Julia Unity Package - How to get started

Get the package

- Download the Julia Unity Package zip archive in the repository under builds>JuliaUnityPackage.zip.
- OR download the package or only the archive from GitHub and add it manually to your project's Packages folder.
- OR copy the package's GitHub URL and choose "Add package from Git URL" in the Unity Package Manager.
- Optional: import the package's samples (StreamingAssets folder with Inclusions.txt and JuliaScripts folder with helper.jl and load_dependencies.jl)

Setup the package Dependencies

- Move the imported StreamingAssets folder directly below your Assets folder. If you already have a StreamingAssets folder move the Inclusions.txt file into the already existing StreamingAssets folder.
- Move the JuliaScripts folder into your Unity project directory on the same level as the Assets folder. This has to be done in the explorer.
- Now move your own Julia files into the JuliaScripts folder and reference them in the Inclusions.txt file.
- The structure of your Unity project should now look like this:

```
Unity Project Directory
├── Assets
│   ├── ...
│   └── StreamingAssets
│       └── Inclusions.txt
├── Packages
│   ├── Julia Unity Package
│   └── ...
└── JuliaScripts
    ├── helper.jl
    ├── load_dependecies.jl
    └── ...
```

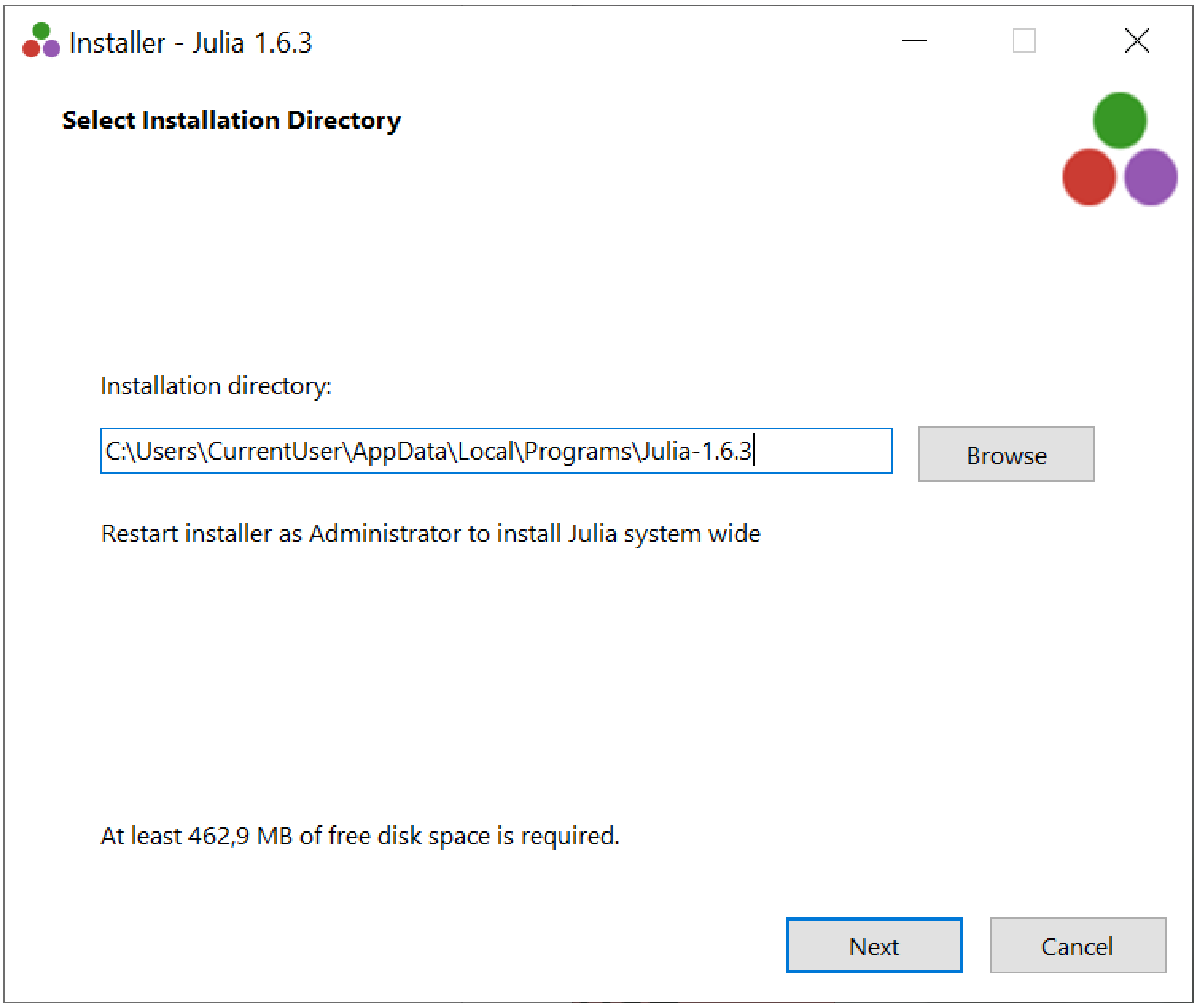
Julia Setup

- Open the JuliaPackage Editor Window in the Unity Editor under Window>JuliaPackage and click "Setup Julia installation".
- The package will automatically search for a Julia installation under "C:\Users\CurrentUser\AppData\Local\Programs\Julia-1.6.3".
- If Julia isn’t found you will be asked to install it via "Download". This will download the "julia-1.6.3-win64.exe" for installing Julia-1.6.3.
- If you already have Julia-1.6.3 installed on your machine, but in a different directory, you can enter this valid path into the input field and update it by clicking "update Julia directory".
- After installing Julia or passing a valid installation path click "Setup Julia installation" again.

- Now you can click "Check Julia installation" or "Setup Julia installation" again to verify that Julia is installed, necessary environment variables are set and that Julia can be called from within Unity.

Julia Installation:

- Execute the downloaded file "julia-1.6.3-win64.exe" to install Julia-1.6.3.
- Choose the recommended installation path.



Installer - Julia 1.6.3

Select Installation Directory

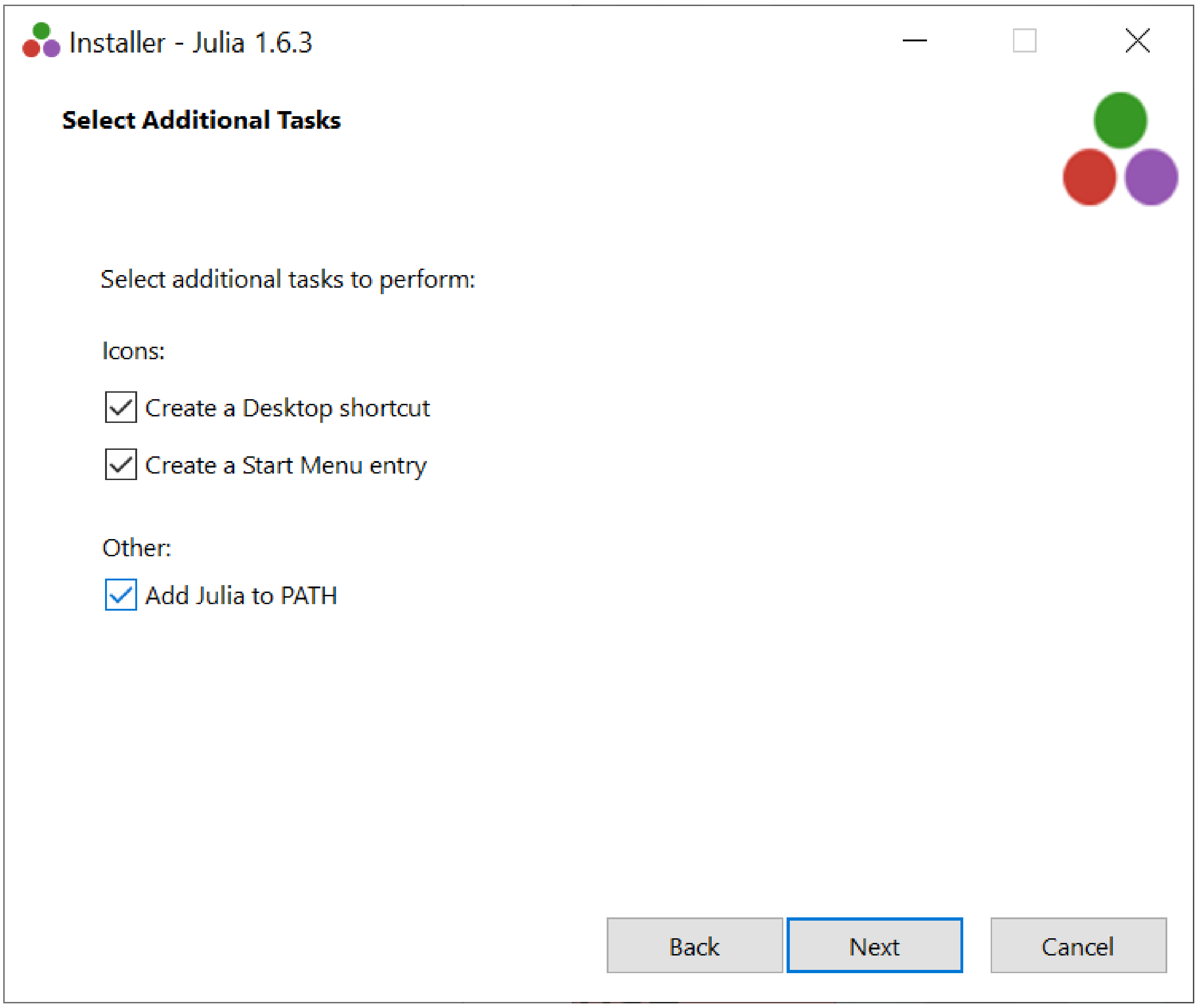
Installation directory:

