CC2Spriter – Manual
Version: v2.9.4
Date: Feb. 14th, 2004
Last revision: Mar. 29th, 2010
Author: Manfred "Mafi" Fischer

"Close Combat - A Bridge Too Far" (abreviated CC2, ABTF, CC2-ABTF) was the second game of the CloseCombat-series created by Atomic and presented by Microsoft to the Mac-community. It was also the last game of this series for the MacOS. The series was then continued by SSI/UbiSoft/Mattel Interactive/Destineer/CSO Simtek/Matrix Games for PCs only (up to day CC3, CC4, CC5, CCM, Road to Baghdad, CC:CoI, CCMT, CCM6, CC:WaR, CC:TLD). CC2-ABTF was released in 1997 on a hybrid-CD, running on PCs and under the MacOS 7.5 up to 9.2.2 / MacOS X as well. Later (localized) releases of CC2 were for PCs only.

Purpose of this tool:

- creating CC sprite files from scratch for new games in development,
- extracting / rebuilding all the CC2 sprite files (header ID (first four bytes) "SPRI"),
- extracting / rebuilding the Soldier/SoldierB files of CC2/3/4/5/M/RtB,
- extracting / rebuilding the CC3's Terrain file (header ID "SPRI"),
- extracting / rebuilding CC3's vehicle shadow files (*.nsd / *.zsd, header ID "SPRI"),
- extracting / rebuilding the CC4/5/M/RtB/WaR/TLD Terrain files (header ID "IRPS"), TerrainA.stm (header ID "ALPH") and CC4-RtB-WaR/TLD-vehicle-shadow *.spr files (header ID "IRPS" with varying version-ID),
- extracting the main graphical contents out of the CC3/.../RtB files stored in *.ZFX and *.AZP archives (includes interpreting the file "FXData.fx", this portion of the source code is copyrighted material by Cpl_Filth, used with his explicit permission),
- extracting / rebuilding CC3's ZFX files,
- analyzing CC1 sprites and tiles and extracting them to 16-bit graphics.

Credits: I have to thank Pekka Saastamoinen aka "Cpl_Filth" for the source code of his CCFx.exe and much more and his advice during the development of this tool. Without his help, this project would have failed completely. And I have to thank Gerry Shaw aka "Tin Tin", Shorbun, Zonbie and Cpl_Filth again for the file format informations. Thanks to Nembo for bug reports on version v2.5 and to David Sears for bug reports on v2.8.

This tool is published for MacOS and PCs as well. The resulting files created by the different executables of this tool are identical.

Known problems:

- there are some graphical bugs inside the original "Terrain" files: in the CC2 file the sprite #61 has 8 bytes too much, CC3's Terrain file contains 206 bytes too much, and I suppose that the CC4/CC5/CCM/RtB "Terrain" also has a minor graphical bug inside. If you extract the contents of the CC4/CC5/CCM/RtB "Terrain" file and rebuild it with my tool, you will get a 10 bytes larger file, but it will work correctly with the original program.
- CC4 and newer versions do not use any more the Mac-like "Motorola" byte order "Big Endian" but the PC-like "Intel" byte order "Little Endian" for the "Terrain" file. This will lead in trouble if you use a CC2/CC3 "Terrrain" file footer-datas directly in newer versions.

My tool cannot convert between these two byte orders! It will rebuild the file always in the byte order of the source file.

- My tool will not control the logical integrity of the datas during rebuilding.
- My tool can add sprite animation sequences to the footer-data **(new since v2.9.1)**. Doing this will create a complete new version of sprite/terrain file which the existing games / tools and this tool might not handle properly.
- My tool will treat all CC2 vehicle sprite files containing 33 or more pictures as "turreted vehicle sprites".
- This tool cannot repack CC1-sprite files.
- CC4 introduces *.spr files to store vehicle shadows. In some cases the "color-depth ?" value is varying in these files between 63 and 64. During repacking this tool will always set this value to 63.
- When editing the 'Appendix.bin' file and you change the byte order encoding, the tool will disable the "Save" button. In this case a "Save as..." is possible and recommended.
- Analyzing of sprite files containing too large images might fail, perhaps turning the tool into an endless loop. Extracting will work anyway, as long as you press the correct button (that means: do not extract CC1-files with the CC2-CC5-routine and vice versa) **(new since v2.9.4)**.

