CeX3D Converter 0.4.1 - Commandline Manual

Ánoq of the Sun, Hardcore Processing *

July 17, 2003

1 Introduction

CeX3D Converter can do:

- `LightWave 3D` to `Unreal Editor` conversion.
- `LightWave 3D` to `RenderMan` conversion.
- `LightWave 3D` to `LightWave 3D` conversion.
- `Unreal Editor` to `RenderMan` conversion.
- `Unreal Editor` to `LightWave 3D` conversion.
- `Unreal Editor` to `Unreal Editor` conversion.

It is available for:

- Windows (x86/win32)
- Linux (x86/Linux)

It is a standalone command line utility which can be installed so that objects can be converted with only 2 mouse clicks in Windows Explorer.

See *Appendix A: Features* for details about what is supported.

2 This document in other formats

This document is a available in the following different file formats:

- `http://www.CeX3D.net/converter/documentation/cmdmanual.html`
- `http://www.CeX3D.net/converter/documentation/cmdmanual.pdf`
- `http://www.CeX3D.net/converter/documentation/cmdmanual.ps`

3 Disclaimers and copyrights

- CeX3D Converter is copyright © 2000-2001 by Hardcore Processing.
- NO ONE MAY REDISTRIBUTE CeX3D Converter or any parts of it.
- The customer of CeX3D Converter will indemnify, hold harmless, and defend Hardcore Processing against lawsuits, claims, costs associated with defense or accusations that results from the use of CeX3D Converter.
- Hardcore Processing is not responsible for any damages whatsoever, including but not limited to loss of data, interruption of business, personal injury and/or any consequential damage without limitation, incurred before, during or after the use of CeX3D Converter. Hardcore Processing's entire liability, without exception, is limited to the customer's reimbursement of the purchase price of the software (maximum being the suggested retail price listed by Hardcore Processing) in exchange for a signed paper contract of assurance that the customer have deleted all versions of CeX3D Converter that the customer has ever had anything to do with.

4 Installing

To install CeX3D Converter on your computer you should follow the instructions given in the one of the subsections below which correspond to your system.

4.1 Installing on Windows

1. Unpack the file `CeX3DConverterCmd*_x86-win32.zip`.
2. Copy the file CeXC.exe somewhere on your computer where you have a path to. The directory `C:\Windows\System32` is usually a good place.

The unpacked directory also contains two example batch files which enables you to convert files with only 2 mouse clicks in Windows Explorer. These batchfiles can be installed as follows:

1. Copy the batch files to where you want them to be. This could be `C:\Windows\System32`.
2. Now you could edit the batch files in notepad (or whatever) to suit your needs. But you can also wait until you have read the rest of this manual :) Remember to edit the files in `C:\Windows\System32` or where ever you copied them to.
3. Open Windows Explorer.
4. Choose the `View` menu and choose `Options...`.
5. Go to the `File Types` page.
6. Press `New Type...`.
7. Type *LightWave Object* in the field `Description of type`.
8. Type *.lwo* in the field `Associated extension`.
9. Press `New...` under the `Actions` field.
10. Type *Convert to Unreal* in the field `Action`.
11. If you placed the batch files under `C:\Windows\System32`, then type `C:\Windows\System32\ExampleToUnreal.bat` in the field `Application used to perform action`.
12. Press `Close` in this window.
13. Press `New...` under the `Actions` field again.
14. Type *Convert to RenderMan* in the field `Action`.
15. If you placed the batch files under `C:\Windows\System32`, then type `C:\Windows\System32\ExampleToRenderMan.bat` in the field `Application used to perform action`.
16. Press `Close` in this window.
17. Press `Close` again.
18. And once more... and you're now set.
19. You can now right-click on a *.lwo* file in Windows Explorer and do either *Convert to Unreal* or *Convert to RenderMan*.