C:\Users\CurrentUser\AppData\Local\Programs\Julia-1.6.3

Restart installer as Administrator to install Julia system wide

At least 462,9 MB of free disk space is required.

- Check the box "Add Julia to PATH".


Installer - Julia 1.6.3

Select Additional Tasks

Select additional tasks to perform:

Icons:

- [✓] Create a Desktop shortcut
- [✓] Create a Start Menu entry

Other:

- [✓] Add Julia to PATH

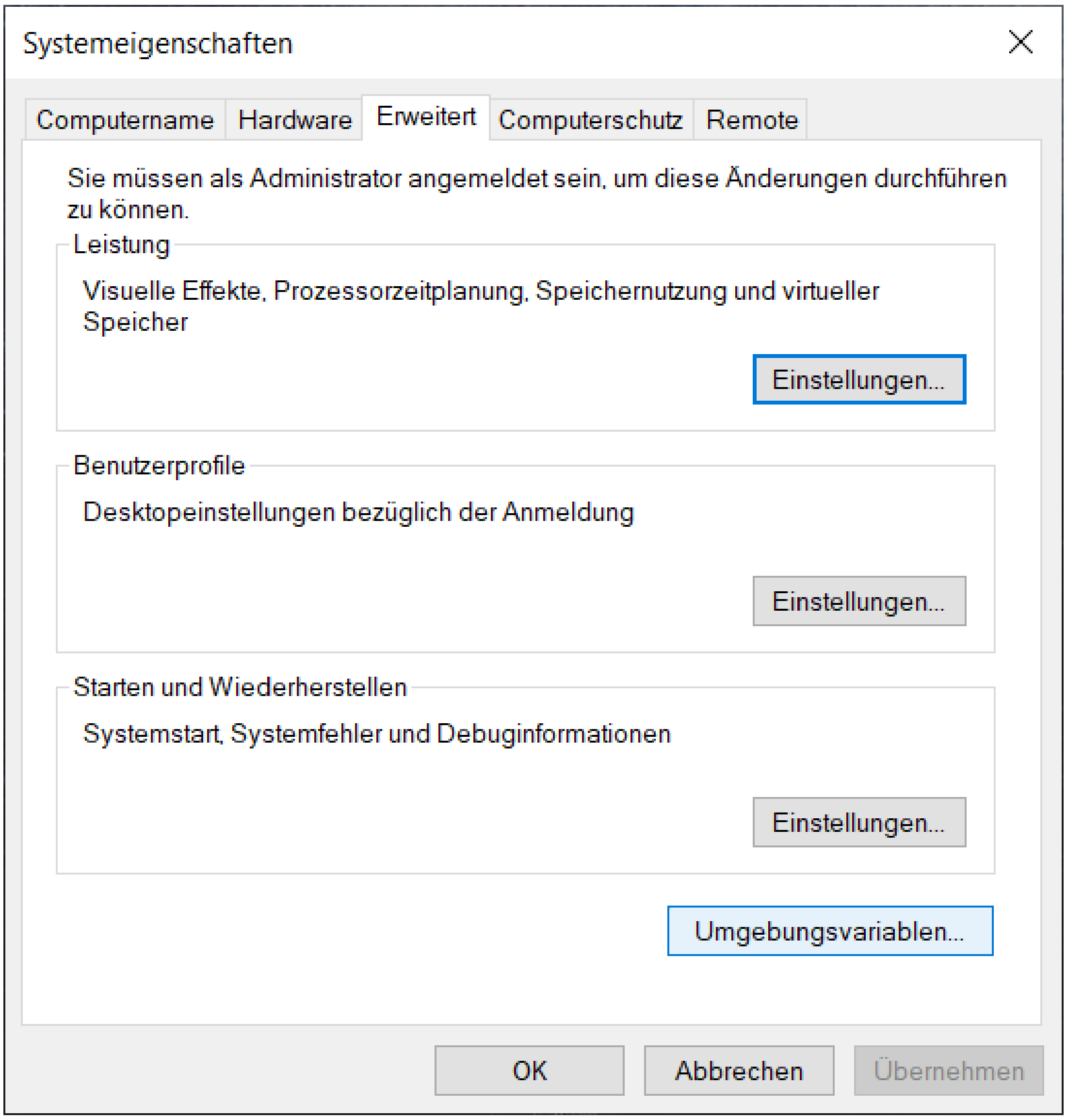
- Finish the installation.

