Compile Bforartists for Windows with Cmake and Visual Studio 2019

Table of content

Overview....2
- Introduction....2
- Outline....2

Tool Requirements....2
- Required Tools....2
  - Building Tools....2
  - Sourcing Tools....2
  - Other Tools....3
- Optional Tools....3

Downloading the Repository....3
Creating the Binary Folder....5
Installing the Dependencies....5
Updating the Repository....6
Compiling, quick way, console....6
Compiling, long way, Cmake....6
- Cmake Preparation....6
- Activate all needed features....8
- Generate....9

Troubleshooting Cmake Errors....9
- Cuda....10
- Cmake Warning at intern/cycles/kernel/CmakeLists....10
- Adding System variables....11
- Missing Language Paths in Blender....11
- Open EXR....12

Compiling with Visual Studio....13
- Congratulations!....14

Overview

Introduction

This is a step-by-step tutorial packed with tips and tricks to help you compile your first Bforartists build – opening doors to develop your artistic ideas with the source code.

Outline

To compile, we first need to gather the correct tools, materials, and blueprints, and then we can build the "house" - which is **Bforartists**. The following tutorial will explain each process step by step with tips and tricks to make sure you can troubleshoot and build without any errors.

Once you have no errors in the building process, you will be free to work in a Github branch and commit and push new improvements to Bforartists – with set tasks that can be worked on collaboratively and approved to be included in future official releases.

So let's begin by getting the right tools, then the correct materials and blueprints – then we can use the tools to build our house.

Enjoy!

Tool Requirements

Required Tools

In order to build Bforartists, you need "building tools" like **compilers, "sourcing tools"** for the materials, "blueprints" like the **repository** and **libraries,** and other third party "resources" that are classified as **dependencies.**

Building Tools

Microsoft Visual Studio 2022 Community: https://visualstudio.microsoft.com/de/downloads/

Cmake: http://www.cmake.org/download/

Sourcing Tools

Git: https://git-scm.com/

Tortoisegit: https://tortoisegit.org/

> ***Quick Tip:*** *You can of course also do everything from the Git command line. But Tortoisegit makes things much easier with a GUI.*

TortoiseSVN for the dependencies: **http://tortoisesvn.net/index.de.html**

You may also need also SlikSVN since TortoiseSVN will not work from the command line. And so a make update will not work: **https://sliksvn.com/download/**

Other Tools

Python 3.x : https://www.python.org/

CUDA Developer Tools: https://developer.nvidia.com/cuda-downloads

Nvidia Optix SDK (requires registration for download)
https://developer.nvidia.com/designworks/optix/download

Quick Tip: *Hitting "Next" "Next" "Next" to install the above is generally ok and risk free.*

HIP: https://www.amd.com/en/developer/resources/rocm-hub/hip-sdk.html

Optional Tools

We need an installed version of Blender if you want to modify the icons. You need the Windows Installer version, the one with the *.msi ending. Make sure that the path to the Blender.exe is added to the Windows system variables. You can find all our Windows Installer verions here: http://download.blender.org/release/

Quick tip: *Check out Page 9 to learn how to add System Variables*

7Z or similar zip software that can handle tar.gz archives: http://www.7-zip.org/

Nice to have but not necessary is **Grepwin** to search the source code:
http://stefanstools.sourceforge.net/grepWin.html

Tools like **Meld**: http://meldmerge.org/ , **Atom:** https://atom.io/ or Winmerge https://winmerge.org are also beneficial but not required. They allow you to use an overview, syntax highlights and document comparisons. Tortoisegit comes also with a code comparison tool.

Downloading the Repository

So you have downloaded and installed the tools – one after the other. Now it's time to download the repository (the blueprints) and get the libs (the materials).

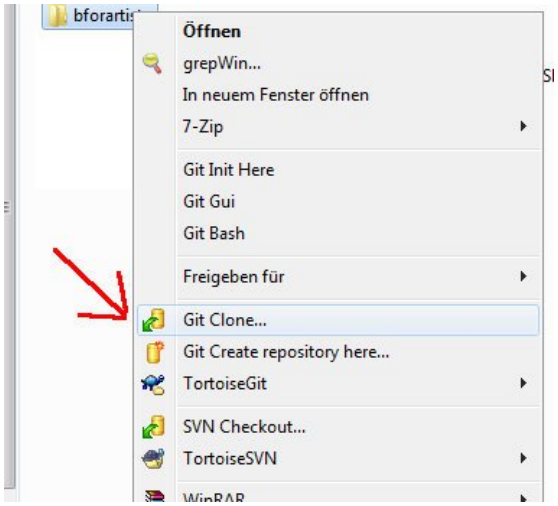
1. Choose a location.

Quick Tip: *When possible choose or create a folder directly at the drive level.*

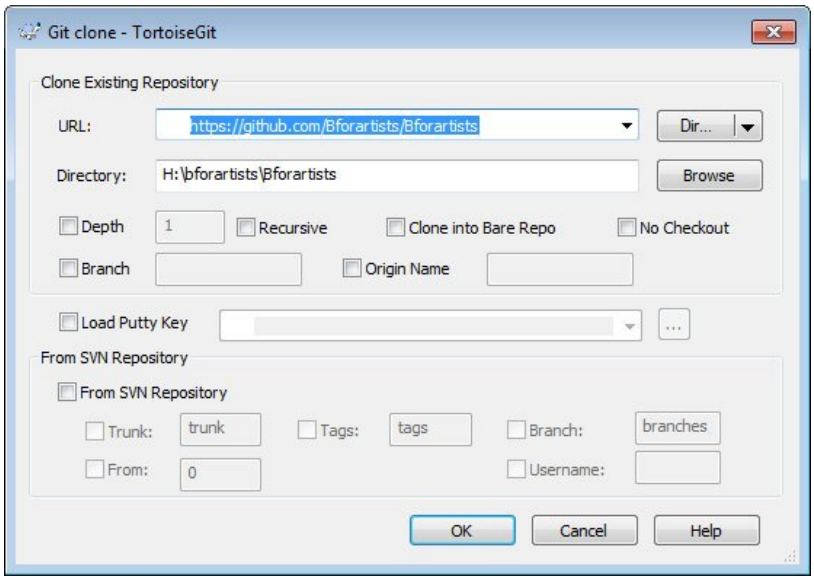
I have chosen drive **H:**\, and have created the folder **bforartists**, which is our target folder.