4.2 Installing on Linux

1. Unpack the file `CeX3DConverterCmd*_x86-linux.tar.gz`. This is can usually be done with these commands:

   ```
   gzip -d CeX3DConverterCmd*_x86-linux.tar.gz
   tar -xvf CeX3DConverterCmd*_x86-linux.tar
   ```

2. Copy the file CeXC somewhere on your computer where you have a path to. The directory `/usr/bin` is usually a good place. You can copy it with this command:

   ```
   cp CeXC /usr/bin
   ```

5 Using CeX3D Converter

CeX3D Converter is a standalone command line utility. This means that under Windows you use it from DOS and under Linux you use it from a Unix shell.

Under Windows you can also convert files with 2 mouse clicks in Windows Explorer. However, this probably still requires that you edit a batchfile to do the conversion the way that you need it.

The following will describe the commandline options for the converter.

5.1 Basic use of CeX3D Converter

CeX3D Converter can be called with the command CeXC like this:

```
CeXC --to=<output format> MyObject.lwo
```

Where <output format> is one of:

- RIB : RenderMan RIB format
- Unreal.T3D : Unreal brush files

So for instance you can type:

```
CeXC --to=Unreal.T3D MyObject.lwo
```

The default for <output format> is RIB so if you just write

```
CeXC MyObject.lwo
```

then CeX3D Converter will convert to RenderMan RIB format. You can convert several files at once by writing a command like this:

```
CeXC MyObject1.lwo MyObject2.Lwo MyObject3.Lwo
```

CeX3D Converter has many options for controlling the conversion process. The options available can be shown if you run the converter without any arguments:

```
CeXC
```

The other options will also be documented in the following sections.

5.2 Tutorial for convertering from LightWave on Windows to RenderMan on Windows

If you are converting `LightWave 3D` objects made on Windows into `RenderMan` RIB files and RenderMan shaders to be rendered with Larry Gritz's `Blue Moon Rendering Tools` (BMRT) on Windows, you would problably want to use a command similar to this:

```
CeXC --fromTextureRoot=C:\MyLWTextures\
  --toTextureRoot=C:\MyBMRTTextures\
  --textureExtension=TIFF
  MyObject.Lwo
```

If you are using `BMRT` then you would probably want to know that BMRT only supports 24bit TIFF images. So you will have to convert and rename your texture files to the appropriate 24bit TIFF files.

To render the exported files with BMRT you need to set up your BMRT paths correctly for shaders and all. You also need to compile the generated shaders with the BMRT command `slc`. Then you can render with the BMRT command `rendrib` - but see the BMRT documentation about these things.

Unfortunately I do not have any experience with Pixar's Photorealistic RenderMan (PRMan) on these matters.

If you need the details of why the CeXC commands looks as described above - then they are documented after these tutorials.

5.3 Tutorial for convertering from LightWave on Windows to RenderMan on Unix

If you are converting `LightWave 3D` objects made on Windows where your textures are stored in `C:\MyLWTextures\` and you want to convert into `RenderMan` RIB files and RenderMan shaders to be rendered with Larry Gritz's `Blue Moon Rendering Tools` (BMRT) on Unix where your textures are stored in `/usr/local/myBMRTTextures/`, you would problably want to use a command similar to this:

```
CeXC --fromTextureRoot=/C/MyLWTextures/
  --toTextureRoot=/usr/local/myBMRTTextures/
  --toOSType=Unix --textureExtension=TIFF
  MyObject.Lwo
```

If you are converting `LightWave 3D` objects made on Unix where your textures are stored in `/usr/local/myLWTextures/` and you want to convert into `RenderMan` RIB files and RenderMan shaders to be rendered with Larry Gritz's `Blue Moon Rendering Tools` (BMRT) on Windows where your textures are stored in `C:\MyLWTextures\`, you would problably want to use a command similar to this if you're using CeX3D Converter for Unix:

```
CeXC --fromTextureRoot=\\usr\\local\\myLWTextures\\
  --toTextureRoot=C:\\MyLWTextures\\
  --toOSType=Windows --textureExtension=TIFF
  MyObject.Lwo
```

If you're running CeX3D Converter on Windows, you should remove the double backslashes in the Windows filepaths `C:\\MyLWTextures\\` and `\\usr\\local\\myLWTextures\\`.