Error Handling:

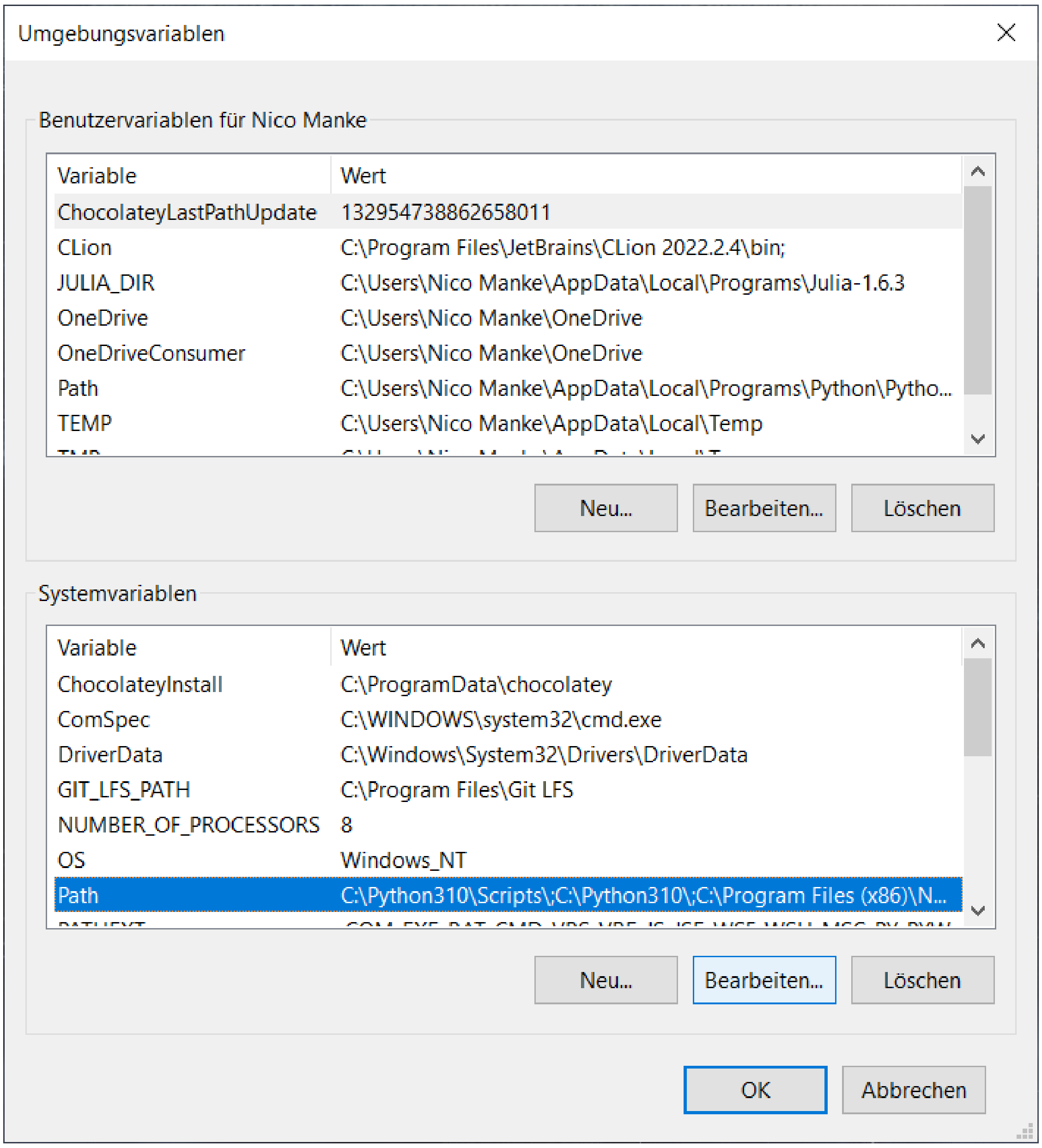
The occurrence of errors is indicated by a DllNotFoundException in the console. This can have several causes:

- You are missing dependencies:
  1. Download and install the latest "Microsoft Visual C++ Redistributable" version from https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170 (at least version 2015-2022) or directly from https://aka.ms/vs/17/release/vc_redist.x64.exe .
  2. Now restart your application.
  3. After this the application should work or should continue the setup process.
- Your system doesn’t allow setting environment variables:
  - *Case Admin User:*
    1. Search and open "environment variables" in the search bar (German: „Systemumgebungsvariablen bearbeiten“).

2. Click "Environment Variables" to the bottom right.


Systemeigenschaften
Computername
Hardware
Erweitert
Computerschutz
Remote
Sie müssen als Administrator angemeldet sein, um diese Änderungen durchführen zu können.
Leistung
Visuelle Effekte, Prozessorzeitplanung, Speichernutzung und virtueller Speicher
Einstellungen...
Benutzerprofile
Desktopeinstellungen bezüglich der Anmeldung
Einstellungen...
Starten und Wiederherstellen
Systemstart, Systemfehler und Debuginformationen
Einstellungen...
Umgebungsvariablen...
OK
Abbrechen
Übernehmen

3. Select „Path“ under System Variables.



Umgebungsvariablen

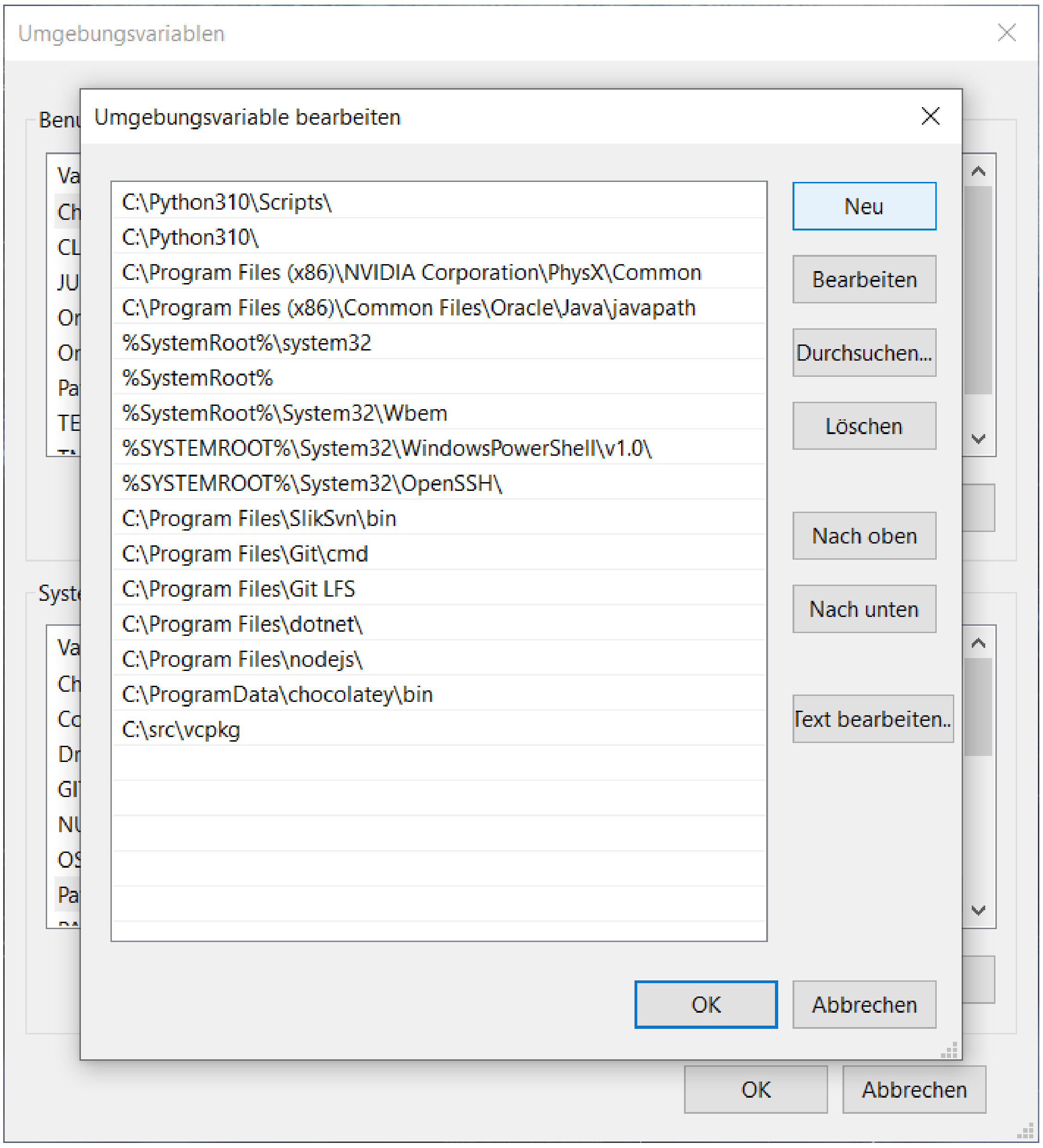
Benutzervariablen für Nico Manke

Variable	Wert
ChocolateyLastPathUpdate	132954738862658011
CLion	C:\Program Files\JetBrains\CLion 2022.2.4\bin;
JULIA_DIR	C:\Users\Nico Manke\AppData\Local\Programs\Julia-1.6.3
OneDrive	C:\Users\Nico Manke\OneDrive
OneDriveConsumer	C:\Users\Nico Manke\OneDrive
Path	C:\Users\Nico Manke\AppData\Local\Programs\Python\Pytho...
TEMP	C:\Users\Nico Manke\AppData\Local\Temp

