SIGRun

Generated by Doxygen 1.8.15

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

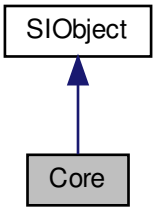
Class Documentation

4.1 Core Class Reference

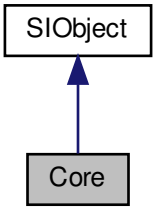
namespace shortening for python object integration

`#include <Core.hpp>`

Inheritance diagram for Core:



Collaboration diagram for Core:



Public Member Functions

- ~Core ()
  destructor
- void start ()
  entry point of core SIGRun initialization
- void stop ()
  exit SIGRun core

Protected Member Functions

- Core ()
  constructor
- void retrieve_available_plugins (std::unordered_map< std::string, std::shared_ptr< bp::object >> &plugins, const std::string &plugin_path)
  retrieve all available plugins before launching SIGRun environment

Friends

- class SIGRun
- class SIGRunTest
- class SIGRunCoreTest

Additional Inherited Members

4.1.1 Detailed Description

namespace shortening for python object integration

SIObject Central Core class registered as SIObject

This class initiates all subsystems required for the SIGRun environment. This class collects all available pulgins first. Second, it launches the SI context and other subsystems. This class is registered as SIObject meta type. This class conctructor is declared private to disable use by external application programmers. Therefore, the friend keyword is used to internally expose the class.

Definition at line 28 of file Core.hpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ~Core()

```
Core::~Core ( )
```

destructor

Shut down the SIGRun environment.

Definition at line 25 of file Core.cpp.

4.1.2.2 Core()

```
Core::Core ( )  [protected]
```

constructor

Constructor which registers instance as an SIObject. Specify, which Logging capabilities are desired.

Definition at line 14 of file Core.cpp.

4.1.3 Member Function Documentation

4.1.3.1 retrieve_available_plugins()

```
void Core::retrieve_available_plugins (
            std::unordered_map< std::string, std::shared_ptr< bp::object >> & plugins,
            const std::string & plugin_path )  [protected]
```

retrieve all available plugins before launching SIGRun environment

Load all plugins in the plugin path of the SIGRun environment.

Parameters

plugins	a mutable reference to a std::unordered map with std::string as key and a std::shared_ptr of boost::python::objects as values which is the out parameter
plugin_path	a std::string which contains the path to the root folder of all plugin files

See also

Scripting::Scripting
PluginCollector::PluginCollector

Definition at line 76 of file Core.cpp.

4.1.3.2 start()

```
void Core::start ( )
```

entry point of core SIGRun initialization

Entry point of SIGRun's core which performs Plugin loading and initializes the SI Context.

Definition at line 36 of file Core.cpp.

4.1.3.3 stop()

```
void Core::stop ( )
```

exit SIGRun core

Initiate the shutdown of the SIGRun core.

Definition at line 60 of file Core.cpp.

4.1.4 Friends And Related Function Documentation

4.1.4.1 SIGRun

```
friend class SIGRun  [friend]
```

Definition at line 41 of file Core.hpp.

4.1.4.2 SIGRunCoreTest

```
friend class SIGRunCoreTest  [friend]
```

Definition at line 43 of file Core.hpp.

4.1.4.3 SIGRunTest

```
friend class SIGRunTest  [friend]
```

Definition at line 42 of file Core.hpp.

The documentation for this class was generated from the following files:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/Core.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/Core.cpp

4.2 IterableConverter Class Reference

```
#include <SuperEffect.hpp>
```

Public Member Functions

- template<typename Container >
  IterableConverter & from_python ()

Static Public Member Functions

- static void * convertible (PyObject *object)

  Check if PyObject is iterable.
- template<typename Container >
  static void construct (PyObject *object, bp::converter::rvalue_from_python_stage1_data *data)

  Convert iterable PyObject to C++ container type.

4.2.1 Detailed Description

Definition at line 10 of file SuperEffect.hpp.

4.2.2 Member Function Documentation

4.2.2.1 construct()

```
template<typename Container >
void IterableConverter::construct (
            PyObject * object,
            bp::converter::rvalue_from_python_stage1_data * data )  [static]
```

Convert iterable PyObject to C++ container type.

Container Concept requirements:

- Container::value_type is CopyConstructable.
- Container can be constructed and populated with two iterators. I.e. Container(begin, end)

Definition at line 23 of file SuperEffect.cpp.

4.2.2.2 convertible()

```
void * IterableConverter::convertible (
            PyObject * object )  [static]
```

Check if PyObject is iterable.

Definition at line 17 of file SuperEffect.cpp.

4.2.2.3 from_python()

```
template<typename Container >
IterableConverter & IterableConverter::from_python ( )
```

Note

Registers converter from a python interable type to the provided type.