You will still need to convert textures and compile RenderMan shaders as described in the previous tutorial.

5.4 Tutorial for converting from LightWave to UnrealEd

If you are converting `LightWave 3D` objects into Unreal.T3D files then you would problably do something like this:

```
CeXC --to=Unreal.T3D --textureWidth=256 --textureHeight=256
     --coordTransformsLH=rz90,ry90 MyObject.Lwo
```

After this is done you will want to import this correctly into the `Unreal Editor`. This is done with the following steps:

1. Convert all textures used in the LightWave object into 24bit BMP files (24bit PCX files did not seem to work). The textures must have the same resolution as you specified during conversion - which is 256x256 pixels in this example.
2. Copy the resulting texture files into the texture directory in the Unreal Editor. This directory is usually `UnrealTournament\Textures\`.
3. Start the Unreal Editor.
4. In the lower right corner of the screen, press *import* for importing a texture file.
5. In the dialogbox which appears, things should work if you type the name of the texture (without filepath or .BMP or .PCX) in the field *Name*, type *None* in the *Group* field and *MyTextures* in the field *Package*.
6. Then confirm this dialogbox.
7. Repeat steps 4 to 6 for all textures used in the LightWave object.
8. In the menu at the top of the screen go to the brushes menu and choose *import brush* (not load brush).
9. Find the .t3d file you exported with the CeXC command just before (but be sure that you're not loading it over a network, since UnrealEd cannot load files over a network).
10. In the dialogbox which appears after this, you can choose how the Unreal Editor should handle the object.
11. Confirm this dialogbox.
12. Now you have to click on one of the 3D views before you will see the imported brush.
13. Place the brush where you want it in the Unreal world.
14. Finally you can go to the brush menu at the top of the screen and choose *Subtract* to subtract the brush from the Unreal world - in case you want to carve the LightWave object out of the Unreal world and see the inside of the object.

15. Or - if you have already carved an environment into the Unreal world which is big enough to contain your LightWave object, then you can go to the brush menu at the top of the screen and choose *Add* to add the brush to the Unreal world. This means that you can see the outside of the object in the Unreal editor.

For more details about the Unreal Editor you should look for the Unreal Editor documentation.

6 CeX3D Converter options reference

The following is a complete reference to all of CeX3D Converter's options.

6.1 Overview

6.1.1 Options for renaming texture filenames

- `--fromTextureRoot`
- `--toTextureRoot`
- `--textureExtension`
- `--toOSType`

6.1.2 Unreal.T3D specific options

- `--textureWidth`
- `--textureHeight`

6.1.3 Coordinate transformation options

- `--scale`
- `--coordTransformsLH`

6.1.4 Other options

- `--destFileName`

6.2 Option reference

6.2.1 –fromTextureRoot

To change the root directory where your texture files are placed during the conversion you can use the options `--fromTextureRoot` and `--toTextureRoot`.

The option `--fromTextureRoot` specifies the prefix of the path to be removed from the texture filenames:

```
--fromTextureRoot=<old texture root>
```

Notice that if you need to use backslashes in the texture root you must type 2 backslashes for each backslash in the filename if you're running CeX3D Converter for Unix. You should not do this if you're using CeX3D Converter for Windows.

As an example of this you could change the texture directory from `C:\Textures\` to `T:\` with the following:

```
CeXC --fromTextureRoot=C:\Textures\ --toTextureRoot=T:\ MyObject.Lwo
```

Again, remember to use double backslashes if you're using CeX3D Converter for Linux.

6.2.2 –toTextureRoot

To change the root directory where your texture files are placed during the conversion you can use the options `--fromTextureRoot` and `--toTextureRoot`.

The option `--toTextureRoot` specifies the new texture root directory:

```
--toTextureRoot=<new texture root>
```

If you need to use backslashes in the texture root you must type 2 backslashes for each backslash in the filename when using CeX3D Converter for Unix.