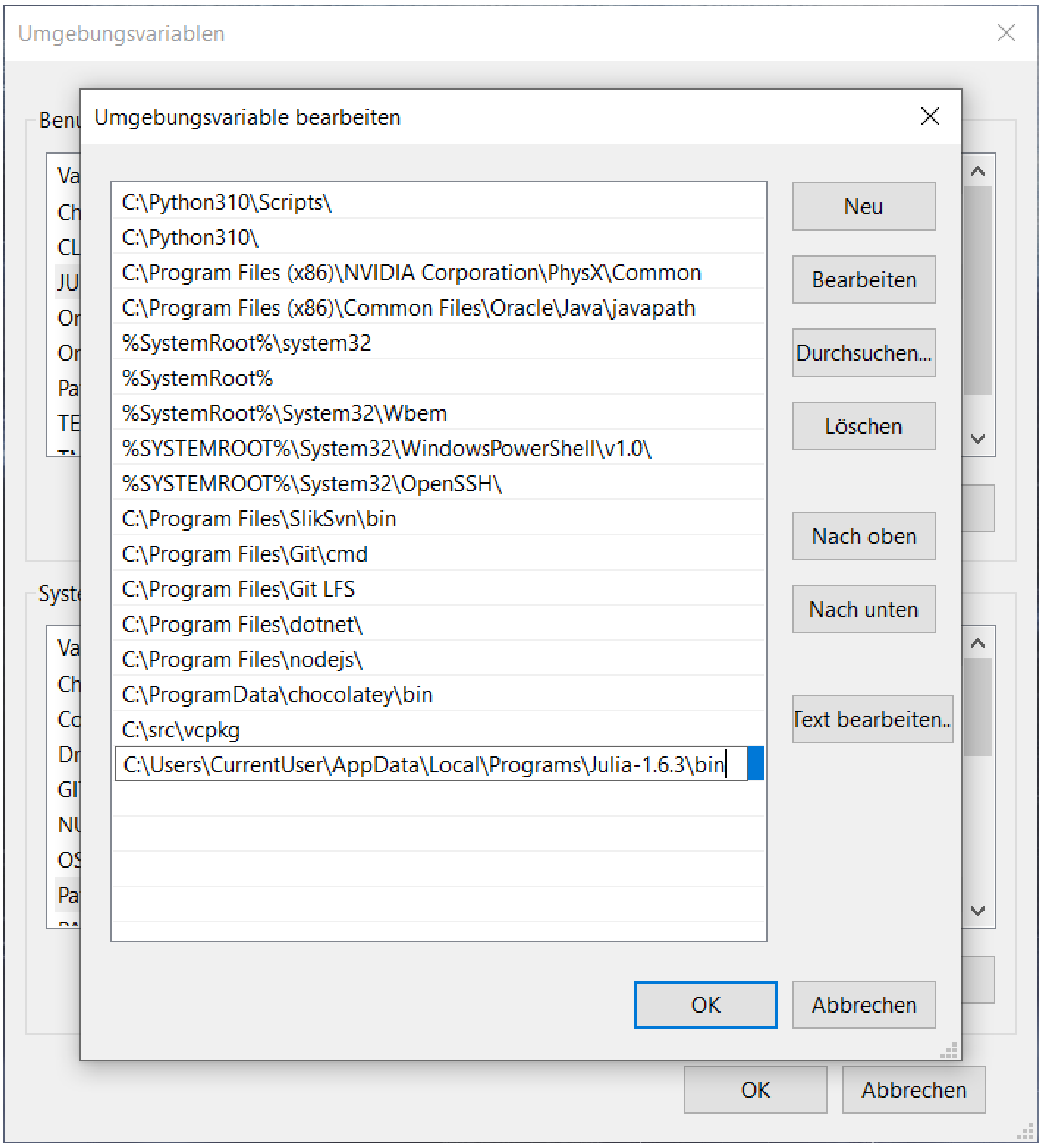
Systemvariablen

Variable	Wert
ChocolateyInstall	C:\ProgramData\chocolatey
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
GIT_LFS_PATH	C:\Program Files\Git LFS
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Python310\Scripts\;C:\Python310\;C:\Program Files (x86)\N...

4. Click "new" and paste the path of your Julia directory with "\bin" appended to the end (for example: "C:\Users\CurrentUser\AppData\Local\Programs\Julia-1.6.3\bin")



```
C:\Python310\Scripts\
C:\Python310\
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\SlikSvn\bin
C:\Program Files\Git\cmd
C:\Program Files\Git LFS
C:\Program Files\dotnet\
C:\Program Files\nodejs\
C:\ProgramData\chocolatey\bin
C:\src\vcpkg
```



```
C:\Python310\Scripts\
C:\Python310\
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\SlikSvn\bin
C:\Program Files\Git\cmd
C:\Program Files\Git LFS
C:\Program Files\dotnet\
C:\Program Files\nodejs\
C:\ProgramData\chocolatey\bin
C:\src\vcpkg
C:\Users\CurrentUser\AppData\Local\Programs\Julia-1.6.3\bin
