symbols by just providing declarations of functions used

__MAKESPR__
Default: undefined
Description: ???

__SPRITETEST__
Default: undefined
Description: ???

__TILETOOL__
Default: undefined
Description: ???

__USING_SPANS__
Default: undefined
Description: ???

_ACTOR_DRAW_OPTIMIZATION
Default: undefined
Description: Try to prevent redrawing of sprites which have already been drawn

_AI_BACKDOOR
Default: ???
Description: ???

_AIDLL
Default: undefined (unless *robotcom.dsp* is active)
Description: ???

BFR
Default: default only within civctp.dsp for configuration *ctp2 - Win32 Final*
Description: Build Final Release

_DEBUG
Default: defined for Debug configurations
Description: Activate additional code for debugging

_DEBUG_INCOMPATIBLE
Default: undefined
Description: ???

_DEBUG_MEMORY
Default: defined for configurations *Win32 - Debug* and *Win32 - Debug Browse*
Description: Generates extra memory debug information (needs **_DEBUG** and **_MEMORYLOGGING**)

_DEBUG_SCHEDULER
Default: undefined
Description: Adds additional assertions for schedular in ai/strategy

_DEBUGTOOLS
Default: defined for configuration *Win32 Debug*
Description: Use debug logs (ctp/debugtools)

JAPANESE
Default: defined within *civctp_j.dsp* only
Description: Build Japanese version

_MEMORYLOGGING
Default: undefined
Description: Generates extra memory debug information (needs **_DEBUG** and **_DEBUG_MEMORY**)

_NO_GAME_WATCH
Default: defined in *ctp/civ3_main.h*
Description: Do not generate a GameWatch file

_PLAYTEST
Default: defined for configurations *Win32 Debug*, *Win32 Test*, *Win32 Debug Browse*, *Win32 Optimized Test*
Description: Activates further stuff for a playtest release

_SMALL_MAPPOINTS
Default: defined in *gs/world/MapPoint.h*
Description: MapPoints wont get a Z-Component

_TEST
Default: defined for configuration *Win32 Test*
Description: Should be used for test code, though it only disables warnings regardings overflows in floating point constant arithmetic

_WAS_ABOUT_TEST_WHEN_DAN_GOT_ME_REPRO_STEPS
Default: undefined
Description: Executes a foobar SlicObject for testing

ACTIVISION_ORIGINAL
Default: undefined
Description: Used for marking original code by activision

CELL_COLOR
Default: defined in *gs/world/Cell.h*
Description: Reset Cell-Color in debug-mode (needs **_DEBUG**)

CLEAN_INSTEAD_OF_CONVERT
Default: undefined
Description: Prefer clean rowData over RGB 565 to 555 conversion for sprites

CUSTOM_ALLOCATE
Default: undefined
Description: Overriden new/delete operators in ctp/ctp2_utils/AvlTree.h

CTP1_HAS_RISEN_FROM_THE_GRAVE
Default: undefined
Description: Do not use the CTP2 worker utilisation style (i.e. use CTP1 behaviour instead)

CTP1_TRADE
Default: undefined
Description: Enable CTP1 trade behaviour

DUMP_ASTAR
Default: undefined
Description: Enable command to dump astar findpath callstack to a filename

GENERATE_ADDRESS_LOG
Default: undefined
Description: Generate a file address.log upon termination, which contains the entry points and names of functions registered via Debug_AddFunction(...)

IANSCROLL
Default: undefined
Description: ???

MEMORY_LOGGED
Default: defined, when **_DEBUG_MEMORY** is defined in *ctp/debugtools/debugmemory.h* (else undefined)
Description: Used to activate leak tracing code in ctp/debugtools/debugmemory.cpp

MEMORY_FAST
Default: defined, when **_DEBUG_MEMORY** is not defined in *ctp/debugtools/debugmemory.h* (else undefined)
Description: ??? Uses an own heap to prevent leaks, no logging

NDEBUG
Default: undefined
Description: Do a release build, i.e. remove asserts, etc.