2. Open your browser and navigate to https://github.com/Bforartists/Bforartists . This brings you to the repository URL recognized by Git.

3. Right click your target folder and choose "git clone" in the right mouse button (rmb) menu.



4. You navigated to the Bforartists Github repository in step 2, so Git should have copied the right link into the URL field already. If not, then type in the URL manually: https://github.com/Bforartists/Bforartists.



5. Click okay. Git will download the Bforartists repository now.

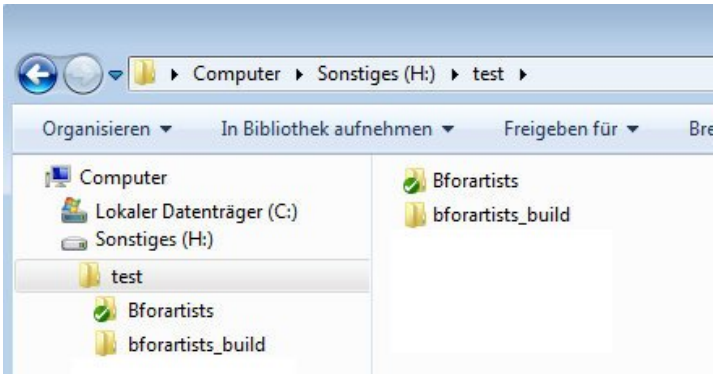
6. If everything has gone well then you will get the following result (notice the green tickbox):

Creating the Binary Folder

On with some more preparations. This one is just needed when you choose the long compilation way, configure cmake manually and then compile with visual studio by clicking at the sln file.

You need a folder where **Cmake** builds the Binaries and where **Visual Studio** compiles the result. In order to do so, create another folder alongside the repository.

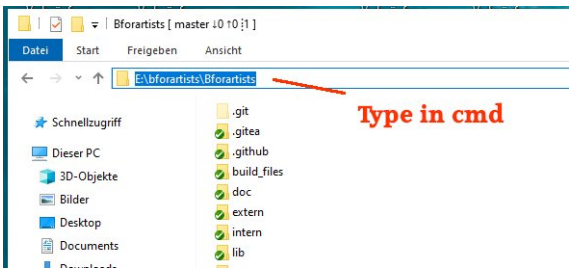
You can name this folder whatever you like. I will call it **bforartists_build**.

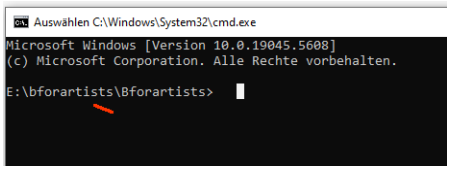


Installing the Dependencies

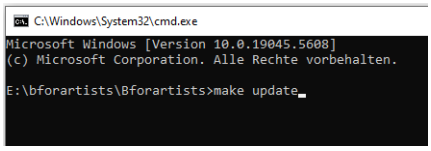
What is most likely missing now are the dependencies. The libs. Bforartists is a fork of Blender, and uses the same dependencies. Here we best use the console. A *make update* in the console will grab them.

So enter the Bforartists folder in the explorer, and in the adress line type in *cmd* . This will open the windows command line tool. The cmd.exe. And now we can tpye in some cryptic commands.





As told, a *make update* pulls all the new files from the Bforartists repository. And in case the libs does not exist yet, also pulls the missing libs.



However, we are not Blender. And to update the libaries at a later time you need to do a

git submodule foreach git pull --rebase origin main

from time to time

Updating the Repository

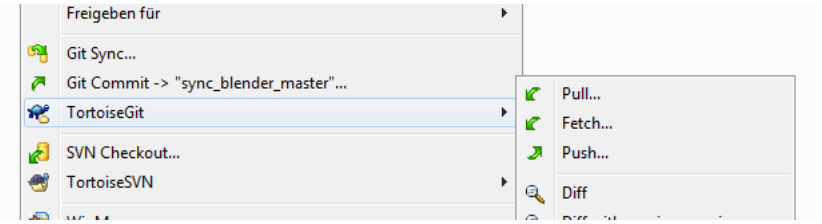
Before you compile you might want to update the repository (the blueprints) to download the latest changes.

We have basically already covered it above. Open console at the Bforartists repository, type in *make update* , followed by a *git submodule foreach git pull --rebase origin main, and you are settled.*

The Tortoisegit way is as followed:

1. First “Fetch”. This updates the repository index with branches, changes etc. This command only updates the index, not the repository itself. To update the repository you must do a “Pull”.

