



TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology



BCA-GUIDE

Manual

December 2022

Alexander Redl, Christian Cupak, Friedrich Aumayr

TU Wien, Institute of Applied Physics

Contents

1. System requirements.....	3
a. Requirements for Python interpreter.....	3
b. Requirements for executable file	4
c. Compiling the executable file.....	4
d. BCA codes	4
2. Configure a simulation.....	5
a. Configuration for SDTrimSP.....	7
b. Configuration for TRIDYN 2022	8
3. Perform a simulation	10
a. Simulation setup	11
b. Files preview	12
c. Log files.....	12
d. Output files.....	12
e. Simulation results	13
4. Perform multiple simulations	14
5. Copy the simulation to a different simulation program	15
6. Internal structure.....	16
7. Add a new simulation program	19
a. SimulationInput class	19
b. SimulationOutput class	20
Bibliography.....	21
List of figures and tables	22

The BCA-GUIDE (**B**inary **C**ollision **A**pproximation - **G**raphical **U**ser **I**nterface for **D**isplaying and **E**xecution of simulations) was developed to provide an easy-to-use graphical user interface for BCA codes. The layout and functionality are based on the GUI for SDTrimSP [1], with the addition of supporting TRIDYN [2] besides SDTrimSP [3]. Another effort was to provide the possibility to add further simulation codes in the future to the GUI with as little effort as possible. The modular structure of the GUI allows for editing, running and analyzing multiple simulations simultaneously for one or multiple simulation settings. For simple applications, the GUI also provides an interface to transform the input files between formats for each simulation code.

BCA-GUIDE is open source, publicly available online on GitHub (<https://github.com/atomicplasmaphysics/BCA-GUIDE>) and distributed under the GPLv3 license. As a condition of free usage, the manuscript of the GUI for SDTrimSP by Szabo, et al. [1] and this manual must be cited in any publication that presents results obtained with help of this GUI.

1. System requirements

a. Requirements for Python interpreter

Since the application is written in Python, the GUI can alternatively be started via this interpreter. For this, Python 3 must be installed, which can be downloaded on the official Python website (<https://www.python.org/downloads/>). The GUI was tested with various python versions and operating systems as listed in Table 1. In addition, packages listed in Table 2 must be installed. These packages can be installed using the official package installer for Python (pip), which is normally included in the default installation of Python 3. The command

```
>> pip3 install *package*
```

must be called in the command line, where `*package*` needs to be replaced by the package names listed in Table 2. Finally, the GUI can be launched by executing the `main.py` file with Python 3 using the command

```
>> python3 main.py
```

after navigating to the root directory of the GUI.

Python version	Operating system
3.7.1	Windows 10
3.8.2	Windows 10
3.8.10	Linux Mint
3.10.0	Windows 10
3.10.0	macOS
3.10.7	Windows 10
3.11.0	Windows 10

Table 1: Tested Python 3 versions and operating systems.

Package name	Tested version number
PyQt5	5.15.7
matplotlib	3.5.3
numpy	1.23.2
scipy	1.9.1

Table 2: Necessary Python packages with a corresponding tested version number.

b. Requirements for executable file

By launching the corresponding executable file, the GUI can be easily run. For Windows operating systems the [BCA_GUIDE_windows.exe](#) and for Linux operating systems the [BCA_GUIDE_linux](#) starts the application. Tested operating systems for the executable are listed in Table 3.

Windows	Linux
Windows 7	Linux Mint 20.2
Windows 10	Ubuntu 22.04
Windows 11	

Table 3: List of operating systems where the executable was tested on.

c. Compiling the executable file

Another option for starting the BCA-GUIDE is available via compilation of a single executable file, which does not require a-priori installation of a Python interpreter, a routine is included that uses PyInstaller for bundling all dependencies into a single package. For this purpose, the package *PyInstaller* needs to be installed additionally to the mentioned packages in chapter [Requirements for Python interpreter](#). Executing the [setup/pyinstaller.py](#) script with Python 3 will generate an executable file *BCA-GUIDE* in the root directory of the GUI.

d. BCA codes

Because the GUI is only a user interface for editing and displaying, but does not inherently include the respective BCA codes, these are additionally required to perform a simulation. Currently, the following BCA codes are supported:

- **SDTrimSP**: can be obtained by contacting A. Mutzke (aam@ipp.mpg.de) [3]
- **TRIDYN**, where only the version of 2022 is supported: can be requested from W. Möller (w.moeller@hzdr.de) [2]

For convenience, an empty folder *Simulation Programs* is included, where it is recommended to save the downloaded BCA programs. This simplifies the selection of path parameters during configuration of a simulation.

2. Configure a simulation

The GUI was designed to assign each simulation configuration to its own tab, where preparation, execution and data visualization is included. To create a simulation configuration the tab *Configurations* has to be active, which corresponds to the interface shown in Figure 1 that is displayed on startup. In the *Simulation Configuration* group, a liberal title, the simulation program (e.g., SDTrimSP or TRIDYN) and path variables for the simulation have to be chosen:

- Configuration title: How the simulation configuration will be referred to (liberal name; will appear as header of the new generated tab, and in the *List of Simulation Configurations*).
- Simulation program: Dropdown menu of supported simulation programs.
- Simulation versions: Dropdown menu of supported versions for the specified simulation program.
- Simulation folder: Path to the root folder of the simulation program.
- Simulation binary: Path to the executable file of the simulation program.
- Save folder: Path to the base folder where simulations of this configuration will be saved. This path is optional, by default it is a subfolder in the directory where the GUI is located.

Path variables are selected via a file dialog that will be displayed when clicking the button with three dots next to the corresponding input field.