NETWORK_PARANOID
Default: undefined
Description: Verify army position during each order execution

NON_STANDART_C_PLUS_PLUS
Default: undefined
Description: Use Activision code which is not standard compliant

REQUIRE_CORRECT_LOG
Default: undefined
Description: Not implemented, perhaps intended for verifying the map file to match executable symbols for logging correct entry points (e.g. for leaks)

SUPER_DEBUG_HEURISTIC
Default: undefined
Description: ???

SUPER_DEBUG_SPEED
Default: ??? defined (*robotcom/planner/Scheduler.h*)
Description: ???

TEST_APP
Default: undefined
Description: ???

TEST_DRIVER
Default: undefined
Description: Activates test replacements in ai/strategy/goal/Goal.cpp

tracklen_CRYPTLOG
Default: undefined
Description: Activate logging of crypt part within tracklen code (needs **tracklen_LOGGING**)

tracklen_LOGGING
Default: undefined
Description: Activate logging of tracklen code

USE_MILES_STUBS
Default: undefined
Description: Deactivate sound and mss32.lib/mss32.dll dependency by using stub functions

USE_NDBG
Default: ???
Description: ???

USE_LOGGING
Default: undefined
Description: Enable logging even on non-debug version

USE_SDL
Default: defined for *Win32 SDL* targets
Description: Use SDL instead of native MM-libs (so far replaces only miles sound)

USE_STOP_ZERO_MOVEMENT
Default: undefined
Description: Prevents unit without movement points from moving

A.1.2 Standard #defines (system specific)

_MBCS
Default: ???
Description: Use multibyte characters

_UNICODE
Default: ???
Description: Use unicode (wide) characters

_WINDOWS
Default: must be defined on Windows and must not be defined on any other platform
Description: Build on a windows system

HAVE_CONFIG_H
Default: ???
Description: Include config.h generated by autoconf/autoheader

LINUX
Default: must be defined on Linux and must not be defined on any other platform
Description: Build on linux

WIN32
Default: must be defined on Windows and not be defined on any other platform
Description: Build on a WIN32 platform

A.1.3 Compiler #defines

__cplusplus
Default: defined, when compiling on C++
Description: Set by the C++ Compiler

__GNUC__
Default: defined, when compiling with the GNU C compiler
Description: GNU C of GNU Compiler Collection

_MSC_VER_
Default: defined by the Visual C/C++ compiler to its version
Description: MS Visual C++/.NET

Appendix B

Build system

B.1 Windows - Visual C++ Configurations

Every windows configuration contains at least these macro definitions: _WINDOWS, WIN32.

B.1.1 ctp/civctp.dsp

ctp2 - Win32 Debug
Target: CivCTP_dbg.exe
Description: Standard Win32 Configuration for building an executable with debug information.
Defines used: __GW_USE_IMPORT, __GWCIV_USE_IMPORT, _AI_BACKDOOR, _DEBUG, _DEBUG_MEMORY, _DEBUGTOOLS, _PLAYTEST

ctp2 - Win32 Debug Browse
Target: CivCTP_dbg.exe
Description: This configuration builds the same executable like ctp2 - Win32 Debug, but copies the files on drive R (has been used by activision for remote debugging).
Defines used: __GW_USE_IMPORT, __GWCIV_USE_IMPORT, _AI_BACKDOOR, _DEBUG, _DEBUG_MEMORY, _DEBUGTOOLS, _PLAYTEST

ctp2 - Win32 Final
Target: ctp2.exe
Description: Builds a final release like the executable shipped on the CtP2 CD.
Defines used: __GW_USE_IMPORT, __GWCIV_USE_IMPORT, _BFR_, NDEBUG

ctp2 - Win32 NDebug
Target: CivCTP_ndbg.exe
Description: ???
Defines used: __GW_USE_IMPORT, __GWCIV_USE_IMPORT, _AI_BACKDOOR, _DEBUG, NDEBUG, USE_NDBG

ctp2 - Win32 Optimized Test
Target: ctp2.exe
Description: Builds a playtest executable without debugging information (overwrites final target).
Defines used: __GW_USE_IMPORT, __GWCIV_USE_IMPORT, _PLAYTEST, NDEBUG

ctp2 - Win32 Release
Target: ctp2r.exe
Description: Standard Win32 Configuration for building an executable without debug information.
Defines used: __GW_USE_IMPORT, __GWCIV_USE_IMPORT, NDEBUG

ctp2 - Win32 Test

Target: CivCTP_test.exe

Description: Build a Test release.