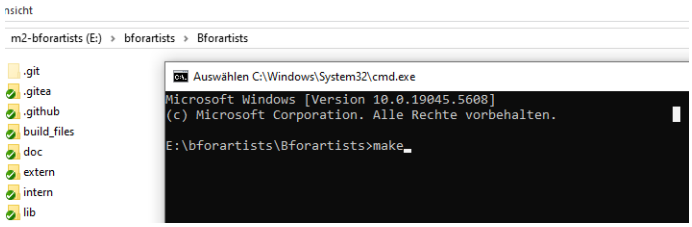
2. So first **Fetch**, then **Pull**. Your repository should be up to date again.



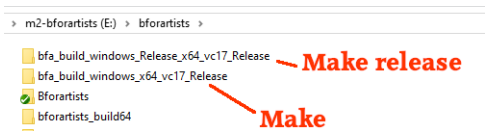
Compiling, quick way, console

You have downloaded and installed everything. The repository is up to date. Now we need to compile the source code. There are two ways. Easy and with more control.

The easy way is to type in a *make* or a *make release* in the console. The rest happens automatically. Even the target folder gets created.



make creates a minimum configuration. No cuda, no hip, no oneapi. This version usually compiles in 5-10 minutes. The *make release* version compiles with all the bells and whistles.



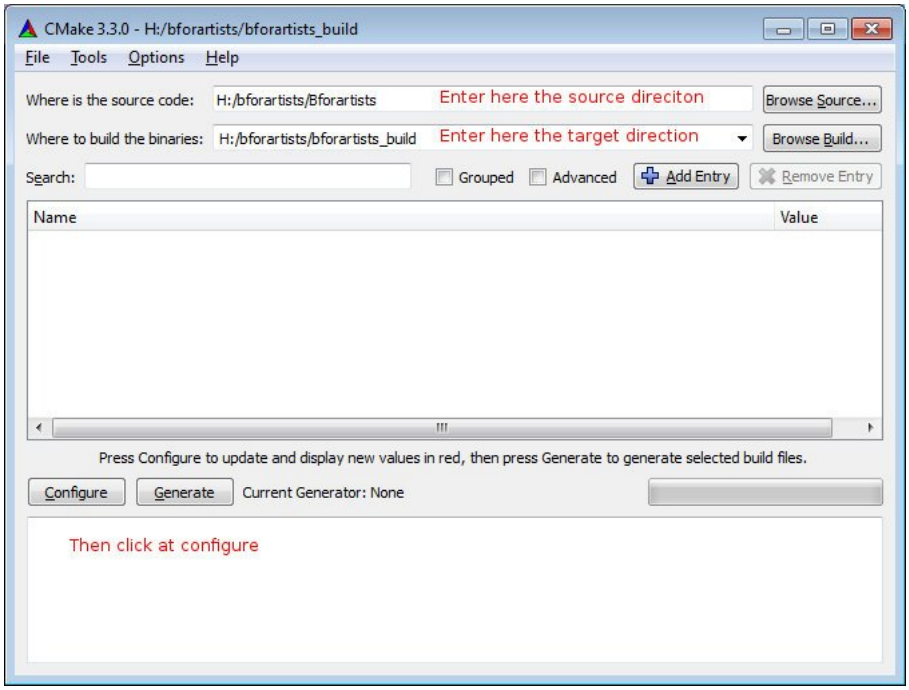
Compiling, long way, Cmake

Then there is the manual way. This one is good when you want to debug and toy around with different cmake settings.

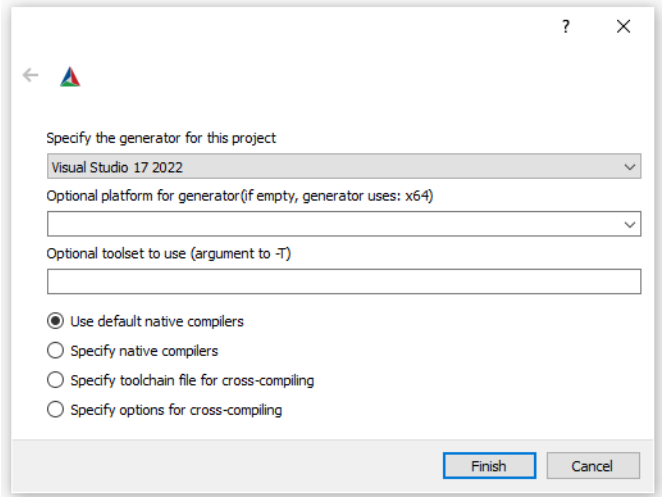
Time to fire up **Cmake** for the first time and create the solution file for Visual Studio.

Cmake Preparation

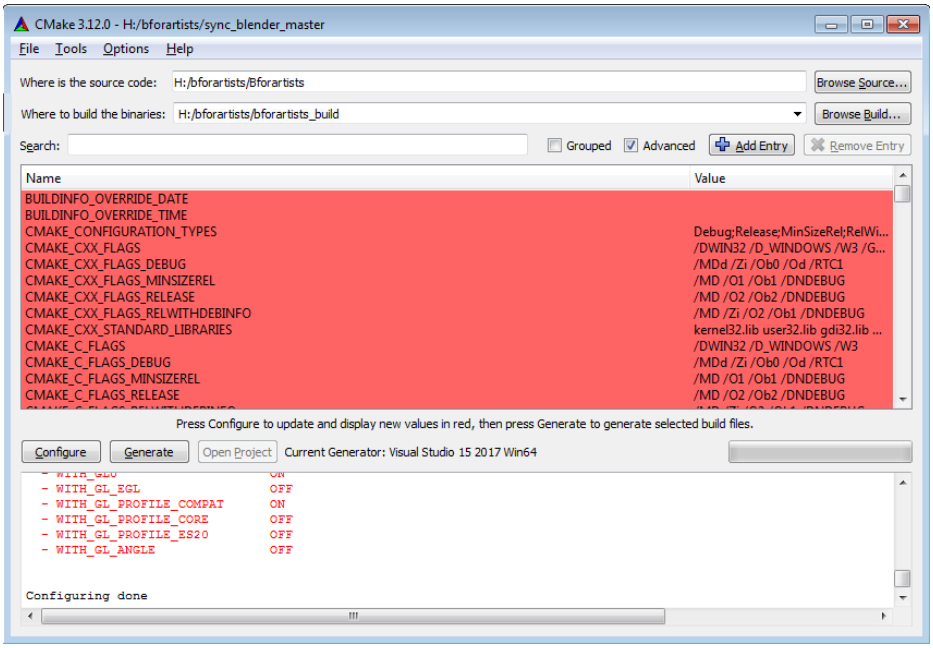
1. Enter the paths to the repository source code and to the build folder then click on **Configure**.