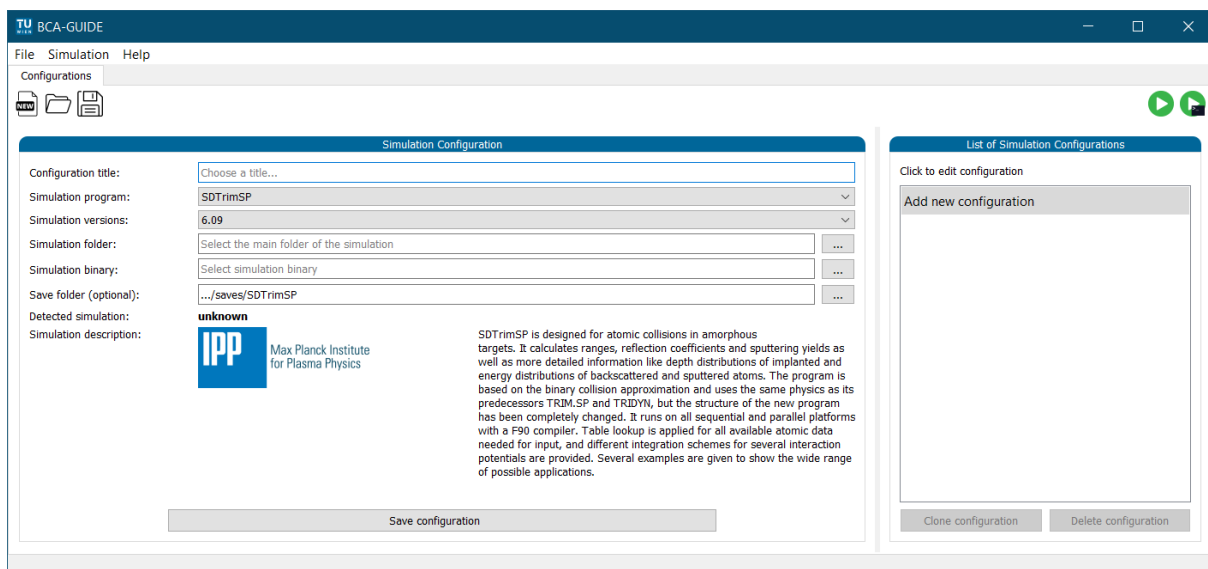


Figure 1: Configurations interface that is displayed on startup and allows for editing simulation configurations.

The GUI will identify the provided BCA code based on the provided input settings and display the result as *Detected simulation*. This allows to check whether all input parameters are set correctly. Depending on the selected *Simulation program* a short *Simulation description* is provided for each simulation. The *Save configuration* button will confirm and create a new

configuration with the provided parameters. Examples for SDTrimSP and TRIDYN 2022 are provided in chapter [Configuration for SDTrimSP](#) and [Configuration for TRIDYN 2022](#) respectively.

To edit an existing simulation configuration, the corresponding simulation configuration is selected from the *List of Simulation Configurations*. The associated parameters are loaded into the *Simulation Configuration* group and can be edited. The *Save configuration* button will confirm and updated the parameters.

Due to the modularity, several simulation configurations can be set up at the same time, which only have to differ in their *Configuration title*. If identical simulation configurations are desired, the previously created simulation configuration can be selected in the *List of Simulation Configurations* and duplicated by pressing the *Clone configuration* button as shown in Figure 2. The duplicated simulation configuration will have an indication added to its title. This is especially useful, if similar simulations with the same BCA code are desired (e.g., for quick parameter sweep studies). Each configuration will have its own tab, which quickly allows to compare results and inputs for each simulation.

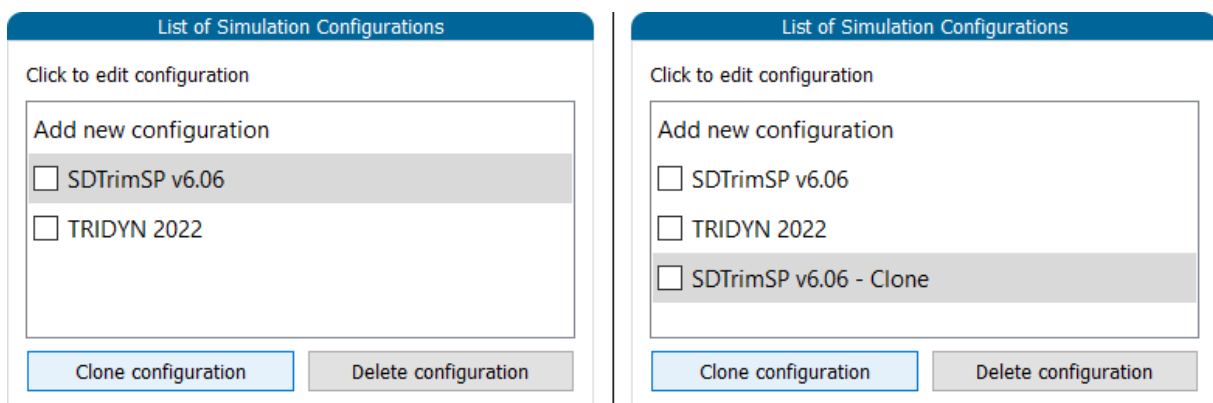


Figure 2: Duplicate a simulation configuration by selecting the simulation configuration and pressing the *Clone configuration* button (left). A new simulation configuration will appear (right).

To delete a simulation configuration, the desired simulation configuration has to be selected from the *List of Simulation Configurations* and the *Delete configuration* button must be pressed as shown in Figure 3. Simulation configurations can only be deleted, if there are no unsaved changes inside of them and they are not currently running a simulation.

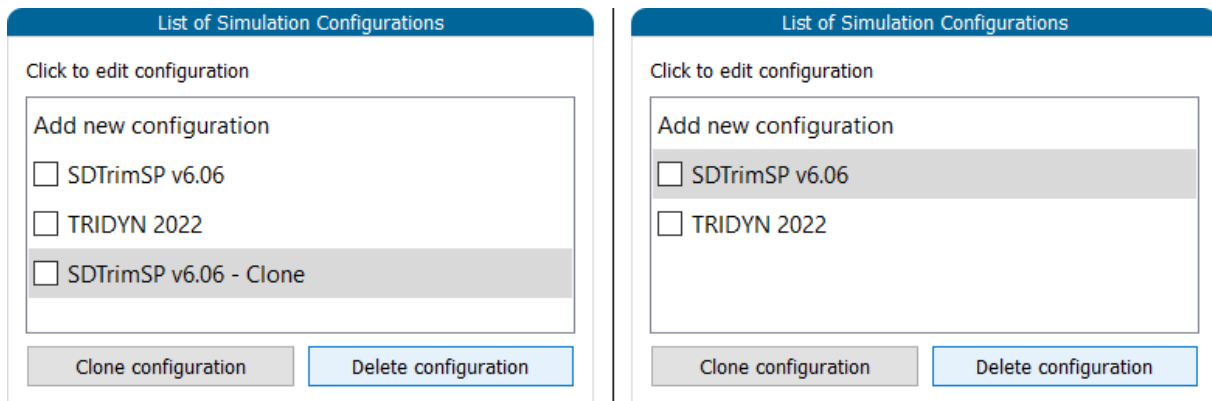


Figure 3: Delete a simulation configuration by selecting the simulation configuration and pressing the Delete configuration button (left). The selected simulation configuration is deleted (right).