Definitions:

- graphical artwork of the CC series is stored in files of different formats.
- These formats vary between the different CC versions, but from CC4 on and newer most of the file formats have not changed any more.
- Some of this files contain only one single graphic, most of these files are called "texture files" (for example some map graphics).
- The other part are graphical archives storing many graphics, sometimes together with a directory list or other informations.
- CC1 (!) uses for storing multiple graphics in a single file the "sprite file" format, indicated by the first four bytes "SPRI". CC1 stores the pixel datas in a different format than all other CC games: 8-bit related to a color look-up table. CC2 or newer use 16-bit for each pixel.
- The CC2 file format survived during the entire series until now (in CC4 and newer the byte order changed and each sprite pixel data entry is terminated by an additional trailing zero byte).
- A "16-bit sprite file" contains always 5 parts, I will refer to this parts as **"Sections"**:
  - Header (8 bytes)
    - Header-ID (4 bytes): "SPRI" or "IRPS"
    - Version-ID (4 bytes): always 00000001hex in CC2 and Terrain-files. *.spr files can have here a 11 different values: 00000001h, AE7800h, AE7A00h, AE8400h, AE9E00h, EE7A00h (airplanes in CC4/CC5), FE9700h, FEA600h, FEBD00h, 12E7800h, 12E7A00h. Meaning of these values is still unknown.
  - Directory (10 bytes)
    - Directory-ID (2 bytes): 03E8hex = 1000 (decimal)
    - Number of sprite graphics in the "Sprite section" (2 bytes)
    - Number of animation sequences in the "Static animation section" (2 bytes)
    - Number of animation sequences in the "Direction-oriented animation section" (2 bytes)
    - 2 bytes of unknown purpose. Cpl_Filth supposes that it might be a color depth definition, but it might be also the maximum length of direction-based animation sequences.

  - Sprite section (varying)
    - SpriteSection-ID (2 bytes): 03E9hex = 1001 (decimal)
    - Sprite entries
  - Static animation sequences section (varying)
    - StaticAnimationSection-ID (2 bytes): 03EAhex = 1002 (decimal)
    - Sprite sequence entries
  - Direction–oriented roto-sprites section (varying)
    - DirectionOrientedAnimationSection-ID (2 bytes): 03EBhex = 1003 (decimal)
    - Sprite sequence entries
- The format of the "sprite entries" differs between CC1, CC2/CC3 and CC4/CC5/CCM/RtB. But the principle is always the same: each sprite entry contains of three parts: the header (describing the image size and hotspots), the line offset table and the pixel datas in 16-bit color (since CC2. CC1 stores them in 8-bit).
- The format of the "sprite sequence entries" is common to all versions.
- "Static animation" sequences have the format:
  - number of entries in sequence list (2 bytes)
  - style indicator (of unknown purpose to me) (2 bytes). In case of the *.spr files these bytes are always filled with FFF2h.
  - 4 zero bytes of unknown purpose. In case of the *.spr files these bytes are always filled with F51Ch and 0068h.
  - sequence list (2 bytes for each sprite entry) of varying length; the sprite numbers are counted from 0.
- "Direction-oriented roto-sprites" sequences have the format:
  - number of entries in sequence list (2 bytes)
  - 4 zero bytes of unknown purpose. In case of the *.spr files these bytes are always filled with FFF2h and F51Ch.
  - sequence list (2 bytes for each sprite entry) of varying length; the sprite numbers are counted from 0.
- Cpl_Filth refered to the datas behind the "sprite section" as **"the footer"**. His PC-tool "SprTool.exe" generates always the original "footer" when rebuilding a sprite file.
- My tool extracts the "footer" (that means the "Static animation section" and the "Direction–oriented roto-sprites section") in a seperate file named "Appendix.bin" without checking it for logical integrity. You can edit this file "Appendix.bin" with my tool and it will be integrated in the sprite file during rebuilding. Sorry to say that "Appendix.bin" files created from *.spr files can not be edited due to a compiler-related list-box limitation. But CC4-RtB are expecting always the same "footer" in *.spr files, no matter if the vehicle has a turret or not (or is an airplane). I think editing is in this case not necessary.
- For the *.spr files with their different version-Ids my tool will create a file "VersionID.bin" containing 4 bytes (the version-ID), which will be read when the sprite file is rebuilded. If this file is not present, the tool will use the value "1" as version-ID.
- Any program of the CC series will abort if it encounters a sprite file containing a "footer" with a different number of animation sequences, but you can add freely sprites to the "sprite section" and sprite numbers to the sequence lists in the "footer".
- CC programs do not refer to all sprites in the "Terrain" file via an animation sequence in the "footer", for example the crosshairs in this file have no corresponding entry in any animation sequence.
- Sorry to say that only CC2 displays animated objects out of the "sprite files". Look at the VL-flags of CC3 and newer: despite they are defined as a 20 entries long animation sequence in the file "Terrain", they are not animated during play at all, only the first sprite defined in the animation sequence is displayed.