2. A window will pop up asking you what Visual Studio version to use. Usually you don't need to change anything here when you just have Visual Studio installed. Else Choose **Visual Studio 17 2022** in the "Specify the Generator for this project" dropdown box to compile the 64 bit version. You can also choose x64 in the Optional platform for generator dropdown box below. But leaving it blank will also work. It chooses x64 as the target platform automatically then.



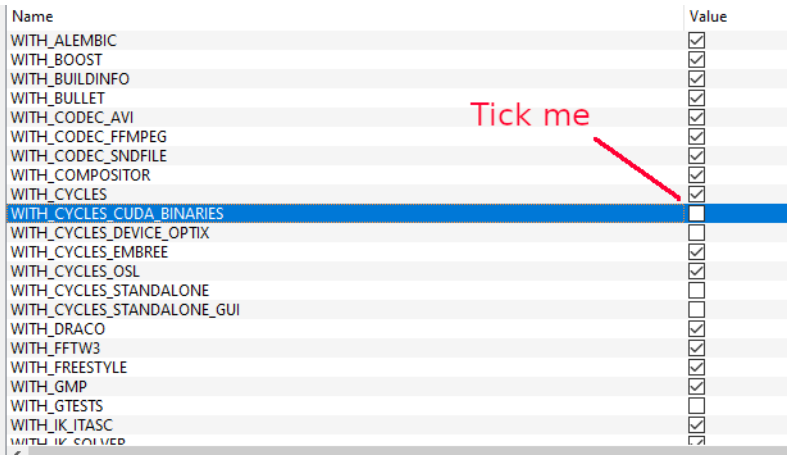
3. Let Cmake run through once. Then click **Configure** again. The second run will turn the upper window back to normal – where the option toggles are displayed.



4. Sometimes, if there are incomplete libraries, missing or corrupt dependencies, and/or previous damaged builds, you might get errors in the bottom window. If you have errors to resolve, skip ahead to the next chapter.

5. If you got no errors, only warnings, then you can proceed to the next steps:

Activate all needed features.



Some features are off by default. **Cuda Binaries**, **Alembic, Open VDB** and **Open Subdiv** needs to be manually activated. You must activate Cuda to have GPU acceleration in the software. And so on.

The current release settings are:

WITH_BUILDINFO OFF

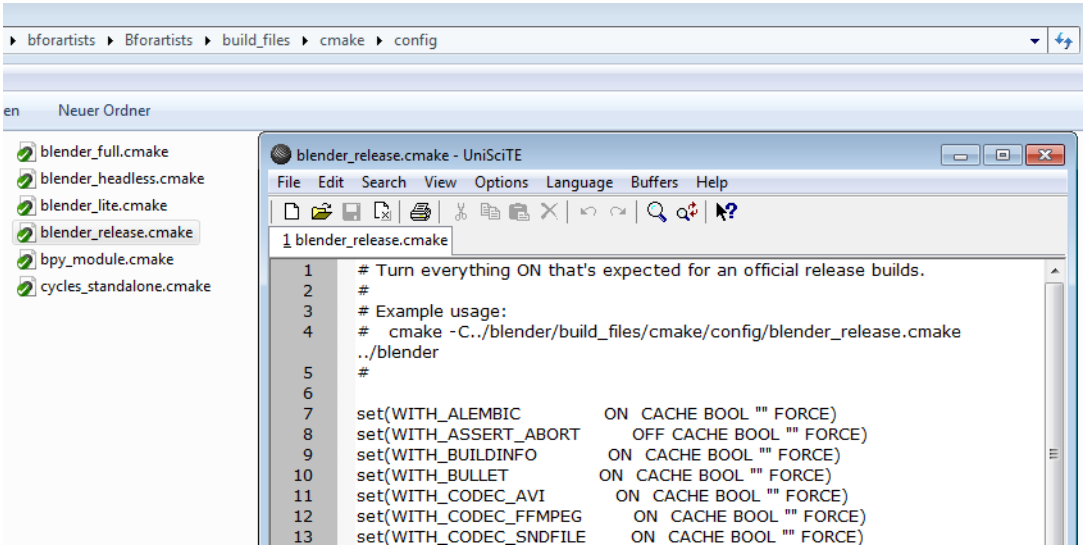
WITH_CYCLES_CUDA_BINARIES ON
WITH_CYCLES_DEVICE_OPTIX ON

You need to have the correct Cuda version and Optix installed.