a. Configuration for SDTrimSP

In Figure 4 the simulation configuration of an exemplary configuration for SDTrimSP version 6.06 for Windows is shown. In the *Configurations* tab a *Configuration title* of “SDTrimSP v6.06” was chosen and “SDTrimSP” was selected as *Simulation program*. The *Simulation folder* path should point to the root folder of SDTrimSP, which has a directory view similar to the left side Figure 5. The *Simulation binary* path should point to the executable file of SDTrimSP, which can be located in the *bin/windows.** folder with respect to the root folder of the simulation as displayed on the right side of Figure 5. Clicking the *Save configuration* button will add a new tab with the corresponding *Configuration title* and extend the *List of Simulation Configurations*.

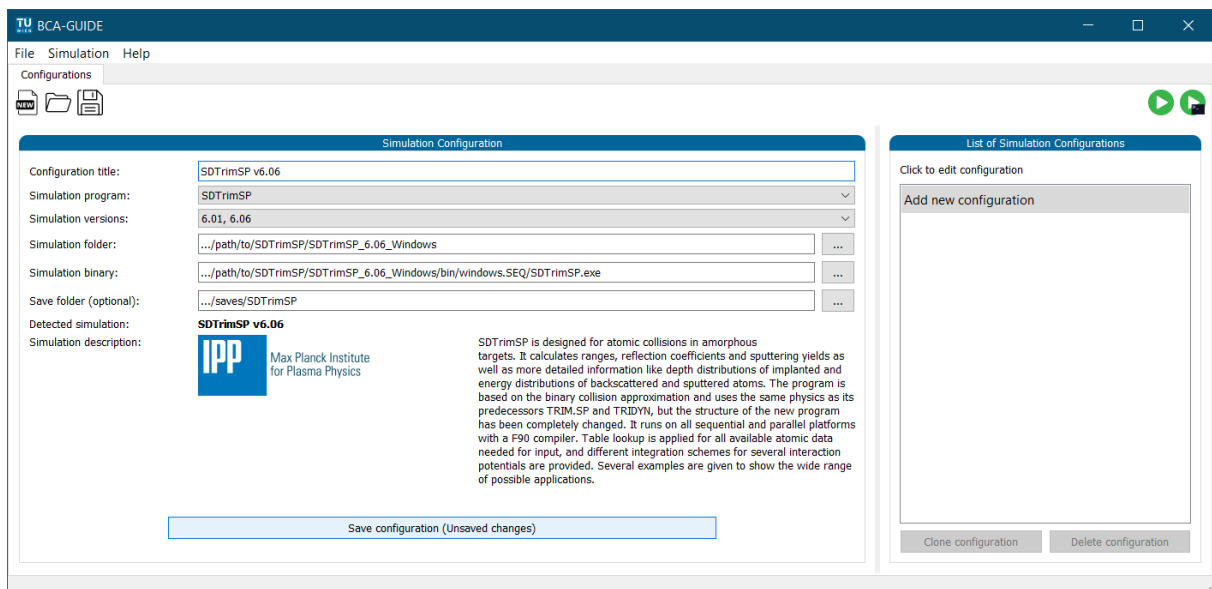


Figure 4: Example of a simulation configuration for SDTrimSP version 6.06 for Windows.

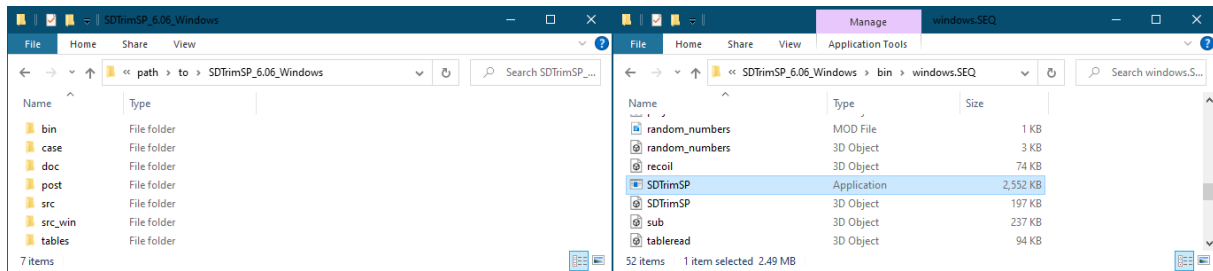


Figure 5: Directory view of the Simulation folder (left) and the Simulation binary (right) for SDTrimSP version 6.06 for Windows.

b. Configuration for TRIDYN 2022

In Figure 6 the simulation configuration of an exemplary configuration for TRIDYN 2022 for Windows is shown. In the *Configurations* tab a *Configuration title* of “TRIDYN 2022” was chosen and “TRIDYN2022” was selected as *Simulation program*. The *Simulation folder* path should point to the root folder of TRIDYN, which has a directory view similar to Figure 7. The *Simulation binary* path should point to the executable file of TRIDYN 2022, which is located in the *Simulation folder*. Clicking the *Save configuration* button will add a new tab with the corresponding *Configuration title* and extend the *List of Simulation Configurations*.