- But in CC2 you can have animated VL-flags and animated trees. The fact that the trees can be animated is new to us.
- In CC2 you can have three different animation styles for the static animation sequences: 1100h = play animation once, 1200h = loop eternally, 1300h = loop back and forth eternally.

Using the program:

The main window of this tool is devided into four parts using the tab-panel technique: one part contains buttons for analyzing files, the next part the extracting of "16-bit sprite files", the third part the rebuilding of "16-bit sprite files" and the last part the extracting/rebuilding of "32-bit sprite files" (TerrainA.stm files). The program can determine the byte order of the "sprite file" and will handle it correctly.

Extracting sprites:

Extracting datas out of a "16-bit sprite file" will store the graphical contents in uncompressed 16-bit TARGA graphic files in a seperate folder. The names of the resulting files will be "Image0001.X.Y.tga" where X stands for the hotpoint in horizontal direction and Y for the hotpoint in vertical direction. The hotpoint indicates the pixel of the sprite graphic which is used as the "center" when displaying it during game play. I have added radio buttons for selecting to have the image filenames counted from 0 (compatible to the internal logic of the game) or from 1 (compatible to Cpl_Filth's "SprTool.exe"). Extracting datas out of a "8-bit CC1 sprite file" will store the graphical contents translated into uncompressed 16-bit TARGA graphic files in a seperate folder, too (using an internal picture as a color look-up table). "8-bit CC1 sprite file" will not be extracted as 8-bit graphics, and my tool cannot repack such CC1 files.

The tool will store all datas of the "footer" unchanged in a seperate file "Appendix.bin". An additional "footer report" file will be produced, compatible to MS-Excel (plain ASCII, TAB-seperated columns, CR-terminated lines). This "footer report" file is only for your convenience when checking the animation sequences. It is not used during rebuilding.

There is a seperate button present which will bring a window to front where you can select the special colors used during extraction. The sprite entries contain not only pixel datas but also special definitions for transparency areas, shadow areas, color filling areas and a 2nd shadow area (and more special level definitions in the files Soldier/SoldierB). As suggested by Cpl_Filth and used by his tool "SprTool.exe" the transparency area color is set to white and the shadow color is set to a special light green. The purpose of the color filling area is: it is used to indicate that CC must not produce a color when displaying the sprite. Example: the crosshairs: the area between the two black circles is filled with a varying color during play depending upon the "reachability" of the selected target. Inside the Soldier/SoldierB files the color filling area are the sprite surrounding outline pixels which will be filled by the selection-indicating color when the soldier is selected. The 2nd shadow area are the part of the sprite surrounding outline pixels which will be filled with shadow if the soldier is not selected.

And you can edit the "Appendix.bin" file. When saved, an automatic backup of the previous version of the file will be done. **Remember: the values entered in the sprite sequences are always counted from 0.** If you enter values of sprites which are not defined in the sprite table the CC game will crash at runtime.

Editing 'Appendix.bin':