Quick tip: *The settings for a release version of Bforartists can be found in the build_files/cmake/config folder in the repository. The blender_release.cmake file contains all needed settings.*

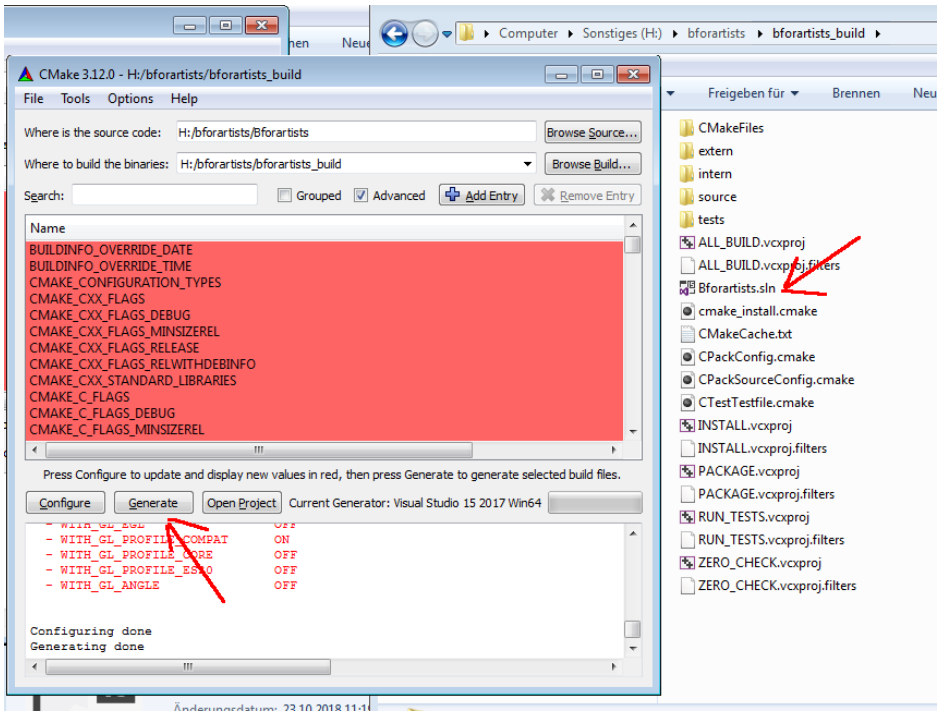
To use it works the same way than in Blender. First open cmd. Then navigate to the directory that contains the Bforartists repository. **make** *starts the build process. When you type in* **make release**, *then the build process uses the settings defined in the blender_release.cmake file.*

We recommend the way across Cmake. That way you have full control.



Generate

When all errors are fixed Click on Generate. This will now create the solution files in the bforartists_build folder. You can now open the B**forartists.sln** file in Visual studio.

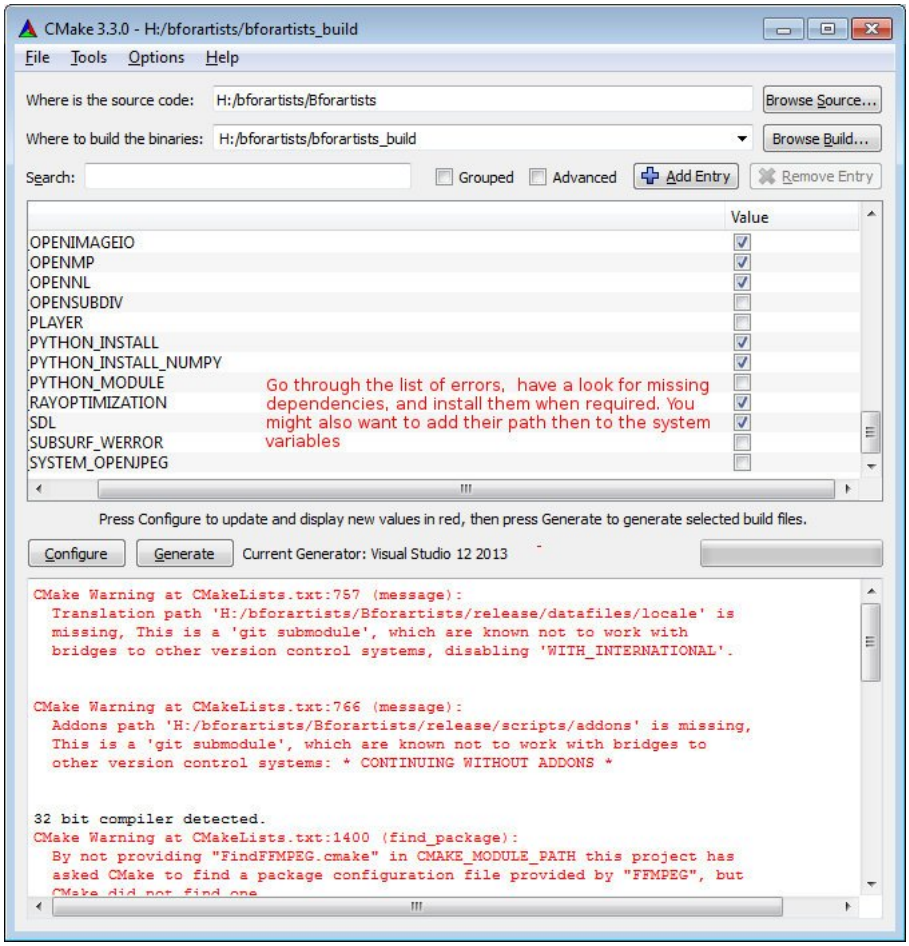


Troubleshooting Cmake Errors

You might get errors when you run Cmake for the first time: missing or corrupt dependencies for example. Make sure you have downloaded and installed all of them. You might also have missing path variables by which the installed software gets recognized by Cmake. These are just some possible errors. Feel free to drop a task in the Github tracker or Forum space for Bforartists asking for help if this step is over your head. We will do our best to help you out.