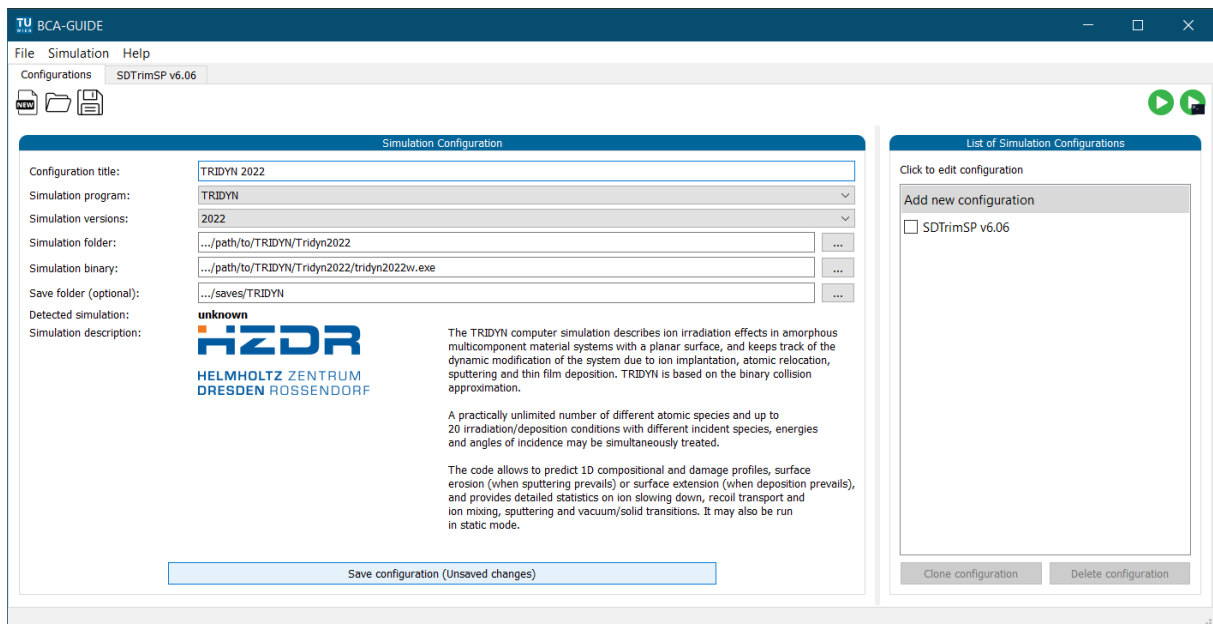


Figure 6: Example of a simulation configuration for TRIDYN 2022 for Windows.

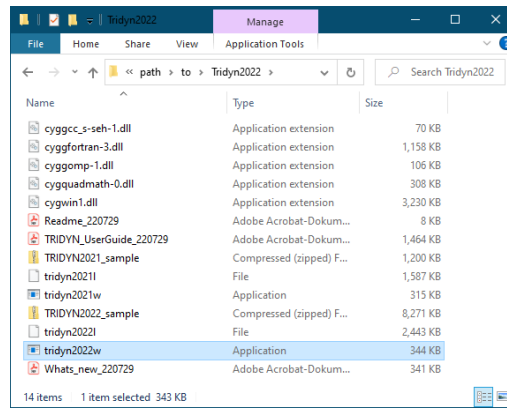


Figure 7: Directory view of the Simulation folder and the Simulation binary for TRIDYN 2022 for Windows. The Simulation binary is located directly in the Simulation folder by default.

3. Perform a simulation

Each simulation configuration provides both an interface to easily edit input parameters for the simulation, as well as an overview of the output files and a graphical preparation of them. By clicking on a tab, created by a previously set up simulation configuration, these interfaces can be accessed. In order to ensure uniformity, the interfaces for all simulation programs are structured similarly. Therefore, the interface structure of SDTrimSP is described in detail below.

At the top of every simulation configuration tab, a menu bar as shown in Figure 8 provides general operations:

- Reset: Deselects an active project and resets all input fields. A warning message will be displayed if the project has been modified and the changes are not saved.
- Open: Opens an already existing simulation input file, which was either created by the GUI, by another program or by hand. SDTrimSP simulation input files are named *tri.inp*, TRIDYN simulation input files end with **.in*. Simulation input files can only be opened with their corresponding simulation program. The directory name of the simulation input file will be displayed as *Project*. If input fields can not be filled with the values from the input file, they will be assigned default values provided by the GUI and marked in a red color.
- Import: Allows to load input files of other simulation programs. This feature is discussed in detail in chapter Copy the simulation to a different simulation program.
- Save: Generates the simulation input files depending on the provided parameters. In addition, an internally used universal simulation input file *input.json* will be created that is used for the importing process. The file dialog for saving the input file will start in the directory specified by *Save Folder* of the simulation configuration as shown in Figure 1.
- Settings: Opens the *Configurations* tab and selects the opened simulation configuration.
- Run: Starts the simulation program with the defined input parameters. The progress will be visualized in the progress bar. While a simulation is running, an hourglass will be displayed in the simulation configuration tab next to its name.
- Abort: This button is only active if a simulation is running and will abort the simulation.
- Run detached: Starts the simulation program with the defined input parameters in an external command window.
- Project: Displays the directory name of the simulation input file. This directory can be opened with the button next to the project name.
- Documentation: Opens the documentation of the simulation program

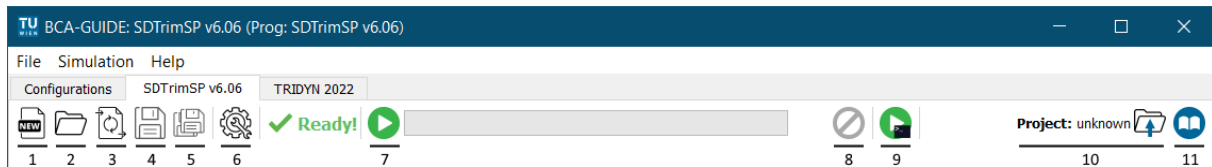


Figure 8: Menu bar for each simulation configuration. Actions are from left to right: 1.Reset, 2.Open, 3.Import, 4.Save, 5.Save As, 6.Settings, 7.Run, 8.Abort, 9.Run detached, 10.Project, 11.Documentation.

Beneath the menu bar, five configuration specific tabs are located, which divide the interfaces into displaying parameters and executing the simulation and are as follows:

a. Simulation setup

The *Simulation setup* tab allows easy editing of the input files in a structured layout and looks similar to Figure 9. For every simulation program the most used input parameters can be modified, which are divided into three groups: *Beam Settings*, *Target Settings* and *Simulation Settings*. The possible input fields depend on the simulation program but should mostly include common parameters for all simulations. Default values relate to the default values used by the individual BCA codes themselves.