The datas of the "footer" saved into the seperate file "Appendix.bin" can be edited with my tool. The editing window can be called via menubar or via a push-button in the subsection "Extract". This window gives you also the possiblity to create a complete new file "Appendix.bin" from scratch (since v2.9.1). The upper part of the window let you edit the static animation sequences, the lower part is for editing the direction-based animation sequences. For each area there are buttons to add entries (adding a line, above or below the selected line), deleting an entry and for renumbering the entries. The numbering of the entries has no effect upon the later on saved datas and is simply a counting. You can toggle the editing possibility for special columns (which are partly of unknown purpose to me) for each area seperately. In the lower part of this window there are the buttons for loading and saving a file. If no file is loaded, you can only use "Save as...". In this case you must define which byte order encoding you want to use (Big Endian for CC1-Mac-, CC2- and some CC3-files, Little Endian for all other files). If you change the byte order via the popupmenu (in the lower-right corner of this window) of a loaded file, the "Save" button will be disabled to avoid accidentally erasure of different byte order versions. When saving a loaded "Appendix.bin" file, the program can create up to 99 backup files. Undo for your editing actions is only possible globally to all changes (recreating the status of the loaded file as it is on disk) in both areas (use the button "Undo all..."). Undo is also possible once you have deleted a single entry via the menubar "Edit>Undo".

Usefull situations for editing the "Appendix.bin" file

- adding sprites rather than changing existing ones will force you to change the references to them,
- extending the length of existing animations (flag waving, explosion duration),
- for Soldier/SoldierB files: adding direction-based animations to give way for different sprites for soldier-weapon types which are currently sharing the same animations. In this case you must also modify the static animations (which point to these directions-based animations) but an extension of the static animations is prohibted there unless you change the EXE-code and the base-data file "SolActn".

Extracting ZFX:

For Mac-users I have integrated the extraction of datas out of *.ZFX archives (CC3) and *.AZP archives (CC4 and newer) and the extraction of graphics out of the resulting *.fx files (CC3 and newer) and *.tex files (CC4 and newer). PC-users please use the already existing tools available on the internet. CC3's *.ZFX archives can not only be extracted, but also rebuilded. These *.ZFX files have a fixed size header (number of files, filename table with 600 entries, offset table with 600 entries, file size table with 600 entries) followed by the file datas. **The extracting process will respect the values entered in the file size table. The only exception are zero values. In this case my tool will calculate the file size from the file offset distances.** The tool will report such errors in the ZFX directory table. Zero values in the file size table are always present if the ZFX archive was made with older tools. Such zero values are not used in the original CC3 files, but zero values here will not prevent CC3 from handling the ZFX archive correctly. When repacking the ZFX archives my tool will only import readable and visible files if their filename does not start with a dot ".". For full extract/repack of *.AZP and *.TEX files please use my tool RtBTool v2.1.

Rebuilding sprites:

Rebuilding of the sprite files can be done in the third tab-panel of the main window. You must select a folder containing the datas. All TARGA graphics must be in uncompressed 16-bit. The tool will take them in the sequence as they are numbered from "Image0000.X.Y.tga" onward. X stands for the hotpoint in horizontal direction and Y for the hotpoint in vertical direction. The last file to be added will be the "Appendix.bin" file. Valid filenames for import must start with "Image" and must end with ".tga". All other files in the selected folder will be ignored by the tool. The tool can skip gaps in the filename numbering. Since v2.0 the tool will handle the TARGA-header informations correctly. But if you want to have the same behavior as Cpl_Filth's tools you can determine that the TARGA-header informations will be ignored. Pixel data transfer will be then done in the sequence as they are stored in the TARGA data area.

If the file "Appendix.bin" is missing, the tool can only add a "footer" from scratch for the CC2/CC3 vehicle shadow files ("VehB###", "VehS###", "*.nsd" and "*.zsd" files). Turretless CC2 vehicles requires 32 sprites (for 32 shadow directions) for correct "sprite file" generating. Turreted CC2 vehicles have 32 gun/turret shadows in addition, so in this case the total number of required sprite graphics is 64. CC3 vehicles require always 64 sprites.

When rebuilding the tool will always try to generate the header in the correct format for the given "sprite file" and will adjust the number of sprite graphics according to the number of TARGA files in the folder. So you can expand the number of sprite graphics in the"Sprite section" and you must not (but you can) overwrite existing sprites. **The tool will set always the number of animation sequences to the real numbers of sequences out of the file 'Appendix.bin' (this is new since v2.9.1: the ignoring of the real numbers and automatical setting to the correct values of the given CC version is no longer implemented, but the tool will give you a warning if the number of sequences is different from the expected values)**. If you want to setup a complete new version of "Terrain" file for CC9 or so (with more animation sequences) you must no longer use a hex-editor to correct the header of the produced output file. Such "expanded" files will not be recognized correctly by existing CC games / tools (perhaps ?).