Defines used: __GW_USE_IMPORT, __GWCIV_USE_IMPORT, _AI_BACKDOOR, _DEBUG, _PLAYTEST

ctp2 - SDL Debug

Target: CivCTP_SDL_dbg.exe

Description: This builds the debug version of CtP2 using SDL instead of DirectX (sound only, so far; the remaining configuration is the same as ctp2 - Win32 Debug.

Defines used: __GW_USE_IMPORT, __GWCIV_USE_IMPORT, _AI_BACKDOOR, _DEBUG, _DEBUG_MEMORY, _DEBUGTOOLS, _PLAYTEST, USE_SDL

ctp2 - SDL Final

Target: ctp2_SDL.exe

Description: This builds the final version of CtP2 using SDL instead of DirectX (sound only, so far; the remaining configuration is the same as ctp2 - Win32 Final.

Defines used: __GW_USE_IMPORT, __GWCIV_USE_IMPORT, _BFR_, NDEBUG, USE_SDL

Appendix C

Code

This chapter deals with code. Currently, just dead code is listed.

C.1 Dead code list

Within this list you see code and files which are **definitely** dead, i.e. not needed anymore. It's most likely that these files once may be scheduled for removal and even don't appear in this list.

`ctp2_code/gfx/gfx.dsp`
`ctp2_code/gs/gs.dsp`
`ctp2_code/net/net.dsp`
`ctp2_code/ui/ui.dsp`

C.2 Possibly dead code list

This list contains code and files which maybe dead, but maybe aren't. Though these files appear to be unused this time, they may never be removed until they have been identified as dead code **without any doubts** and been in the dead code list for some while.

`ctp2_code/robotcom/robotcom.dsp`
`ctp2_code/robotcom/robotcom.mak`
`ctp2_code/user-robotcom/robotcom.dsp`

Appendix D

FAQs

D.1 CtP2 Source Code Project FAQ (v2)

D.1.1 What is this FAQ for?

This FAQ is for general questions related to the CtP2 Source Code Project as a whole and general questions regarding the source code. If you have a such question but it's not listed here, feel free to post it. If you have specific question about the code itself, what is or isn't in it, if we can implement specific features, etc, please start a new thread about it (or use an existing one).

This is the second version of the FAQ, the old one ran out of space. You can still find it here: http://apolyton.net/forums/showthread.php?s=&threadid=100290. Note that this FAQ extends over several posts, don't stop reading after the end of this post

D.1.2 Why document the code?

Of course, the CtP2 source code is an extremely complex piece of work. It is expected to be extremely difficult to understand, and there will be an infinite number of ways to change it once we do. There are an estimated 1.8 million lines of coude – too much for a single person to work through on his/her own. That's why this Source Code Project exists: if everyone involving in the code works together to document it (what can be found where, and how everything works), this will make it much easier for everyone to understand and modify the code.

D.1.3 What about changing the code?

Once we start making changes to the code, there is a huge risk that everyone will try to make different changes and have very different views on what the improved game should be like and which changes should and shouldn't be made. As with the mods, there may eventually be many different versions of the game. This Source Code Project will also serve to develop a more or less 'official' Apolyton version of the (improved) game, which will make those changes to the game which the CtP community as a whole feels are most needed – in the same way that the Apolyton Pack does this for the mods (in both CtP1 and CtP2). Hopefully this will also serve to more or less control the number of different versions of the game that come out, as things could get really ugly for players if there are 400 different versions of the game. By combining our views and ideas as much as possible into a single vision, we can hopefully create one version of the game that encompasses many people's opinions on what the game should be like and (more or less) satisfies their tastes.

D.1.4 What is this forum for?

This forum is to achieve the goals outlined above. It is to document the code, to figure out how it works and how to get it running, and to discuss changes to the game that could be made to it and how to implement them. Mostly this forum is to support the CtP2 Source Code Project, but if you have any issues related to the CtP2 source code that somehow don't fall under the scope of the CtP2 Source Code Project, this is probably the best place to discuss those as well.