Cuda

You might get a **CUDA_SDK_ROOT_DIR-NOTFOUND** warning in the upper section. Bforartists can compile with this error present. If you want to fix this, search for nvcc in the C:\Program Files\Nvidia Corporation folder.



Cmake Warning at intern/cycles/kernel/CmakeLists

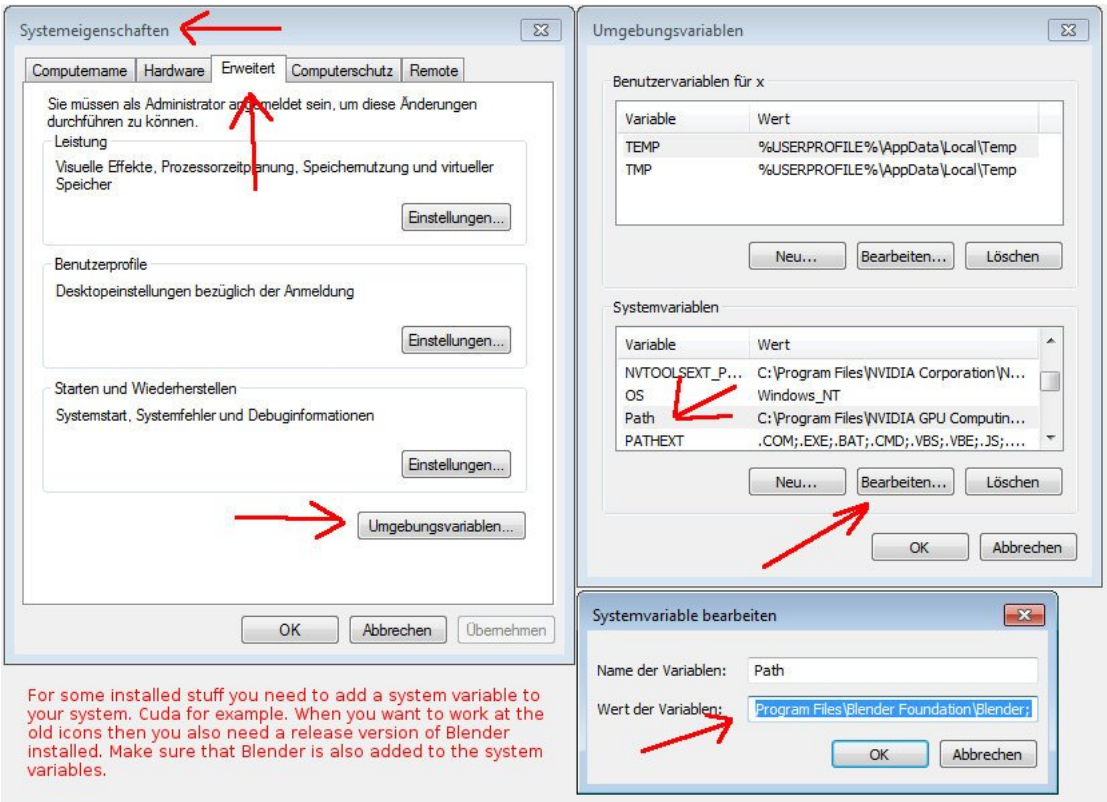
Sometimes you will get this warning (which can cause compiling errors) if you have the incorrect CUDA Developer Tools installed. Uninstall and reinstall the correct version. 9.1.85 is the last tested version for Bforartists 1.0.0

```
64 bit compiler detected.
Visual Studio 2013 detected.
CMake Warning at build_files/cmake/platform/platform_win32.cmake:436 (message):
  LLVM debug libs not present on this system.  Using release libs for debug
  builds.
Call Stack (most recent call first):
  CMakeLists.txt:899 (include)


Found OpenMP_C: -openmp
Found OpenMP_CXX: -openmp
Found OpenMP: TRUE
Found CUDA: C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v9.2 (found version "9.2")
CUDA nvcc = C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v9.2/bin/nvcc.exe
nvcc not supported for this compiler version, using cycles_cubin_cc instead.
CMake Warning at intern/cycles/kernel/CMakeLists.txt:332 (message):
  CUDA version 9.2 detected, build may succeed but only CUDA 8.0 is
  officially supported


Blender Skipping: (bf_intern_ctr;bf_intern_opencl;extern_sdlew)
Disabling Cycles tests because tests folder does not exist
Configuring done
Generating done
```

Adding System variables

Some things are required to be added to the system variables.