When rebuilding CC4/CC5/CCM/RtB vehicle shadow files ("*.spr" files), it is possible to have different version-IDs. If this version-ID is not equal to 1, then my tool will write the version-ID to an additional file "VersionID.bin". This file will contains always 4 bytes (Little Endian encoded). The presence of such a file will be detected during rebuilding, importing these 4 bytes as version-ID for the new created "*.spr" file.

Be warned: what we have discovered in 2010: the internal line offset table for each sprite image is limited to short integers (16 bit). **So you cannot import images as sprites larger than around 146x146 solid color pixels** unless you have a lot of white lines in the image. Such entire white lines can be compressed (saving pixel datas), and CC2 itself is using images of maximum size 180x180 pixels due to the fact that these images contain a lot of white lines.

STM file format (CC4-CC5-CCM-RtB) – the TerrainA.stm files

This file format is used by all the TerrainA.stm files of the CC versions since CC4.
Whole file is encoded in Little Endian (PC/Intel like). All "sprites" are single graphics encoded in 32-bit (same color encoding as it is used in the *.fx files since CC3).

// header
ALPH // 4 bytes ASCII containing the string “ALPH” = sprites with alpha-channel
long // 4 bytes, containing value “2”
long // 4 bytes, number of sprites in this file

// directory with sprite definition entries
// each entry 20 bytes of the format:
long // 4 bytes, HotSpot X
long // 4 bytes, HotSpot Y
long // 4 bytes, image width
long // 4 bytes, image height
long // 4 bytes, offset of pixel datas (from top of file)

// sprite graphics data
// 4 bytes per pixel: blue, green, red, alpha
// line orientation is top-to-bottom, pixel orientation is left-to-right

```
We must use Cpl_Filths algorithm of making 32-bit TARGAS
after reading the CC values myBlue, myGreen, myRed, myAlpha:
                    if ( ( 8 * myBlue - 1 ) > 0 ) then
                      myBlue = (8 * myBlue) -1
                    end
                    if ( ( 8 * myGreen - 1 ) > 0 ) then
                      myGreen = (8 * myGreen) -1
                    end
                    if ( ( 8 * myRed - 1 ) > 0 ) then
                      myRed = (8 * myRed) -1
                    end
                    if ( myAlpha <> 0 ) then
                      myAlpha = (8 * myAlpha) -1
                    end

To write them to a valid 32-bit color-depth TARGA file, the sequence must be
changed:
                      OutFile.WriteByte(myRed)
                      OutFile.WriteByte(myGreen)
                      OutFile.WriteByte(myBlue)
                      OutFile.WriteByte(myAlpha)
```

The TerrainA.stm file of CC4 contains 74 sprites, all newer TerrainA.stm files contains 80 sprites. The "Analyze" part of the main window of my tool can detect which format is used by the sprite file (16-bit or 32-bit) by checking the first four bytes of the file. Extracting/Repacking is only possible for my tool when using 32-bit TARGA graphics (8-bit for alpha channel) (like Cpl_Filth’s tool "Stm.exe", thanks again Cpl! Without his tool I would be never able to know for what I have to look inside the ALPH). Manipulating 32-bit TARGA files with alpha channel can be done using Photoshop.

ZFX file format (CC3) – the *.zfx archive files

This file format is used by all the *.zfx files of CC3.
Whole file is encoded in Little Endian (PC/Intel like). First description of this file format was made by Gerry Shaw aka "Tin Tin", he wrote: "… the *.zfx file contain a bunch of other files concatenated into a single file. This is done to decrease seek times when reading files from disk or CD." With the help of Zonbie I can give here a complete description of the directory:

```
// header
long     // 4 bytes containing number of files in this ZFX-archive

// directory with 600 file definition entries, fixed size
// 1.: file name table with 600 entries, fixed size of 7800 bytes
// each entry 13 bytes ASCIIZ: 8.3 DOS like filenames (up to 8 chars, dot, 3 chars extension)
//                          terminated by a zero byte.
array(600) of char(13)           // ASCIIZ, padded with zeros to a fixed length of 13 bytes.
// 2.: file offset table with 600 entries, fixed size of 2400 bytes
array(600) of long      // 4 bytes per entry, offsets counted from top of file.
//3.: file size table with 600 entries, fixed size of 2400 bytes
array(600) of long      // 4 bytes per entry.

// file datas to follow, concatenated without gap
data
```

The original ZFX archives are filled with garbage datas in the higher unused areas of the file offset and file size table.