Definition at line 9 of file SuperEffect.cpp.

The documentation for this class was generated from the following files:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/pysi/SuperEffect.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/pysi/SuperEffect.cpp

4.3 Log Class Reference

Log class serving as central logging functionality for easy logging data output.

```
#include <Log.hpp>
```

Public Types

- enum LOG_LEVEL {
  INFO_LEVEL = 0b00001, WARN_LEVEL = 0b00010, DEBUG_LEVEL = 0b00100, ERROR_LEVEL = 0b01000,
  UNDEFINED_LEVEL = 0b10000 }

  enum for log level selection modelled as a bitfield
- enum MODE { NONE = 0, CONSOLE = 1, FILE = 2 }

  enum for log mode selection modelled as a bitfield
- enum SHOW_TYPE {
  HIDDEN = 0, INFO = 1, WARN = 2, DEBUG = 4,
  ERROR = 8, UNDEFINED = 16 }

  enum for log show type selection modelled as a bitfield

Static Public Member Functions

- static void log (const std::string &what, int level, int logging_flags, const std::string &type, const std::string &file="", const std::string &func="", const std::string &line="")

  central logging function outputting log messages according to its params
- static void set_log_file_path (const std::string &path)
- static std::string log_level (int log_level)

  return the level of a log message as tag according to its id
- static std::string time ()

  return current system time with milliseconds precision

Static Public Attributes

- static std::string log_file_path = Log::PATH_DEFAULT
- static int SHOW = -1

  the integer variable containing which log messages are outputted based on their tag
- static bool __DEBUG__ = false

  the flag which is required to be set to true if the logging system is required to be used.

4.3.1 Detailed Description

Log class serving as central logging functionality for easy logging data output.

This class serves as the central knot for all output operations in terms of log messages. This class is a static class featuring no ctor or dtor. Shortcut macros make the access to this class logging functionality more easier.

See also

DEBUG(what, log_mode)
WARN(what, log_mode)
ERROR(what, log_mode)
INFO(what, log_mode)
UNDEFINED(what, log_mode)

Definition at line 181 of file Log.hpp.

4.3.2 Member Enumeration Documentation

4.3.2.1 LOG_LEVEL

```
enum Log::LOG_LEVEL
```

enum for log level selection modelled as a bitfield

The log level describes which tag is assigned to a log message.

Enumerator

INFO_LEVEL	
WARN_LEVEL	
DEBUG_LEVEL	
ERROR_LEVEL	
UNDEFINED_LEVEL	

Definition at line 206 of file Log.hpp.

4.3.2.2 MODE

```
enum Log::MODE
```

enum for log mode selection modelled as a bitfield

The log mode describes where a log message is outputted. A mode is ignored if it is not specified. Due to the enum being modelled as a bitfield, users can use the | operator to selectively enable modes for logging output. Example for enabling printing to stdout as well as to a file: int mode = CONSOLE | FILE;

Enumerator

NONE	
CONSOLE	
FILE	

Definition at line 224 of file Log.hpp.

4.3.2.3 SHOW_TYPE

```
enum Log::SHOW_TYPE
```

enum for log show type selection modelled as a bitfield

The log show type describes which log messages are outputted based on their tags. Tags which are not specified are ignored. Due to the enum being modelled as a bitfield, users can use the | operator to selectively enable tags for logging output. Example for enabling DEBUG and WARN tags without the INFO tag: int loglevel = WARN | DEBUG

Enumerator

HIDDEN	
INFO	
WARN	
DEBUG	
ERROR	
UNDEFINED	

Definition at line 239 of file Log.hpp.

4.3.3 Member Function Documentation

4.3.3.1 log()

```
void Log::log (
            const std::string & what,
```

```
            int level,
            int logging_flags,
            const std::string & type,
            const std::string & file = "",
            const std::string & func = "",
            const std::string & line = "" )  [static]
```

central logging function outputting log messages according to its params

This is the central logging function of SIGRun. It requires to be called from class which are registered as SIObject. The parameters of this function, besides what (log message), configure the way the message is outputted. This static method is easier accessible via the shortcut macros.

Parameters

what	a std::string containing the log message
level	an integer containing the id of the desired tag
logging_flags	an integer containing where the log message is to be outputted
type	a std::string containing the description of the functions caller via an SIObject
file	a std::string containing the name of the file in which the log call is implemented
func	a std::string containing the name of the function in which the log call was issued
line	a std::string containing the number of the line of the file in which the log call is implemented

See also

DEBUG(what, log_mode)
WARN(what, log_mode)
ERROR(what, log_mode)
INFO(what, log_mode)
UNDEFINED(what, log_mode)
SIObject

Definition at line 37 of file Log.cpp.