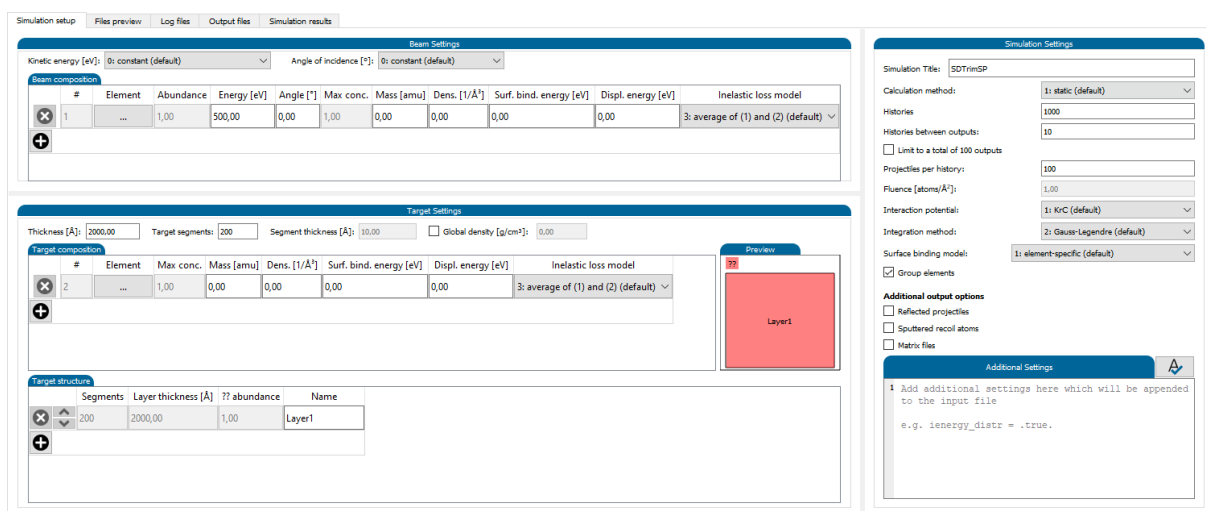


Figure 9: Empty simulation setup layout for a SDTrimSP simulation.

The *Beam Settings* section lists all beam related input parameters. In the *Beam composition* table, every element of the beam can be modified. The *Target Settings* section contains target related input parameters as well as the target structure. In the *Target composition* table, every element of the target can be modified. The *Target structure* table allows build a layered target with the elements specified in the target composition. A preview visualizes the structuring of the target. The *Simulation Settings* section lists general input parameters for the simulation. Parameters that are not selectable in the GUI are changed by entering them in the *Additional Settings*, where their validity is automatically checked. If invalid parameters are detected, an error message will be displayed and the input box will be outlined with a red border.

b. Files preview

The *Files preview* tab displays the input files that will be created. Figure 10 shows an exemplary simulation. The previews are read-only, therefore can not be edited. If no target structure is present, the layer file will not be created, which will be indicated in the preview.

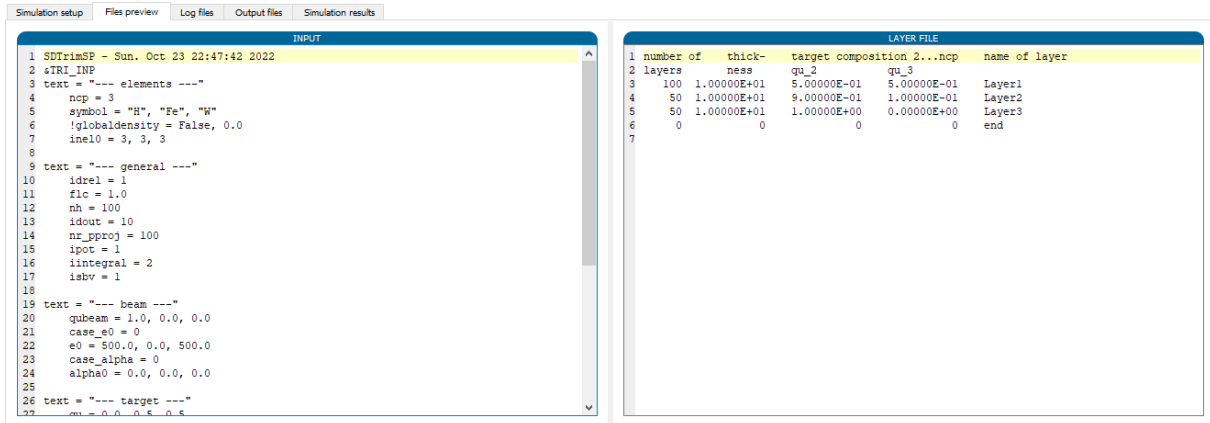


Figure 10: File preview for an exemplary SDTrimSP simulation.

c. Log files

After the simulation is started, the *Log files* tab shows the console messages of the simulation program, as can be seen in Figure 11. If an error occurs while running the simulation, error messages can often be found in this tab and used for adjusting the input parameters.

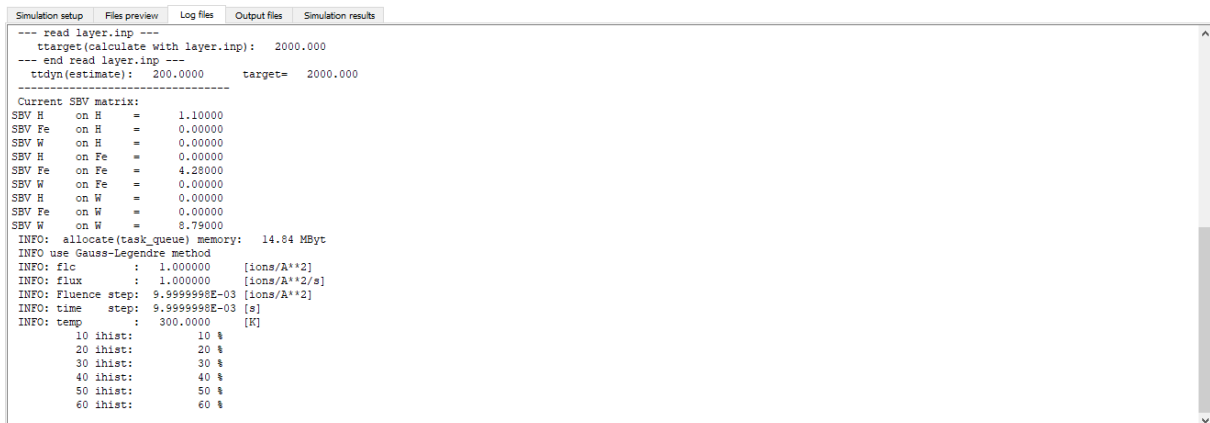


Figure 11: Log files of an exemplary simulation.

d. Output files

All output files of the simulation program are listed in the *Output files* tab as shown in Figure 12. By selecting the desired output file from the *List of files* a preview of the file will be displayed. The number of previewed lines can be adjusted. While the simulation is running, the file list and preview of the selected file will periodically refresh.