A shadow problem (January 2006)

In early 2006 a problem with CC3 vehicle shadow sprite files was discussed at the CCS.net forum. The task was to give existing vehicles correct shadows for a new sun direction. In original CC the sun is shining from the upper left edge of the map. David Sears aka "GameRat" wanted to change the sun to be shining from the lower left edge of the map (as if the sun is shining from about 8 o'clock). This is the resulting instruction to do this, compiled together by him, Zonbie, Randall and myself:

1. first you must know what kind of sun-direction you need
2. then you must know that the shadow images in the sprite file has to be treaten as a "sequence"
3. you can bundle them together into one single GIF-animation to see what I mean.
4. you will then see the shadow of a "rotating" vehicle!
5. from this point on you will know:
   - rotation direction (clockwise or counterclockwise), I suppose it should be clockwise.
   - number of necessary images for 360° rotation (in CC3: should be 64)
   - you will see where the rotation center is (= hotpoint), this hotpoint must be identical with the numerical hotpoint description when repacking the images together into a CC-sprite file. Please notify: each image CAN have different hotpoint, but in case of SHADOW sequences, these images should be always of same size and have same hotpoint. Please control this by studying original shadow sprites.
6. what you must change is the graphical contents.

7. And I think at this point you are correct: should be sufficient to do this with a single Photoshop action to be applied to all images of this sequence. And I believe that a flipping (if it fits to your needs) will be right.
8. But you must also change the starting image in your sequence: in your case you must use the 32nd image as new starting image.

And this was the solution by David:

9. Used Mafi's CC2Sprite tool to unpack the Shadows.zfx ... then unpacked the nsd file which resulted in 64 tga's. Flipped all images vertically. Files where listed as 0000 to 0063.
10. Figured out a new sequence: starting at 0032 counting down to 0000 then 0063 counting down to 0033. This order from beginning to end was then renamed 0000 to 0064.
11. File 0000 was renamed to 0032 and 0032 to 0000. 0016 and 0048 were both left unchanged other than being flipped vertically.
12. Repacked it all back to Shadows.zfx and tested.

Works great especially with the crew soldiers also having matching shadows. All that needs figured out now is the soldier fire animation problem.

Another problem with the non-editable CC3/4/5/M/RtB vehicle shadow appendix.bin

Vehicle shadow files of CC3/C4/CC5/CCM/RtB/CC6/CC-whatever-since-2005 are always of same identical format:
- 64 pictures,
- only one dynamic animation sequence, containing 64 entries,
- every entry is needed for one of the 64 directions a vehicle can be turned to,
- these 64 entries are referencing usually to one of the pictures,
- usuall sequence is 0,1,2,3,4,...
- no dynamic animation sequence for the cannon of the turret,

so turreted and turret-less vehicles and guns and planes have always identical shadow file format.

CC2Spriter is unable to display all these 64 entries (a limit of the compiler I'm using, sorry). But from my point of view it is not necessary to change this sequence unless you have something very special with only one picture for non-moving static objects. In this case use HexEdit to patch the shadow file manually.

Under normal conditions (yes, I know the one exception, the CC2-88FlaK) it will be sufficient to exchange the pictures. The datas of the appendix.bin can always remain untouched.

CC1 TILES File Formats

CC1 stores it’s map tile graphics in squares of 40x40 pixels bundled together in one file. The file’s name is "TILES" (PC version) or "Terrain Tiles" (MAC version). Both files differ only in their 12 bytes header, the rest of the tile datas are identical:

MAC version:
// header encoded in BIG Endian
TILE // 4 bytes ASCII containing the string “TILE” = 8-bit map tiles
long // 4 bytes, containing value “65536” = 00010000h
long // 4 bytes, number of tiles in this file (usually 1170)

// data, 8-bit encoded pixels, 1600 pixel per tile entry without any header

PC version:
// header encoded in LITTLE Endian
ELIT // 4 bytes ASCII containing the string “ELIT” = 8-bit map tiles
long // 4 bytes, containing value “65536” = 00010000h
long // 4 bytes, number of tiles in this file (usually 1170)

// data, 8-bit encoded pixels, 1600 pixel per tile entry without any header

What is new in v1.1:
- error corrected preventing the tool from analyzing CC4/CC5/RtB Soldier/SoldierB files,
- radio buttons added to select the numbering of filenames when extracting sprite graphics,
- the tool can now skip over gaps in the input file numbering,
- editing of unused columns in the "Appendix.bin" is now possible, but will have (perhaps) no effect during gameplay.