4.3.3.2 log_level()

```
std::string Log::log_level (
            int log_level )  [static]
```

return the level of a log message as tag according to its id

Retrieves the level of a log message according to the value of the parameter which is compared to the Log::LOG_LEVEL enum/bitfield.

Parameters

log_level	an integer containing the id of the desired tag

Returns

a std::string which contains a human readable version of the desired tag

Definition at line 112 of file Log.cpp.

4.3.3.3 set_log_file_path()

```
void Log::set_log_file_path (
            const std::string & path )  [static]
```

set the path of the file for logging output Set the value of the static variable log_file_path to the value of the given parameter to specifiy the file path of the log output.

Parameters

path	a std::string containing the desired file path for logging to files

Definition at line 98 of file Log.cpp.

4.3.3.4 time()

```
std::string Log::time ( )  [static]
```

return current system time with milliseconds precision

Compute current system time with milliseconds precision. Format the date data to yyyy-MM-dd hh:mm↩ :ss.<milliseconds>. Concatenate the date data to a std::string.

Returns

a std::string containing the formatted date data

Definition at line 138 of file Log.cpp.

4.3.4 Member Data Documentation

4.3.4.1 __DEBUG__

```
bool Log::__DEBUG__ = false  [static]
```

the flag which is required to be set to true if the logging system is required to be used.

This flag is the center of enabling (**DEBUG** is set to true) or disabling (**DEBUG** is set to false) the entire logging system.

Definition at line 261 of file Log.hpp.

4.3.4.2 log_file_path

```
std::string Log::log_file_path = Log::PATH_DEFAULT  [static]
```

actual path to logfile

Definition at line 199 of file Log.hpp.

4.3.4.3 SHOW

```
int Log::SHOW = -1  [static]
```

the integer variable containing which log messages are outputted based on their tag

This integer variable regulates which log messages are outputted, according to their tags.

Definition at line 254 of file Log.hpp.

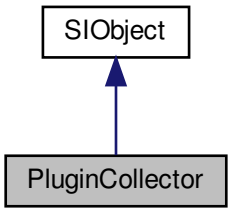
The documentation for this class was generated from the following files:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/log/Log.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/log/Log.cpp

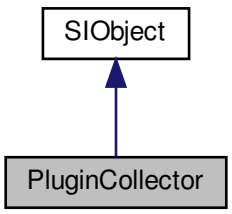
4.4 PluginCollector Class Reference

```
#include <PluginCollector.hpp>
```

Inheritance diagram for PluginCollector:



Collaboration diagram for PluginCollector:



Public Member Functions

- PluginCollector ()
- ~PluginCollector ()=default
- void collect (const std::string &rel_path, std::vector< std::string > &files)

Additional Inherited Members

4.4.1 Detailed Description

Definition at line 9 of file PluginCollector.hpp.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 PluginCollector()

```
PluginCollector::PluginCollector ( )  [inline]
```

Definition at line 12 of file PluginCollector.hpp.

4.4.2.2 ~PluginCollector()

```
PluginCollector::~PluginCollector ( )  [default]
```

4.4.3 Member Function Documentation

4.4.3.1 collect()

```
void PluginCollector::collect (
            const std::string & rel_path,
            std::vector< std::string > & files )
```

Definition at line 8 of file PluginCollector.cpp.

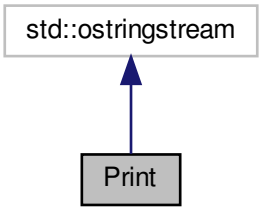
The documentation for this class was generated from the following files:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/PluginCollector.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/PluginCollector.cpp

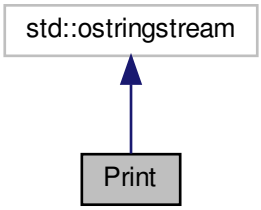
4.5 Print Class Reference

`#include <Print.hpp>`

Inheritance diagram for Print:



Collaboration diagram for Print:



Public Member Functions

- Print ()=default
- ~Print ()

Static Public Member Functions

- template<typename T >
  static void print (const std::vector< std::vector< T >> &v)
- template<typename T >
  static void print (const std::vector< T > &v)
- template<typename T1 , typename T2 >
  static void print (const std::map< T1, T2 > &map)
- template<typename T >
  static void print (const T &arg)

4.5.1 Detailed Description

Definition at line 14 of file Print.hpp.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Print()

```
Print::Print ( )  [default]
```

4.5.2.2 ~Print()

```
Print::~Print ( )  [inline]
```

Definition at line 19 of file Print.hpp.

4.5.3 Member Function Documentation

4.5.3.1 print() [1/4]

```
template<typename T >
static void Print::print (
            const std::vector< std::vector< T >> & v )  [inline], [static]
```

Definition at line 26 of file Print.hpp.