D.1.5 What is the CtP2 source code?

The source code for any software product is like its recipe. If you buy for example a cake in the shop, you normally just eat it. Similarly, if you buy a game, you just play it. CtP2 (and CtP1 as well) has always been somewhat of an exception to this, as you could always modify a lot about the game (and increasingly this is true for other PC games as well). One could say that CtP is a not a finished cake but just a set of ready-made 'sub-products' that you can put together and mix yourself, in whatever way you like best. The source code takes this one step further: instead of just ready-made 'prefab' subcomponents, you get the raw ingredients and the recipe, and can make the cake from scratch all by yourself. This of course also allows you to change anything about the recipe you don't like, replacing or adding ingredients as you please.

For CtP2 this means that we can do anything with the game we want, there are no boundaries whatsoever to what we can change (contrary to the mods). If we wanted to, we could turn this game into a Real-Time Strategy game, or even a First Person Shooter (though in that case we'd have to change so much code it'd probably be better to start from scratch) The only boundaries to what we can do with the source code are our own capabilities to understand and change the code (since most of us aren't professional game designers and there are a few caveats to the release of this source code, some things may simply prove to be too difficult to be worthwhile).

D.1.6 Where can I get the source code? And how can I view it?

The code can be downloaded right here from Apolyton, get it here. It's an 8 MB file. It's an executable: double-click and it will extract the files. But before it does so, it will first show you a user agreement: this is the screen where you usually just click 'I agree' and continue. However, in this case I highly recommend you read it carefully! At least parts of it are not your standard legal disclaimer but were written specifically for this source code and detail what you are and aren't allowed to do with it. Anyway, after you agreed to the End User License Agreement (EULA) about 2650 files and 200 folders are extracted to the folder you specified. Most of these files can be viewed them by opening them in any text editor (e.g. Notepad or Word).

Update (2003-11-6): A zip version of the source code is now also available for download: ctp2source.zip. Note that if you want to link/redistribute this zip file elsewhere, you must ensure that anyone must agree to the EULA before being able to gain access to the files.

D.1.7 How can turn I this code into a working game?

If you just want to have a look at the source code, to see what the code for a professional game looks like, the above file is all you need. If you want to help understand and document the code, you probably won't need much else either. However, if you actually want to be able to compile the game (make it into a executable file that actually works) and modify and test it, you'll need more than that. For one thing, you'll need a compiler, a software program that can transform the code into a working game. For another, you'll need to download about 270 MB worth of DirectX SDK (= Software Development Kit) files from the Microsoft website, as detailed in the readme. The discussion about the details of how to make the file running without errors can be found here. If downloading 270 MB worth of files is a bit much for you, you can also find a workaround for that in that thread.

Update (2004-04-06): Several people have independantly successfully compiled the code, and most of the problems you might encounter in doing so yourself should be addressed in the thread mentioned above. For a brief description of the steps needed to get the code compiling, see the attachment to this post.

D.1.8 Why did Activision release the source code? How about making a no-CD patch?

You may wonder why Activision would release the source code: surely, if anyone can compile the game and edit it as they see fit, can't they just remove the copyright protection and play the game without paying for it? The answer is that they covered this option: the data and media files are not included in the source code file, so you still need to buy a copy of the CD to be able to use the code. Also, in the EULA (user agreement) is set up in such a way that changing the game so that the original game is no longer needed to be able play is strictly forbidden (this means no no-cd patches). Basically, we may edit the game as we see fit, as long as resulting updates/new products still requires the CtP2 CD to play. Releasing the source code was a very generous act of Activision which most software companies would never even consider.

D.1.9 Do I still need to own a copy of CtP2 to be able to play it/ use the source code?

As you can read above, yes, you still need a copy of the game to be able to play it, or you'll miss the data and media files. If you're only interested in understanding how the game works, you can do without the original product, but in that case you need to realized that you can never actually test what will happen if you change certain things about the code, which is often a great help in trying to understand complex products.

D.1.10 This all still sounds too good to be true. What's the catch?