What is new in v1.2:
- debugging (error while saving Appendix.bin file after editing).

What is new in v1.3:
- CCM-compatibility added (CCM was released to the public in August 2004).

What is new in v2.0:
- TerrainA.stm extraction/rebuilding added. TARGA-header interpreting corrected.

What is new in v2.5:
- *.spr extraction/rebuilding added. *.tex file extraction now compatible to my RtBTool.

What is new in v2.7:
- the tool will now respect the sprite entries’ byte-size definition when extracting to TGA. This will avoid trouble with the bug-filled original CC3 Terrain file. Pixels exceeding the sprite’s width or height will be ignored. A report after extraction will be shown in a dialog box (will only show up when extracting original CC2 / CC3 Terrain file).

What is new in v2.8.1:
- *.nsd / *.zsd extraction/rebuilding added.
- *.zfx extraction/rebuilding added.

What is new in v2.8.2:
- *.zfx extraction/rebuilding debugged. CC1 sprite analyzing/extracting added.

What is new in v2.8.3:
- Thanks to Nembo: CC2-88-FlaK-bug removed: the vehicle shadow of the original 88-FlaK contains 33 pictures. The tool will now treat all 33 pictures or more containing CC2 vehicle sprite files as "turreted" vehicle sprite files with 2 direction animation sequences. That will mean on the other hand that heavily compressed CC2 shadow sprite files of turreted vehicles

(for example with only one picture per direction animation sequence) will be perhaps detected and rebuilded as turretless vehicle sprite files.

What is new in v2.8.4:

- CC1 color look-up table debugged.
- CC1 file "TILES" analyzing and extracting added.
- Converting section added: the tool can now convert CC1's sprite and tile files between the PC and MAC version of CC1. In addition the tool is able to convert the Soldier / SoldierB files from CC2/CC3-file format (BIG Endian encoding) to CC4/5/M/RtB-file format (LITTLE Endian) and back. This functionality is valid only for these files!

What is new in v2.9.0:

- Soldier/SoldierB unpacking/repacking added.

What is new in v2.9.1 (2006):

- Editing the "Appendix.bin" extended to enable creating them from scratch including better backup-file handling. This manual extended (see page 4f).

What is new in v2.9.2 (2010):

- Recognition of CC:WaR/TLD Terrain files added.
- Maximum number of editable columns for 'Appendix.bin'-Editor reduced to 63! This means that this tool can only edit 'Appendix.bin' files containing sequences with up to 57 entries, not more. Therefore, it is not suitable to edit 'Appendix.bin'-files of shadow-sprite-files (*.spr).
- 'Appendix.bin'-Editor: sorting by column is possible by pressing into the header, warning: no specific undo possible, but sorting will respect numerical order.
- 'Appendix.bin'-Editor: renumbering will have no effect on saved datas, because these numbers will not be written to file. But a renumbering might be usefull to reorder the list by sorting. And such a new ordering will be the sequence in which the lines will be written back to the file 'Appendix.bin'. Warning: no specific undo possible after renumbering and/or sorting, only undo back to "undo all" = back to situation after loading the file.

What is new in v2.9.3 (2010):

- Debugging the buttons of "Repacking CC:WaR/TLD Terrain file".
- Supporting larger images added on repacking, but this does not make much sense as long as the internal sprite-file line-offset table is limited to 2-bytes values (see text above: "Rebuilding Sprites").

What is new in v2.9.4 (2010):

- Debugging the analyzing and repacking of files with too large images. Analyzing might fail.
- UI of 'Appendix.bin'-Editor improved.

This tool is still under development. All in all you are using this tool at your own risk. The author is not responsible for any damage or loss of data this tool will cause.

Mafi
closecombat2@claranet.de
http://www.closecombat2.claranet.de
http://www.ftf.claranet.de/
http://cc2revival.npage.de/
http://closecombat2.fortunecity.com/
http://www.geocities.com/cc2revival/
http://members.fortunecity.de/closecombat2/