4.5.3.2 print() [2/4]

```
template<typename T >
static void Print::print (
            const std::vector< T > & v )  [inline], [static]
```

Definition at line 51 of file Print.hpp.

4.5.3.3 print() [3/4]

```
template<typename T1 , typename T2 >
static void Print::print (
            const std::map< T1, T2 > & map )  [inline], [static]
```

Definition at line 71 of file Print.hpp.

4.5.3.4 print() [4/4]

```
template<typename T >
static void Print::print (
            const T & arg )  [inline], [static]
```

Definition at line 82 of file Print.hpp.

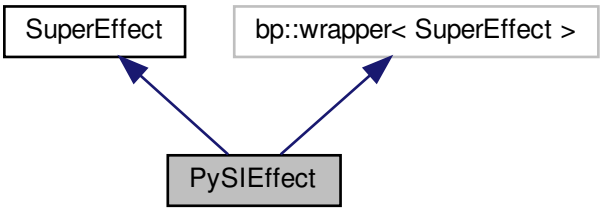
The documentation for this class was generated from the following files:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/debug/Print.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/debug/Print.cpp

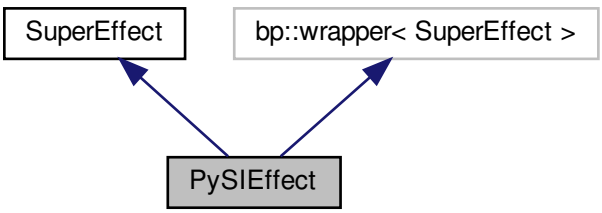
4.6 PySIEffect Class Reference

`#include <SuperEffect.hpp>`

Inheritance diagram for PySIEffect:



Collaboration diagram for PySIEffect:



Public Member Functions

- int on_enter (bp::object &other) override
- int on_continuous (bp::object &other) override
- int on_leave (bp::object &other) override

4.6.1 Detailed Description

Definition at line 40 of file SuperEffect.hpp.

4.6.2 Member Function Documentation

4.6.2.1 on_continuous()

```
int PySIEffect::on_continuous (
            bp::object & other )  [override], [virtual]
```

Implements SuperEffect.

Definition at line 50 of file SuperEffect.cpp.

4.6.2.2 on_enter()

```
int PySIEffect::on_enter (
            bp::object & other )  [override], [virtual]
```

Implements SuperEffect.

Definition at line 45 of file SuperEffect.cpp.

4.6.2.3 on_leave()

```
int PySIEffect::on_leave (
            bp::object & other )  [override], [virtual]
```

Implements SuperEffect.

Definition at line 55 of file SuperEffect.cpp.

The documentation for this class was generated from the following files:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/pysi/SuperEffect.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/pysi/SuperEffect.cpp

4.7 PythonInvoker Class Reference

`#include <PythonInvoker.hpp>`

Public Member Functions

- PythonInvoker ()
- ~PythonInvoker ()
- template<typename T >
  T invoke_extract_attribute (const bp::object &self, const std::string &attribute_name)
- template<typename T >
  void invoke_set_attribute (bp::object &self, std::string &attribute_name, T &value, bool is_pointer=false)
- template<typename T >
  T invoke_function (bp::object &self, const std::string &function_name, bp::object &other)

4.7.1 Detailed Description

Definition at line 10 of file PythonInvoker.hpp.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 PythonInvoker()

```
PythonInvoker::PythonInvoker ( )  [default]
```

4.7.2.2 ~PythonInvoker()

```
PythonInvoker::~PythonInvoker ( )  [default]
```

4.7.3 Member Function Documentation

4.7.3.1 invoke_extract_attribute()

```
template<typename T >
T PythonInvoker::invoke_extract_attribute (
            const bp::object & self,
            const std::string & attribute_name )  [inline]
```

Definition at line 17 of file PythonInvoker.hpp.

4.7.3.2 invoke_function()

```
template<typename T >
T PythonInvoker::invoke_function (
             bp::object & self,
             const std::string & function_name,
             bp::object & other )  [inline]
```

Definition at line 48 of file PythonInvoker.hpp.

4.7.3.3 invoke_set_attribute()

```
template<typename T >
void PythonInvoker::invoke_set_attribute (
             bp::object & self,
             std::string & attribute_name,
             T & value,
             bool is_pointer = false )  [inline]
```

Definition at line 32 of file PythonInvoker.hpp.