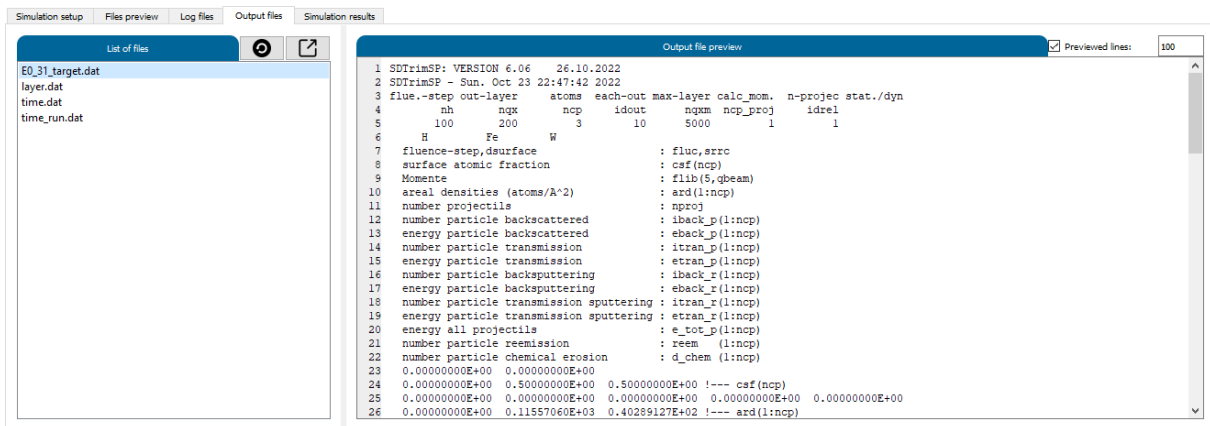


Figure 12: List and preview of output files of an exemplary simulation.

e. Simulation results

The *Simulation results* tab shown in Figure 13 displays data and plots generated from the output files of the simulation. Depending on the simulation, these may be updated periodically while a simulation is running or may not be available until the simulation is complete. Possible plots differ depending on the simulation parameters and selected output options in the *Simulation setup* tab. General data and available plots can be found in the left list. After selecting the desired plot, it will be displayed on the right side. By clicking the save icon, the data used in the active plot can be saved externally.

Attention: While SDTrimSP lists the mean projectile energy loss in the statistics overview, TRIDYN outputs the mean deposited energy by projectiles and recoils. This effects the nuclear-related value. For SDTrimSP, it is the summed energy transferred from the projectile to target atoms in the elastic collisions. In TRIDYN, however, this value is the sum of the residual energies below the cut-off energy of the atoms involved in the collision cascade.

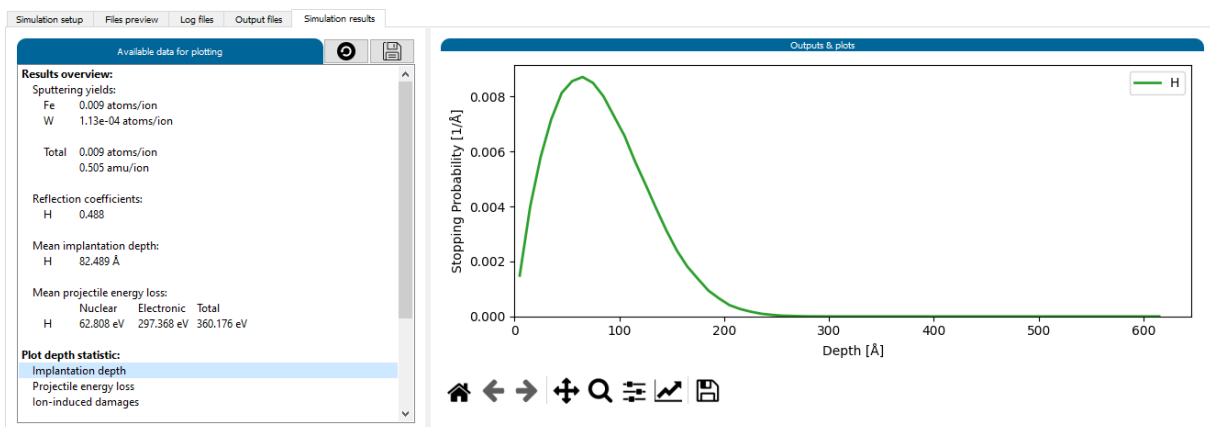


Figure 13: Simulation results of an exemplary simulation. Available plots can be selected from the list (left) and will be displayed on the right side.

4. Perform multiple simulations

The GUI is designed to set up multiple simulation configurations simultaneously. Therefore, multiple simulations can be edited without opening the GUI multiple times. If multiple simulations are set up, they can be started together. To avoid starting each simulation individually, the desired simulations can be selected in the *Configurations* tab in the *List of Simulation Configurations* as shown in Figure 14. The selected simulations can be started simultaneously by clicking the run button.

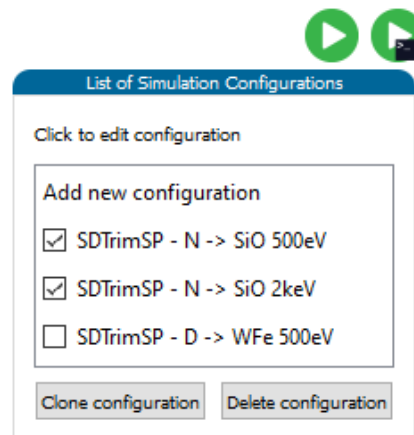



Figure 14: Selection of several simulations that can be started simultaneously.