```

5. Now confirm the changes.

- *Case Non-Admin User:*
  1. Type „cmd“ into the search bar. This will open the windows console.
  2. Now type "set PATH=%PATH%;C:\your\Julia\directory\bin" to set it for the user.
     ```
     J:\>set PATH=%PATH%;C:\Users\CurrentUser\AppData\Local\Programs\Julia-1.6.3\bin
     ```
  3. Or type "setx PATH %PATH%;C:\your\Julia\directory\bin" to set it for the whole system (requires restart if set isn’t used additionally).
     ```
     J:\>setx PATH %PATH%;C:\Users\CurrentUser\AppData\Local\Programs\Julia-1.6.3\bin
     ```

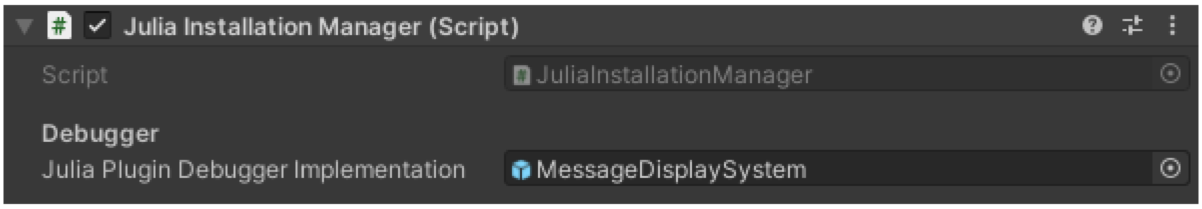
- The JuliaInterface.dll is missing in the package's Plugins folder for some reason.
  1. Download the JuliaInterface.dll again and put it into the package's Plugins folder.

Required Unity Project Settings

- In order to use the setup functionality in the built version of your project, the Unity project's API Compatibility Level* needs to be .NET 4.0 or .NET Framework. You can change it under Project Settings>Player>Api Compatibility Level*.