The documentation for this class was generated from the following files:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/PythonInvoker.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/PythonInvoker.cpp

4.8 RegionMask Class Reference

RegionMask class which stores a bit array used for true collision testing.

```
#include <RegionMask.hpp>
```

Public Member Functions

- RegionMask (int canvas_width, int canvas_height, const std::vector< glm::vec3 > &contour, const std::vector< glm::vec3 > &aabb)

  constructor of the RegionMask class
- RegionMask (const RegionMask &rm)

  copy constructor
- ~RegionMask ()

  default destructor
- int size () const

  retrieve the size of the mask datastructure
- void set_bit (int i)

  set the bit at index i of d_values to one/true
- void set_bit (const glm::vec3 &v)

  set the bit at point v to one/true in d_values
- void clear_bit (int i)

set the bit at index i of d_values to zero/false

- void clear_bit (const glm::vec3 &v)

  set the bit at point v of d_values to zero/false
- int width () const
- int height () const
- void move (const glm::vec2 &v)

  update the AABB relations according to desired translation of a parent Region
- bool operator[ ] (int i) const

  [] operator overloaded for returning the value of d_values at index i
- bool operator[ ] (const glm::vec3 &v) const

  [] operator overloaded for returning the value of d_values at point v

Friends

- class SIGRunRegionMaskTest

4.8.1 Detailed Description

RegionMask class which stores a bit array used for true collision testing.

Functionality
RegionMask class storing a bit array as std::vector<bool>. std::vector<bool> has a special implementation where its bool is stored in exactly one bit. See: https://en.cppreference.com/w/cpp/container/vector_bool This vector has the size of width * height of the AABB of the contour of the parent region. The array is filled with ones and zeroes according to a scanline algorithm. Every pixel which is part of the parent Region is set to one in that way. Others are left at 0. The array is relatively accessed according to the top left corner of that AABB.
Rationale:
The use of the AABB allows for creating a secondary coordinate system which is translated relatively to the parent coordinate system (canvas coordinate system). Therefore, each point which is to be tested with the mask is subtracted by the position vector of the AABB. In this way, that point in the canvas coordinate system is converted to the mask coordinate system. This leads to querying collosion occurrences relatively to the AABB. Through that, simple region translation does not require recomputation of the mask. Instead, the internal AABB is translated the same amount and the coordinate system conversion provides correct collision detection behaviour.

Definition at line 35 of file RegionMask.hpp.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 RegionMask() [1/2]

```
RegionMask::RegionMask (
            int canvas_width,
            int canvas_height,
            const std::vector< glm::vec3 > & contour,
            const std::vector< glm::vec3 > & aabb )
```

constructor of the RegionMask class

Constructor of the RegionMask class. Initializes all datastructures required to maintain a RegionMask for Collision↩ Detection according to parameters. Performs scanline algorithm for generation of the actual mask relatively to AABB of parent region.

Parameters

canvas_width	int containing the width of the canvas
canvas_height	int containing the height of the canvas
contour	constant reference to a std::vector object containing glm::vec3 objects containing all the points of the contour of the parent Region
aabb	constant reference to a std::vector object containing glm::vec3 objects containing the four points of the AABB of the parent Region

See also

d_canvas_width
d_canvas_height
d_tlc_aabb_x
d_tlc_aabb_y
d_brc_aabb_x
d_brc_aabb_y
d_width_aabb
d_height_aabb
d_values

Definition at line 28 of file RegionMask.cpp.

4.8.2.2 RegionMask() [2/2]

```
RegionMask::RegionMask (
            const RegionMask & rm )
```

copy constructor

Parameters

rm	the constant reference to a RegionMask object to be copied

See also

d_canvas_width
d_canvas_height
d_tlc_aabb_x
d_tlc_aabb_y
d_brc_aabb_x
d_brc_aabb_y
d_width_aabb
d_height_aabb
d_values

Definition at line 68 of file RegionMask.cpp.

4.8.2.3 ~RegionMask()

```
RegionMask::~RegionMask ( )
```

default destructor

Definition at line 86 of file RegionMask.cpp.

4.8.3 Member Function Documentation

4.8.3.1 clear_bit() [1/2]

```
void RegionMask::clear_bit (
            int i )
```

set the bit at index i of d_values to zero/false

Sets the bit at index i of d_values to zero or false according to a bounds check.

Parameters

i	int which contains the index of the bit to be set to zero/false in d_values

See also

d_values

Definition at line 153 of file RegionMask.cpp.

4.8.3.2 clear_bit() [2/2]

```
void RegionMask::clear_bit (
            const glm::vec3 & v )
```

set the bit at point v of d_values to zero/false

Sets the bit at point v of d_values to zero or false according to a bounds check. The bounds check is performed based on the actual index of the bit to set. The actual index is calculated according to AABB_WIDTH ∗ (v.y - A↩ ABB_TOP_LEFT_CORNER_Y) + v.x - AABB_TOP_LEFT_CORNER_X If the bounds check is negative, nothing happens.

Parameters

v	a constant reference to a glm::vec3 object containing the corresponding coordinates of the point to a bit of d_values which is to be set to zero or false.