5. Copy the simulation to a different simulation program

One main feature of the GUI is to allow transforming the input files of one BCA code into compatible input files for another code. The import functionality was developed for quick transforming simple input files between codes, therefore transforming more complex setups may fail. Therefore, the adapted parameters for the new code should always be checked after importing. This can be achieved by following steps, which are shown for a scenario, where input files for SDTrimSP are present and should be transformed into TRIDYN input files:

1. Set up simulation configurations for both simulations; the simulation where the input file can be read (here SDTrimSP) and the simulation where the simulation parameters should be transferred to (here TRIDYN).
2. Select the simulation configuration tab where the input file can be read (here SDTrimSP) and open the input file. This fills the parameter fields of the GUI.
3. Without editing, click the save button. This will create a new file (*input.json*) in the directory of the input file, which is internally used as a universal simulation input file.
4. Select the simulation configuration tab where the simulation parameters should be transferred to (here TRIDYN) and click the import button (). Navigate to the directory where the input file is located and select the file *input.json*. A new file dialog will be displayed, where a save folder must be selected. After selecting the new save folder, the parameter fields of the GUI are filled with the copied values. Since different simulation programs have different input fields, some parameters might not have values assigned to them. For these parameters, default values of the GUI are used, and the affected input fields are marked in a red color.
5. Without editing, click the save button. This will create the necessary input files for the simulation.

Additional settings which are specific for one simulation program will not be changed. Since the simulation program has changed, these values might result in errors and thus have to be checked individually.

6. Internal structure

To create a clear structure the python code was split into several files and directories. All necessary files of the GUI are located in the root folder of the GUI. A detailed directory structure of the GUI is given in Table 4, where each file or folder is listed with a short description. The three main executable files are marked in green.

Graphic files of are saved in the *icons* folder; however, they are not used directly by the GUI, instead they need to be compiled into the [resources.py](#) file. This is achieved by listing all graphics in the [resources.qrc](#) file and compiling them with the command

```
>> pyrrcc5 -o resources.py resources.qrc
```

The *Containers*, *Pages*, *TableWidgets* and *Utility* folders combined with the **.py* files in the root folder contain the general Python code of the GUI while the *Simulations* folder contains simulation specific code, which is responsible for different simulation programs. Adding new simulation programs to the GUI requires an additional file in this directory as explained in chapter [Add a new simulation program](#).

The *saves* folder contains a *config* folder, that stores created simulation configurations. Upon closing the GUI, the [autosave.json](#) file contains all active simulation configurations the user has entered. For each simulation program, there exists another directory inside the *saves* folder, named after the simulation program and is used as default save folder for each simulation run under this program.

The *setup* folder contains scripts for compiling the Python code into an executable file. A detailed description of how to create executable files is given in chapter [Requirements for executable file](#)

By launching the corresponding executable file, the GUI can be easily run. For Windows operating systems the [BCA_GUIDE_windows.exe](#) and for Linux operating systems the [BCA_GUIDE_linux](#) starts the application. Tested operating systems for the executable are listed in Table 3.

Windows	Linux
Windows 7	Linux Mint 20.2
Windows 10	Ubuntu 22.04
Windows 11	

Table 3: List of operating systems where the executable was tested on.

Compiling the executable file.

❏ BCA-GUIDE	root folder of GUI
↳ 📄 BCA-GUIDE_linux	executable for Linux
↳ 📄 BCA-GUIDE_windows.exe	executable for Windows
↳ 📄 BCA-GUIDE_manual.pdf	this manual
↳ 📄 Dialogs.py	contains manual, preferences and about dialogs
↳ 📄 GlobalConf.py	global variables
↳ 📄 License.txt	GNU general public license
↳ 📄 main.py	Python executable
↳ 📄 MainWindow.py	main window of the GUI
↳ 📄 README.md	short description
↳ 📄 resources.qrc	list of resources
↳ 📄 resources.py	compiled resources; created from <i>resources.qrc</i>
↳ 📄 Styles.py	styles for various objects
↳ ❏ Containers	contains classes that are used for storing data
↳ 📄 Arguments.py	stores input parameters; routine for saving and loading <i>input.json</i> files
↳ 📄 Compound.py	stores a compound
↳ 📄 Element.py.....	stores an element or isotope and list of elements
↳ 📄 GlobalDensity.py.....	routine to evaluate densities if a global density is given
↳ 📄 SimulationConfiguration.py	stores a simulation configuration; routine for saving and loading <i>config.json</i> files
↳ ❏ Pages	contains main pages of the GUI
↳ 📄 ConfigurationPage.py.....	page for configuring the simulation configuration
↳ 📄 ProgramPage.py.....	page for general simulation editing and displaying
↳ ❏ Simulations	contains specific simulations pages
↳ 📄 SimulationsList.py	lists all available simulation pages
↳ 📄 Simulations.py	basic functionality for all simulations
↳ 📄 SDTrimSP.py.....	specific functionality for SDTrimSP
↳ 📄 TRIDYN2022.py	specific functionality for TRIDYN2022
↳ ❏ Simulation Programs.....	empty folder where all BCA codes can be stored
↳ ❏ TableWidgets	contains widgets
↳ 📄 CustomTable.py	extends the PyQt5 table widgets
↳ 📄 CompTable.py	extends <i>CustomTable.py</i> ; used for beam and target table
↳ 📄 TargetTable.py	extends <i>CustomTable.py</i> ; used for the target structure
↳ 📄 PeriodicTable.py	periodic table window
↳ 📄 TargetPreview.py	widget for preview of the target structure
↳ 📄 CompoundList.py	widget for possible compound list

Table 4a: Detailed file structure of the GUI with short description for each file. Main executable files are marked in green.