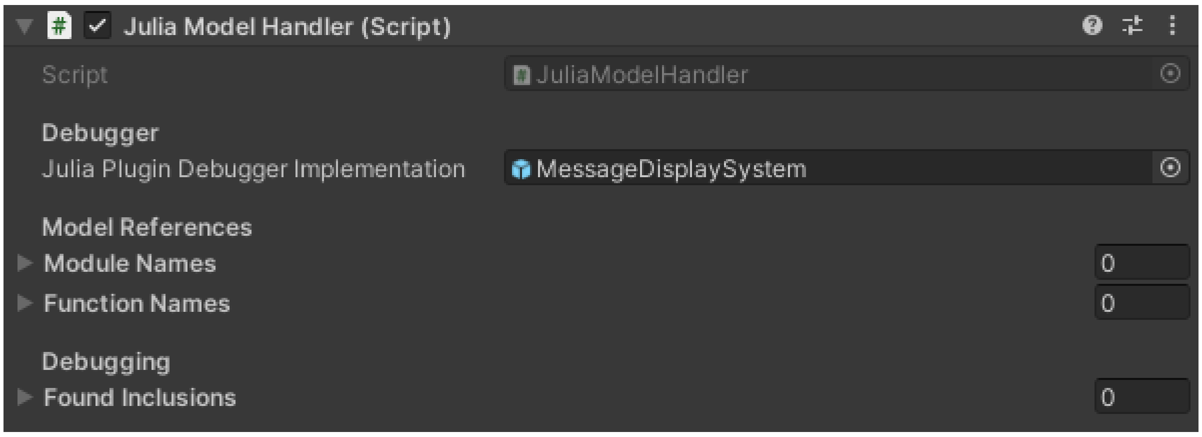
MonoBehaviours in the Scene

In order to use Julia during runtime some of the above-mentioned MonoBehavoirs need to be placed inside the scene.

1. The JuliaInstallationManager should be placed in your starting scene to test and set up the Julia installation if necessary. This instance should also receive an instance of your IJuliaPluginDebugger implementation. In the following example the MessageDisplaySystem is such an implementation:


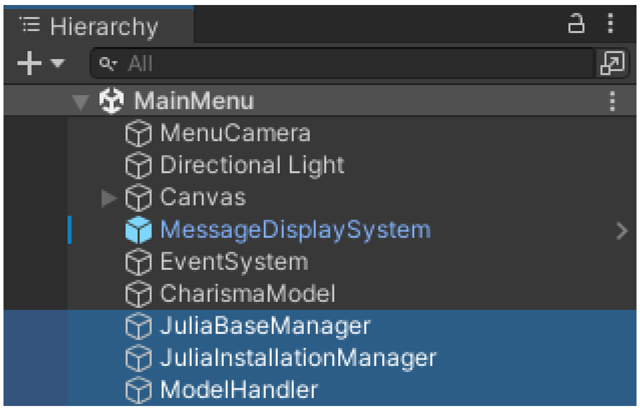
   Julia Installation Manager (Script)

Script	JuliaInstallationManager
Debugger	
Julia Plugin Debugger Implementation	MessageDisplaySystem

2. The JuliaBaseManager should be placed in your starting scene so that the necessary helper functions for handling multi-dimensional Julia arrays can be loaded and referenced.
3. It is recommended to implement the IJuliaPluginDebugger interface inside a class deriving from Monobehaviour. By doing this you can parse an instance of this class to the JuliaInstallationManager or the JuliaModelHandler inside the Inspector for logging and debugging occurring errors and warnings.
4. If you want to use an inspector-oriented solution for referencing Julia functions and modules you could place an instance of the JuliaModelHandler inside the scene. This is purely optional and based on your preferences. This instance should also receive an instance of your IJuliaPluginDebugger implementation. In the following example the MessageDisplaySystem is such an implementation:


   Julia Model Handler (Script)

Script	JuliaModelHandler
Debugger	
Julia Plugin Debugger Implementation	MessageDisplaySystem
Model References	
Module Names	0
Function Names	0
Debugging	
Found Inclusions	0

- The following example is taken from the MGM Simulation and shows instances of JuliaBaseManager, JuliaInstallationManager, and JuliaModelHandler (here ModelHandler) in the scene hierarchy. The MessageDisplaySystem represents an example implementation of the IJuliaPluginDebugger interface:


   ```
   Hierarchy
   All
   MainMenu
       MenuCamera
       Directional Light
       Canvas
       MessageDisplaySystem
       EventSystem
       CharismaModel
       JuliaBaseManager
       JuliaInstallationManager
       ModelHandler
   ```