See also

d_values
d_width_aabb
d_tlc_aabb_y
d_tlc_aabb_x

Definition at line 174 of file RegionMask.cpp.

4.8.3.3 height()

```
int RegionMask::height ( ) const
```

Returns

the height of the AABB of the parent Region

See also

d_height_aabb

Definition at line 197 of file RegionMask.cpp.

4.8.3.4 move()

```
void RegionMask::move (
            const glm::vec2 & v )
```

update the AABB relations according to desired translation of a parent Region

Use of RegionMasks occurs relatively to the AABB of its parent Region. Is the parent region moved / translated within the canvas, the AABB is also moved or translated. Therefore, the RegionMasks is upated according to that translation by storing the new translation parameters. In this way, the mask coordinate system is moved within the canvas coordinate system. So, after updating the AABB with the new translation values, the mask continues to function, due to its relative dependence on the AABB. Therefore, no recomputation is required.

Parameters

v	a constant reference to a glm::vec2 object containg the translation vector

Definition at line 259 of file RegionMask.cpp.

4.8.3.5 operator[]() [1/2]

```
bool RegionMask::operator[] (
            int i ) const
```

[] operator overloaded for returning the value of d_values at index i

Overloads the [] operator. Retrieves the bool value at index i of d_values.

Parameters

i	int containing the index

Returns

a bool containing whether the queried bit is set or not in d_values

See also

d_values

Definition at line 213 of file RegionMask.cpp.

4.8.3.6 operator[]() [2/2]

```
bool RegionMask::operator[] (
            const glm::vec3 & v ) const
```

[] operator overloaded for returning the value of d_values at point v

Overloads the [] operator. Retrieves the bool value at point v of d_values. The actual index is calculated according to AABB_WIDTH $*$ (v.y - AABB_TOP_LEFT_CORNER_Y) + v.x - AABB_TOP_LEFT_CORNER_X.

Parameters

v	a constant reference to a glm::vec3 object containing the corresponding coordinates of the point to a bit of d_values which is to be tested whether the queried bit is set or not in d_values.

Returns

a bool containing whether the queried bit is set or not in d_values

See also

d_values
d_width_aabb
d_tlc_aabb_x
d_tlc_aabb_y

Definition at line 237 of file RegionMask.cpp.

4.8.3.7 set_bit() [1/2]

```
void RegionMask::set_bit (
            int i )
```

set the bit at index i of d_values to one/true

Sets the bit at index i of d_values to one or true according to a bounds check.

Parameters

i	int which contains the index of the bit to be set to one/true in d_values

See also

d_values

Definition at line 115 of file RegionMask.cpp.

4.8.3.8 set_bit() [2/2]

```
void RegionMask::set_bit (
            const glm::vec3 & v )
```

set the bit at point v to one/true in d_values

Sets the bit at point v of d_values to one or true according to a bounds check. The bounds check is performed based on the actual index of the bit to be set. The actual index is calculated according to AABB_WIDTH $*$ (v.y - AABB_TOP_LEFT_CORNER_Y) + v.x - AABB_TOP_LEFT_CORNER_X If the bounds check is negative, the bit will be set to false.

Parameters

v	a constant reference to a glm::vec3 object containing the corresponding coordinates of the point to a bit of d_values which is to be set to one or true.

See also

d_values
d_width_aabb
d_tlc_aabb_y
d_tlc_aabb_x

Definition at line 136 of file RegionMask.cpp.

4.8.3.9 size()

```
int RegionMask::size ( ) const
```

retrieve the size of the mask datastructure

Retrieves the size of the mask datastructure. This datastructure is called d_values.

Returns

the size of d_values

See also

d_values

Definition at line 101 of file RegionMask.cpp.

4.8.3.10 width()

```
int RegionMask::width ( ) const
```

Returns

the width of the AABB of the parent Region

See also

d_width_aabb

Definition at line 187 of file RegionMask.cpp.

4.8.4 Friends And Related Function Documentation

4.8.4.1 SIGRunRegionMaskTest

```
friend class SIGRunRegionMaskTest  [friend]
```

Definition at line 106 of file RegionMask.hpp.

The documentation for this class was generated from the following files:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/region/RegionMask.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/region/RegionMask.cpp

4.9 RegionTransform Class Reference

RegionTransform class storing the relative translation, rotation and scale of a contour.

`#include <RegionTransform.hpp>`

Public Member Functions

- RegionTransform ()

  default constructor initializing instance variables to default values
- ~RegionTransform ()

  default destructor
- void update (const glm::vec2 &translation=glm::vec2(0, 0), float angle=0, float scale=1)

  central function to update transformation matrix with new, relative translation, relative rotation and absolute scale values
- const glm::mat3x3 & transform ()
- const glm::vec3 & operator[ ] (int index)

  overloading of [] operator

4.9.1 Detailed Description

RegionTransform class storing the relative translation, rotation and scale of a contour.