As an example of this you could change the texture directory from `C:\Textures\` to `T:\` with the following:

```
CeXC --fromTextureRoot=C:\Textures\ --toTextureRoot=T:\ MyObject.Lwo
```

Again, remember to use double backslashes if you're using CeX3D Converter for Linux.

6.2.3 –textureExtension

If you set the following option:

```
--textureExtension=<extension>
```

then all texture filenames will have their old file extension removed and the specified <extension> will be used instead. For example if you specify

```
--textureExtension=TIFF
```

and a texture is called Texture.JPEG then it will be renamed to Texture.TIFF.

6.2.4 –toOSType

Setting the option:

```
--toOSType=Unix
```

will convert all texture filenames to Unix filenames. For instance backslashes in Windows filenames will become slashes and volume names in windows like `C:\` will become directories - in this example `/C/`. This is handy when converting objects from Windows to Unix.

Setting the option:

```
--toOSType=Windows
```

will convert all texture filenames to Windows filenames. For instance slashes in Unix filenames will become backslashes. This is handy when converting objects from Unix to Windows.

The default value for `--toOSType` is `Windows` if you are using CeX3D Converter for Windows. The default is `Unix` if you are using CeX3D Converter for Linux.

6.2.5 –textureWidth and –textureHeight

If you are converting to Unreal.T3D format then you will most likely want to specify the resolution of the textures you are using. This can be done with the following options:

```
--textureWidth=256 --textureHeight=256
```

In this case the textures used in the object must have the resolution 256x256 pixels in the Unreal Editor. The default value for `--textureWidth` and `--textureHeight` is 512, so if you don't set the `--textureWidth` and `--textureHeight` option, your textures must be 512x512 pixels.

6.2.6 –scale

If you wish to scale objects during conversion it can be done by setting the option:

```
--scale=<factor>
```

For instance if you wish to scale an object by a factor of 2.5 it is done like this:

```
--scale=2.5
```

The default scaling factor when converting to RenderMan is 1. When converting to Unreal the default scaling factor is 40.

The reason for setting the Unreal scale factor to 40 as default is that one unit in `LightWave` is 1 meter. One unit in `Unreal` on the other hand, is about one inch.

6.2.7 –coordTransformsLH

When converting objects it is possible to rotate them. This can be done with the option:

```
--coordTransformsLH=<transforms>
```

Here <transforms> is a comma separated list of transformation operations. Each transformation operation can be one of:

- rx<angle> - for rotating <angle> degrees around the X axis.
- ry<angle> - for rotating <angle> degrees around the Y axis.
- rz<angle> - for rotating <angle> degrees around the Z axis.

For example if you want first to rotate 12.5 degrees around the X axis and then 45 degrees around the Y axis you can write:

```
--coordTransformsLH=rx12.5,ry45
```

When converting from LightWave to the Unreal Editor you will probably want to use:

```
--coordTransformsLH=rz90,ry90
```

This is because in LightWave (and in RenderMan) the coordinate system looks like this:

- Positive direction of X axis points to the right.
- Positive direction of Y axis points up.
- Positive direction of Z axis points away from the camera.

In the Unreal Editor however, it looks like this:

- Positive direction of Y axis points to the right.
- Positive direction of Z axis points up.
- Positive direction of X axis points away from the camera.

So rotating first 90 degrees around the Z axis and then 90 degrees around the Y axis will transform objects so that *up* in LightWave becomes *up* in the Unreal Editor, *away from the camera* in LightWave becomes *away from the camera* in the Unreal Editor and so forth. It should also be noted that when rotating some multiple of 90 degrees the rotations will be completely accurate - this is not always the case when working in LightWave, so you are encouraged to use CeX3D Converter to handle this :)

You can also consider using the option:

```
--coordTransformsLH=rx90
```

This will at least convert *up* in LightWave to *up* in the Unreal Editor.