BCA-GUIDE	root folder of GUI
↳ Utility	contains general functions
↳ Dialogs.py	routine for file dialog, message box and download dialog
↳ Functions.py	short, general used functions
↳ Indexing.py	routines for counting objects
↳ Layouts.py	extends the PyQt5 layouts
↳ ModifyWidget.py	routines for modifying widgets
↳ icons	contains graphic files
↳ source	contains source files for graphics
↳ *.ai, *.svg	source files for graphics are Adobe Illustrator files (*.ai) or scalable vector graphic files (*.svg)
↳ *.png	icons are stored as portable network graphic files (*.png)
↳ tu_logo.ico	GUI logo
↳ saves	contains save files
↳ config	contains simulation configuration save files
↳ autosave.json	created when GUI is closed to store active simulation configurations
↳ PROG_NAME	contains save files for one specific simulation program; the folder name depends on the simulation program
↳ setup	contains scripts to make an executable file of the GUI
↳ pyinstaller.py	routine that uses PyInstaller for compiling the GUI

*Table 4b: (continuation) Detailed file structure of the GUI with short description for each file.
Main executable files are marked in green.*

7. Add a new simulation program

To add a new simulation program, an additional Python file needs to be added into the *Simulations* folder of the GUI. The Python file must contain two classes, the *SimulationInput* and *SimulationOutput* class. The GUI will fail upon startup if either class is missing. A detailed descriptions of each property and function of both classes is listed in the *Simulations.py* file.

a. *SimulationInput* class

The *SimulationInput* class must inherit from the base class *SimulationsInput*, which is defined in the *Simulations.py* file located in the same directory. This class contains simulation specific parameters and functions, and is responsible for the specific layout of the simulation setup tab. General properties of this class have following names and functionality:

- Name: name of the simulation program; will be displayed as name of the *Simulation program* in the drop-down menu in the *Configurations* tab; must be unique
- Description: detailed description of the simulation program; will be displayed as *Simulation description* in the *Configurations* tab
- Logo: path to icon for the simulation program; will be displayed as *Simulation description* in the *Configurations* tab and the *About* dialog
- About: information about the simulation program; will be displayed in the *About* dialog
- SaveFolder: name of default save folder for this simulation program; the save folder will be created inside the *saves* directory of the GUI
- InputFilename: name of the input file for the simulation program; allows wildcard specifiers (*) in the file name
- LayerFilename: name of the input layer file for the simulation program; allows wildcard specifiers (*) in the file name
- ExampleAdditionalSetting: example for an additional setting; will be displayed in the *Additional Settings* input as an example
- SkipList: list of file names that will not be shown in the *List of files* of the *Output files* tab
- OutputTooltips: tooltips for files listed in the *List of files* of the *Output files* tab
- InputParameters: input parameters supported in the simulation program
- CompoundList: list of compounds supported in the simulation program
- GroupElements: if same elements should be grouped in beam and target
- MaxComponents: maximum number of different components
- HLBeamSettings: references custom beam-settings layout class
- HLTargetSettings: references custom target-settings layout class
- VISimulationSettings: references custom simulation-settings layout class
- CompRowBeamSettings: references custom beam-row layout class
- CompRowTargetSettings: references custom target-row layout class

Possible elements supported in the simulation program are stored as an **Elements** class, which is defined in the *Containers/Element.py* file, in the `element-data` property.

General functions of this class have following names and functionality:

- `update(folder, binary)`: called on startup; used if some parameters need to be updated dynamically depending on the simulation program folder or simulation program binary
- `nameInputFile()`: returns name of input file
- `nameLayerFile()`: returns name of input layer file
- `makeInputFile()`: returns the contents of the input file
- `makeLayerFile()`: returns the contents of the input layer file
- `loadFiles(folder)`: loads all input files in the provided folder
- `cmd(binary, save-folder, input-file)`: returns command to execute the simulation program with the desired input
- `getProgress(save-folder, process-log)`: returns the progress of the simulation while it is running

b. SimulationOutput class

The **SimulationOutput** class must inherit from the base class **SimulationsOutput**, which is defined in the *Simulations.py* file located in the same directory. This class contains simulation specific parameters related to evaluating and displaying the output files of a simulation. General properties of this class have following names and functionality:

- `HIPlot`: references custom layout class used for manipulating the plot

General functions of this class have following names and functionality:

- `reset()`: resets the plot and list of available plots
- `plotFct(plot, plot-args)`: calls a plotting routine with the plot arguments
- `listParameters(save-folder)`: lists all available plots

Bibliography

- [1] P. Szabo, D. Weichselbaum, H. Biber, C. Cupak, A. Mutzke, R. Wilhelm and F. Aumayr, "Graphical user interface for SDTrimSP to simulate sputtering, ion implantation and the dynamic effects of ion irradiation," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 522, pp. 47-53, 2022.
- [2] W. Möller and W. Eckstein, "Tridyn - A TRIM simulation code including dynamic composition changes," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 2, pp. 814-818, 1984.
- [3] A. Mutzke, R. Schneider, W. Eckstein, R. Dohmen, K. Schmid, U. von Toussaint and G. Bandelow, "SDTrimSP Version 6.00," Max-Planck-Institut, 2019.

List of figures and tables

Figure 1: Configurations interface that is displayed on startup and allows for editing simulation configurations.....	5
Figure 2: Duplicate a simulation configuration by selecting the simulation configuration and pressing the Clone configuration button (left). A new simulation configuration will appear (right).	6
Figure 3: Delete a simulation configuration by selecting the simulation configuration and pressing the Delete configuration button (left). The selected simulation configuration is deleted (right).....	7
Figure 4: Example of a simulation configuration for SDTrimSP version 6.06 for Windows.	7
Figure 5: Directory view of the Simulation folder (left) and the Simulation binary (right) for SDTrimSP version 6.06 for Windows.....	8
Figure 6: Example of a simulation configuration for TRIDYN 2022 for Windows.....	8
Figure 7: Directory view of the Simulation folder and the Simulation binary for TRIDYN 2022 for Windows. The Simulation binary is located directly in the Simulation folder by default.	9
Figure 8: Menu bar for each simulation configuration. Actions are from left to right: 1.Reset, 2.Open, 3.Import, 4.Save, 5.Save As, 6.Settings, 7.Run, 8.Abort, 9.Run detached, 10.Project, 11.Documentation.	11
Figure 9: Empty simulation setup layout for a SDTrimSP simulation.	11
Figure 10: File preview for an exemplary SDTrimSP simulation.	12
Figure 11: Log files of an exemplary simulation.	12
Figure 12: List and preview of output files of an exemplary simulation.	13
Figure 13: Simulation results of an exemplary simulation. Available plots can be selected from the list (left) and will be displayed on the right side.	13
Figure 14: Selection of several simulations that can be started simultaneously.	14
 Table 1: Tested Python 3 versions and operating systems.	3
Table 2: Necessary Python packages with a corresponding tested version number.	4
Table 3: List of operating systems where the executable was tested on.	4
Table 4: Detailed file structure of the GUI with short description for each file. Main executable files are marked in green.....	17