This class stores the relative translation, rotation and scale of a contour. The initial contour remains unchanged and change in one of those three aspects does mutate this transform but not the initial contour. The translation, rotation and scale are stored as a 3x3 transformation matrix. The transformation matrix is stored ROW MAJOR and requires LEFT pr PRE-Multiplication. Therefore, multiplications with points look such as : p $*$ T, where p is a point and T is the transformation matrix. Due to matrix multiplications being not commutative, T $*$ p will not yield desired results.

See also

d_translation
d_rotation
d_scale
d_transform
d_angle

Definition at line 34 of file RegionTransform.hpp.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 RegionTransform()

```
RegionTransform::RegionTransform ( )
```

default constructor initializing instance variables to default values

Default constructor. Initializes all matrix objects to identity matrices. Sets cumulative angle to 0

See also

d_transform
d_translation
d_rotation
d_scale \scale d_angle

Definition at line 17 of file RegionTransform.cpp.

4.9.2.2 ~RegionTransform()

```
RegionTransform::~RegionTransform ( )
```

default destructor

Default destructor.

Definition at line 30 of file RegionTransform.cpp.

4.9.3 Member Function Documentation

4.9.3.1 operator[]()

```
const glm::vec3 & RegionTransform::operator[] (
            int index )
```

overloading of [] operator

Overloading of [] operator. Makes it easier to use the transformation matrix stored in this class. This function returns a constant glm::vec3 reference which itself is subscriptable with the [] operator.

Parameters

index	an integer containing the index of the row of the transformation matrix to be retrieved.

Returns

a constant reference of glm::vec3 object containing the queried row of the transformation matrix

See also

d_transform

Definition at line 96 of file RegionTransform.cpp.

4.9.3.2 transform()

```
const glm::mat3x3 & RegionTransform::transform ( )
```

Returns

a constant reference to a glm::mat3x3 object containing the current transformation matrix

See also

d_transform

Definition at line 78 of file RegionTransform.cpp.

4.9.3.3 update()

```
void RegionTransform::update (
            const glm::vec2 & translation = glm::vec2(0, 0),
            float angle = 0,
            float scale = 1 )
```

central function to update transformation matrix with new, relative translation, relative rotation and absolute scale values

Updates translation matrix T, rotation matrix R, and scale matrix S according to the given parameters. Too small angle increments are ignored to save computations of required trigonometric functions. Computes a the new transformation matrix according to $T * R * S$.

Parameters

translation	a constant reference to a glm::vec2 datastructure containing the new, relative translation of the parent contour
angle	a float containing the new relative angle of the parent contour according to x-axis
scale	a float containing the new absolute scale factor of the contour

See also

d_translation
d_angle
d_rotation
d_scale
d_transform

Definition at line 50 of file RegionTransform.cpp.

The documentation for this class was generated from the following files:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/region/RegionTransform.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/region/RegionTransform.cpp

4.10 Scripting Class Reference

`#include <Scripting.hpp>`

Public Member Functions

- Scripting ()
- ~Scripting ()
- bp::object si_plugin (std::string &module_name, std::string &path, std::string &class_name)
- std::string load_plugin_source (const char *source)
- void load_class_names (std::vector< std::string > &classes, const std::string &path)
- bp::object import (const std::string &module, const std::string &path)

Friends

- std::ostream & operator<< (std::ostream &os, const Scripting &scripting)

4.10.1 Detailed Description

Definition at line 13 of file Scripting.hpp.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Scripting()

`Scripting::Scripting ( )`

Definition at line 11 of file Scripting.cpp.

4.10.2.2 ~Scripting()

```
Scripting::~Scripting ( )
```

Definition at line 21 of file Scripting.cpp.

4.10.3 Member Function Documentation

4.10.3.1 import()

```
bp::object Scripting::import (
            const std::string & module,
            const std::string & path )
```

Definition at line 95 of file Scripting.cpp.

4.10.3.2 load_class_names()

```
void Scripting::load_class_names (
            std::vector< std::string > & classes,
            const std::string & path )
```

Definition at line 65 of file Scripting.cpp.

4.10.3.3 load_plugin_source()

```
std::string Scripting::load_plugin_source (
            const char * source )
```

Definition at line 29 of file Scripting.cpp.

4.10.3.4 si_plugin()

```
bp::object Scripting::si_plugin (
            std::string & module_name,
            std::string & path,
            std::string & class_name )
```

Definition at line 24 of file Scripting.cpp.