The attentive reader will already have noticed that both of the above coordinate systems are lefthanded coordinate systems. This is the reason why the option is called `--coordTransformsLH` and not just `--coordTransforms`. When `--coordTransformsLH` is used, the objects are simply kept in the lefthanded coordinate systems - and this is also what makes most sense, since all supported formats use lefthanded coordinatesystems. However it should be noted that mathematical standard coordinate systems are right handed coordinate systems, so this is what CeX3D Converter uses internally - but the option `coordTransformsLH` will not disappear in the future.

6.2.8 –destFileName

When converting files, the filename of each converted file is automatically generated from the input filename. For instance if you convert an object called MyObject.Lwo into Unreal.T3D format, then the destination file will be called MyObject.t3d. If you don't want your file to have these default names you can specify your own destination filename with this option:

```
--destFileName=<filename>
```

7 Appendix A: Features

7.1 Supported input formats

- `LightWave 3D` Object files from version 6 and above (LWO2).
- `LightWave 3D` Object files before version 6 (LWOB).
- `Unreal` brush (Unreal.T3D) files for the Unreal Editor.

7.2 Supported output formats

- `RenderMan` RIB files with RenderMan shaders.
- `Unreal` brush (Unreal.T3D) files for the Unreal Editor.
- `LightWave 3D` Object files from version 6 and above (LWO2).

7.3 General features

- CeX3D Converter for Windows comes with example batch files for converting objects with only 2 mouse clicks in Windows Explorer.
- Input fileformats are automatically detected.
- Objects can be scaled during conversion.
- Objects can be rotated during conversion - with all multiples of 90 degrees being completely accurate.
- Texture paths can be changed during conversion.
- Support for arbitrary polygons.
- Support for planar mapped textures.
- Support for UV textures.

7.4 Features for each format

	read LWOB	read LWO2	read Unreal.T3D	write RIB	write Unreal.T3D
Triangles	Yes	Yes	Yes	Yes	Yes
Arbitrary polygons	Yes	Yes	Yes	Yes	Yes
Planar mapped textures	Yes	Yes	No	Yes	One per polygon
UV textures	No	Yes	No	Yes	One per polygon
Other surface data	Yes	Yes	No	Yes	No
Additive surfaces	No	No	No	No	No
Reflections	Yes	Yes	No	No	No
Refractions	No	No	No	No	No
Caustics	No	No	No	No	No

Notice that a feature has to be supported for both the file format you are reading and the file format you are writing before it will actually work.

7.4.1 Limitations in the formats

The Unreal.T3D file format has limited capabilities for texture support. It only supports one color texture per polygon. So if there are multiple textures on a polygon, only the first color texture will be exported in the Unreal.T3D file. However, you can create several LightWave surfaces for different polygons, each with different texture. The Unreal Editor only supports texturing of up to 3 accurate UV texture coordinates - so when working with UV textures you should probably triangulate your objects. The Unreal.T3D file format also only seems to be capable of handling polygons with up to 16 vertices each and only polygon meshes containing up to 500 polygons in total. CeX3D Converter will try to split polygons with more than 16 vertices - but this will of course generate additional polygons, so you may want to split these polygons manually.

8 Appendix B: Planned features

The following features are currently planned to be implemented in a near future:

1. Split LightWave objects based on layers or surfaces.
2. Support for importing LightWave 3D Scene files.
3. Support for exporting Unreal levels.
4. Support for the rest of the surface attributes when converting to RenderMan RIB and Shading Language files.
5. Support for LightWave's fractal noise textures when exporting to RenderMan.

9 Credits

Credits in no particular order goes to:

- The people at ION Storm, in particular Clay Hoffman, Robert Kovach and Peter Marquardt - for using CeX3D Converter on a real production, doing lots of testing, giving lots of ideas and feedback and for being very enthusiastic.
- Erik De Neve from Epic Games - for helping with all my questions.
- Larry Gritz from Pixar - for help and for tolerating my unjustified claims of bugs in BMRT ;)
- Brad Peebler from NewTek - for his help, enthusiasm and interest in CeX3D Converter.
- Virtual Effects and Fantasies - for betatesting CeX3D Converter on a real production.
- Josh Tsui - for feedback and for supplying test material.