Yes, unfortunately there is a catch. Actually, there are two. The first one is sound: the game uses an external commercial package to control the sound in the game: the Miles Sound Library. Because this is a commercial package not owned by Activision, they had to remove it from the game. This leaves you with a bunch of 'mss.h not found' errors when you try to compile the game. The references to this file and other Miles Sound Library related stuff will need to be removed before the code will run, which will leave you with a working but soundless game.

Update (2003-11-6): Thanks to the excellent work of jonwil, we now have sound working again. See this thread for the files required to reinstate it.

The second caveat is that Activision had to remove all the comments from the code. These comments don't affect the way the code runs, but they greatly facilitate in understanding how it works. The reason for removing these comments is that some of them may have made references that could in some way embarrass Activision of (ex-)Activision employees, or worse: get them into legal or other problems. Because going through millions of lines of code and removing such undesirable comments by hand would have taking a huge amount of resources (and time), it was decided to simply delete all comments (though 'all' is a big word as we've already found a whole

bunch of interesting comments in various places in the code, so some remnants are still left).

D.1.11 What leftovers from CtP1 are still present in the source code? Are the space layer, Fuzzy AI or the UI still there?

Much of that is still unclear at this point, but it was made very clear by Activision that most of the CtP1 features that were actually cut, completely removed from the game. The exception here may be the Fuzzy AI: there still seem to be a lot of references to CtP1-related AI stuff that was no longer supposed to be there in CtP2. How much exactly is left is unclear at this point.

D.1.12 Is it now possible to make a Linux/Mac/other port of CtP2?

Again, it's still too early to tell, but yes, in theory that should be possible. How much work it is, in other words, is it practical enough to actually do it, remains to be seen. However, things are looking good in this department, as the CtP2 code is based on CtP1 code, which was ported to numerous other platforms (even obscure and no longer existing ones such as BeOS). It will require rewriting significant parts of the (most I/O) code though, so don't count on seeing ports appear in the next few weeks. How long that will take will also depends on how much manpower will be available and how much demand for certain ports exists.

Update (2003-11-6): An early start with a Linux port has been made, as you can read here.

Update (2004-04-06): The linux port has been reported to successfully compile (see this post), and there is even a report that someone has managed to get the game working far enough to play a few turns under linux.

D.1.13 Can I use other compilers than Visual Studio 6 to compile the code?

At this point (but it's still early), the answer seems to be no. .NET compilers give a lot more errors than Visual Studio and for now we have no way of working around those other errors yet. For completely different compilers, such as gcc, the necessary makefile is missing, for one thing. However, we hope that we'll eventually get the code working on as many compilers as possible, and if you can help us out with that we would greatly appreciate that.

Update (2004-04-06): There are reports of successful compilation to the point of having a playable game both on VS.NET (here and here) and under linux (here). The changes necessary for this have not, however, moved into the mainstream of the project.

D.1.14 How can I help out with the project?

If you want to help out with the project, we're eternally grateful as we can use all the help we can get. Most notably programmers are needed at this point, but playtesters, artists, webmasters, etc could all come in handy as well. The details of setting up a project team are still being worked out, but you can sign up here. If you're a programmer, for the time being you can just pick your favourite part of the code and start documenting/fixing it. Updated source files for the project can be posted here. Non-programmers are just advised to read the forum and keep up-to-date. They can also help out by playtesting (see here) and translating (see here). A better

organisation for the project will hopefully be set up in the future.

D.1.15 How can I see what has been done so far?

"Official" changes to the source code are posted in this thread - collections of all the changes made are fairly regular, so look near the bottom of the thread if you wish to download a complete update.

If you just want to play with the updated version of the game, playtest versions of the executable are posted in this thread. Bear in mind that, while it should be fairly playable, it's intended for playtesting only: there may be serious bugs in it and it might theoretically break your game (or worse). It is strongly recommended you backup any data before installing. Use at your own risk.

D.1.16 When will you be releasing a 'stable' (non-playtest) version?

We don't know. We have a rough plan of what we wish to achieve here, so watch that space.

D.2 Licence FAQ

D.2.1 Is the CtP2 source code Open Source?