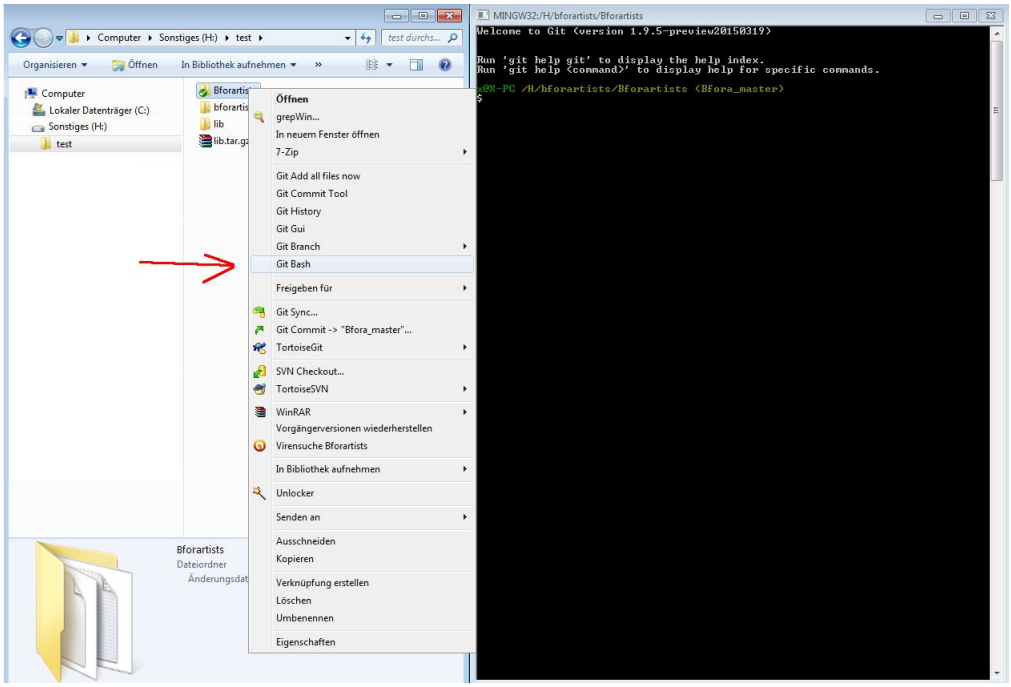
Here's an example where you add the path to the Blender exe in case it is missing. Windows System control, Advanced system settings -> System properties.



Missing Language Paths in Blender

When you compile Blender, and not Bforartists, then you might have the above errors with missing language paths. Bforartists does not have this problem anymore, we have united everything in one repository. Blender

still has this issue. Here's how to fix it:



Type in the following commands into the bash console, one after another. Wait until each single task is finished:

```
git submodule update --init --recursive
git submodule foreach --recursive git checkout master
git submodule foreach --recursive git pull --rebase origin master
```

This should download the needed submodules. Cmake shouldn't moan anymore.

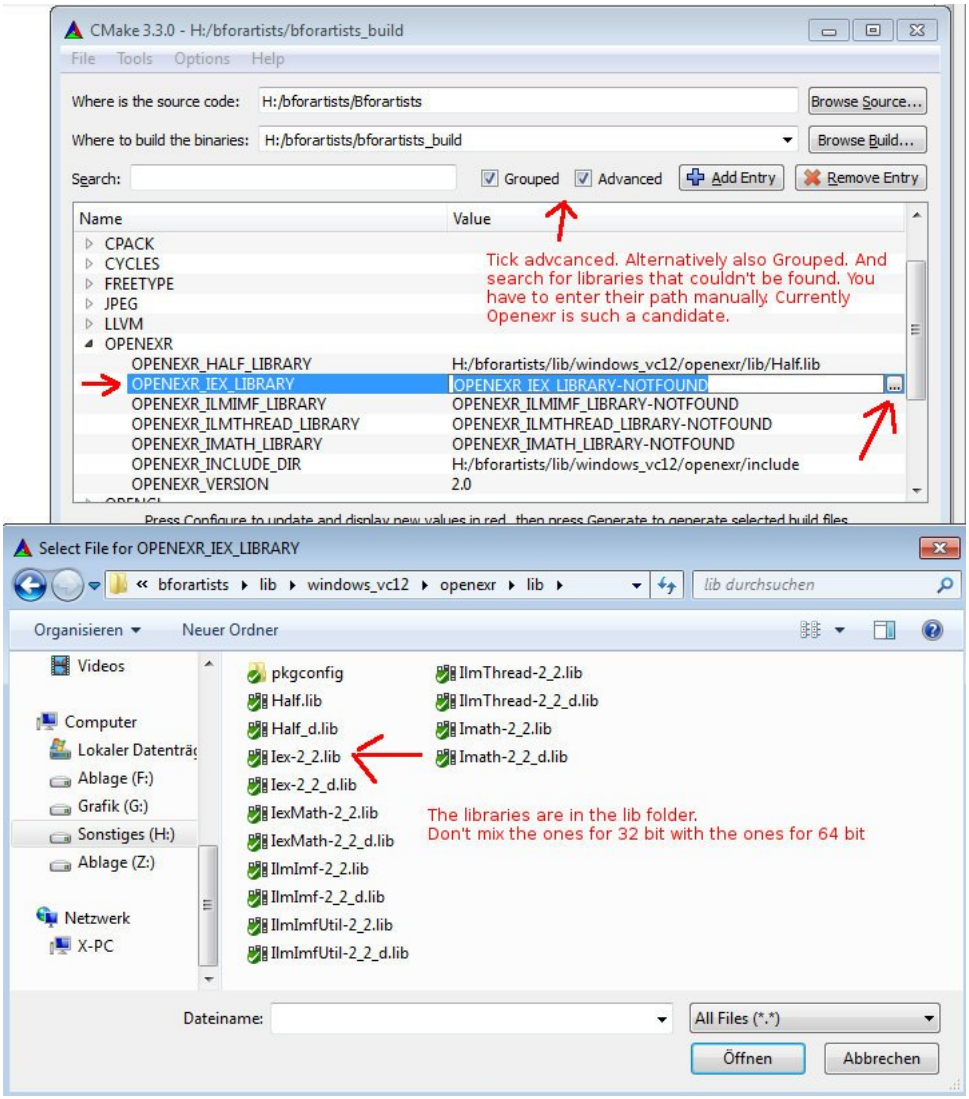
Later, when you want to update the submodules, type in:

```
git pull --rebase
git submodule foreach --recursive git pull --rebase origin master
```

This just a sidenote for Blender itself. Bforartists does not need this step.

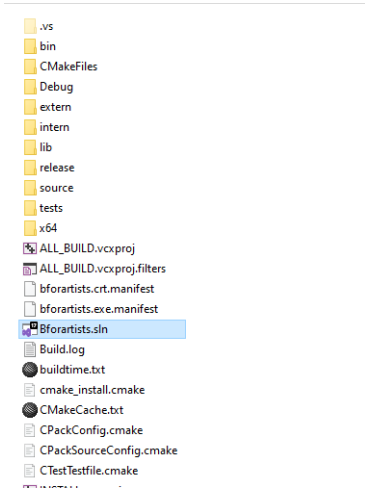
Open EXR

Often the Openexr libraries are not found on Windows, but you can fix them by hand whenever you want. It is not necessary to fix this error to successfully compile and run Bforartists. You can happily ignore the warning. It's just a cosmetic problem in the Cmake GUI. If you choose to fix them by hand, then don't mix the libs for the 32 bit version with the libs for the 64 bit version.



Compiling with Visual Studio

Cmake should have produced everything needed in the build folder. Now you can open the sln file and compile it with Visual Studio.

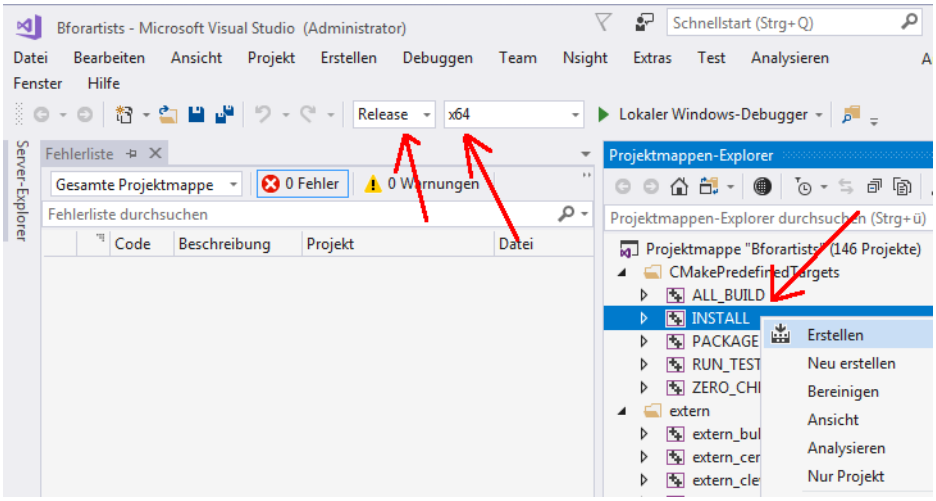


1. Click on the **Bforartists.sln** file to open Visual Studio.

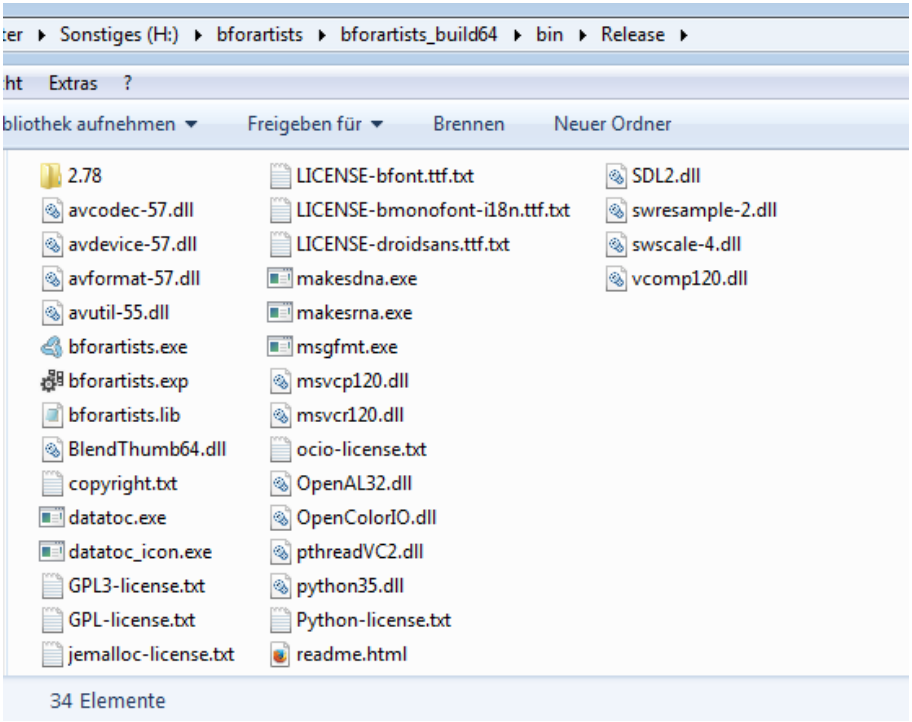
2. In VS, change the build type to **Release** (VS starts usually with **Debug**, but that's only of interest for development).

3. Select the **Install** item in the Solution Explorer, right click on it, and choose **Build**.

4. Visual Studio should now start the Build process.



5. When everything works as planned then you should have a result in the **bin/Release** folder.



Congratulations!

You have compiled Bforartists for the first time. You should be able to work with the source code now. Double click on the *.exe to run and see that everything is working fine. When it does, you are a winner.

> ***Quick Tip:*** *Now that it's compiled, you can do Python code changes to the built data and test it almost live. Don't forget to copy your python changes in the* ***bin/Release*** *folder back to the Repository or else when you rebuild, you will loose all your changes!*
>
> *For C code changes, you will need to make the changes in the repository before building, then re-build to test. If you get errors on re-build, just delete the old build in the* ***bin/Release*** *folder.*