No, it is not. The End User License Agreement (which can be found here) has a number of restrictions on it that prevent it from being called 'open source' software by the common and most widely accepted definitions of that term. Most importantly, for an open source product, the distribution of the code should be entirely free, while the the EULA of the CtP2 source code doesn't allow distribution without written permission from Activision, or any kind of commericial distribution. You can find the industry standard for an Open Source Definition on the Open Source Initiative (OSI) website.

Note that the text of the EULA and Activision's practical interpretation of it seem to be rather different, the interpretation Activision chooses (as became clear from contact with them) seems to be far less restrictive than the official EULA. Of course, in case of doubt it is advisable to refer to the EULA for reference, as "Activision's interpretation", though useful for daily affairs, offers little to no legal basis. The rest of this FAQ will where possible explain both the official EULA and Activision's interpretation of it (or as what I perceive to be their interpretation – to make things even more ambiguous).

D.2.2 Am I allowed to distribute the source?

The EULA forbids any kind of distribution of the original source code without prior written consent by Activision. Also, under no conditions whatsoever are you allowed to in any way distribute the source commercially.

After communicating about this with them, Activision said their interpretation is that it's okay to distribute the source, as long as you make sure that anyone who has access to it must agree to the EULA first. For the standard exe installer this applies, as the installer itself has an EULA screen that you must agree to. If (parts of) the source is (are) distributed in any other format than with the original exe installer (e.g. in zip format), some kind of other mechanism must be in place to ensure the user agrees to the EULA before he can gain access to the code. For commercial distribution of code, one should contact Activision to obtain prior written permisison (contact info is in the EULA). All of this applies to both the original source code and any modified source code.

D.2.3 Am I allowed to distribute modified versions of the game?

Note: this question is about playable, compiled versions of the game. For uncompiled code, see the previous issue.

According to the EULA, you are indeed allowed to distribute new versions, as long as you do not distribute them commercially, label them clearly as a non-Activision product, don't include illegal/obscene/privacy-sensitive infrormation and as long as they require the retail product to function.

Activision says that distribution of new updates are okay, as long as they require the original game to function and all other EULA conditions apply (e.g. regarding labeling the opening

screen, illegal/private content, being free of charge, etc).

D.2.4 Am I allowed to use a CVS server to develop the game?

The EULA forbids keeping copies of the source code for reasons other than backup. Using a CVS server would require you to keep a copy of the code for another reason (i.e. to coordinate the work of several people), so is therefore strictly speaking not allowed.

Activision's interpretation is that the use of CVS servers and similar tools is allowed, as long as anyone who has access to the code has agreed to the EULA.

D.2.5 Is it allowed to port the game to other platforms (Linux, Mac, Amiga)?

The EULA isn't very specific in this regard, but seems to allow it as long as you make sure the retail product is still required to play the ported version.

Activision's interpretation is that it is certainly allowed, as long as you make sure the retail product is still required to play the ported version.

D.2.6 How can I legally make files available for download in this forum?

For your own creations you can just post them as you would with any files. For the CtP2 source code and materials originating from it, there are basically two situations:

1. Files that contains (portions of) the source code. Only people who've agreed to the EULA are allowed to have access to these. To make sure of that, we created a little script: if you want to make any source code files available for download, make sure the URL has the format http://apolyton.net/csd.php?{url}, where {url} is the actual URL address of the file (Example: http://apolyton.net/csd.php?http://apolyton.net/ctp2/files/CTP2_Source.exe – right-click and select properties to see the URL; left-click the link to see the effect). This will force anyone who wants to download your source code files to agree to the EULA first. (Note: disable 'automatically parse URLs' in the reply screen if you're having difficulties getting the link right.)

2. Actual executables to run the game. You can just upload these like other files, but these do require the warning at start-up and in the documentation that they're not Activision material and author & email info, as the EULA specifies. At least for the Apolyton project you can simly use 'Apolyton' or 'Apolyton CtP2 Source Code Project' or similar as author, and for email address you can use ctp2source@apolyton.net (we created that address for this purpose). You can either use an message box to display this error, or replace upsg001.tga with a modified version (or use yet another solution). See here for two practical implementatons.