4.10.4 Friends And Related Function Documentation

4.10.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Scripting & scripting )  [friend]
```

Definition at line 110 of file Scripting.cpp.

The documentation for this class was generated from the following files:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/Scripting.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/Scripting.cpp

4.11 SIGRun Class Reference

SIGRun class serving as entry point of an SI environment.

`#include <SIGRun.hpp>`

Public Member Functions

- SIGRun ()
  constructor
- ~SIGRun ()
  destructor
- int exec (int argc, char **argv)
  entry point of SIGRun

Static Public Member Functions

- static int quit ()
  exit SIGRun

4.11.1 Detailed Description

SIGRun class serving as entry point of an SI environment.

This class serves as the entry point of an SI environment. It is directly exposed in SI.hpp. An instance of this class is used to launch an SI environment.

See also

up_core

Definition at line 17 of file SIGRun.hpp.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 SIGRun()

```
SIGRun::SIGRun ( )
```

constructor

Constructor of SIGRun class. Used for instantiating objects.

Definition at line 19 of file SIGRun.cpp.

4.11.2.2 ~SIGRun()

```
SIGRun::~SIGRun ( )
```

destructor

Destructor of SIGRun class. Used for destroying objects.

Definition at line 30 of file SIGRun.cpp.

4.11.3 Member Function Documentation

4.11.3.1 exec()

```
int SIGRun::exec (
            int argc,
            char ** argv )
```

entry point of SIGRun

Entry point of SIGRun initializing all further systems.

Parameters

argc	cli argc
argv	cli argv

Definition at line 42 of file SIGRun.cpp.

4.11.3.2 quit()

```
int SIGRun::quit ( )  [static]
```

exit SIGRun

static exit function of SIGRun terminating all other systems

Definition at line 54 of file SIGRun.cpp.

The documentation for this class was generated from the following files:

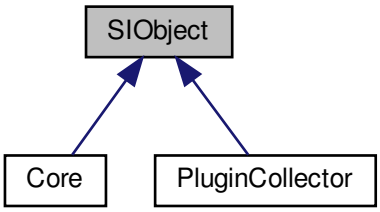
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/SIGRun.hpp
- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/SIGRun.cpp

4.12 SIObject Class Reference

A meta class from which other classes are derived from to register them as SIObject meta types.

```
#include <SIObject.hpp>
```

Inheritance diagram for SIObject:



Public Member Functions

- SIObject ()=default
  default constructor
- ~SIObject ()=default
  default destructor
- const std::string & meta_type () const
  function for retrieving meta type name

Protected Attributes

- std::string d_meta_type
  a std::string containing the name of the class to be registered as SIObject meta type

4.12.1 Detailed Description

A meta class from which other classes are derived from to register them as SIObject meta types.

This class enables registering other classes as SIObject meta types. This is currently achieved by storing std::strings containing the classes individual names. Currently, this meta typing is only used for Logging.

See also

Log::Log
d_meta_type

Definition at line 32 of file SIObject.hpp.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 SIObject()

```
SIObject::SIObject ( )  [default]
```

default constructor

4.12.2.2 ~SIObject()

```
SIObject::~SIObject ( )  [default]
```

default destructor

4.12.3 Member Function Documentation

4.12.3.1 meta_type()

```
const std::string& SIObject::meta_type ( ) const  [inline]
```

function for retrieving meta type name

The function for retrieving meta type name in a constant manner. Therefore, the instance calling this function will not mutate.

Returns

d_meta_type a const std::string reference of the type name of the clas

Definition at line 52 of file SIObject.hpp.

4.12.4 Member Data Documentation

4.12.4.1 d_meta_type

```
std::string SIObject::d_meta_type  [protected]
```

a std::string containing the name of the class to be registered as SIObject meta type

Definition at line 61 of file SIObject.hpp.

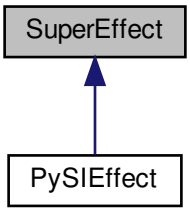
The documentation for this class was generated from the following file:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/SIObject.hpp

4.13 SuperEffect Class Reference

```
#include <SuperEffect.hpp>
```

Inheritance diagram for SuperEffect:



Public Member Functions

- virtual int on_enter (bp::object &other)=0
- virtual int on_continuous (bp::object &other)=0
- virtual int on_leave (bp::object &other)=0

4.13.1 Detailed Description

Definition at line 32 of file SuperEffect.hpp.

4.13.2 Member Function Documentation

4.13.2.1 on_continuous()

```
virtual int SuperEffect::on_continuous (
            bp::object & other )  [pure virtual]
```

Implemented in PySIEffect.

4.13.2.2 on_enter()

```
virtual int SuperEffect::on_enter (
            bp::object & other )  [pure virtual]
```

Implemented in PySIEffect.

4.13.2.3 on_leave()

```
virtual int SuperEffect::on_leave (
            bp::object & other )  [pure virtual]
```

Implemented in PySIEffect.

The documentation for this class was generated from the following file:

- /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/pysi/SuperEffect.hpp

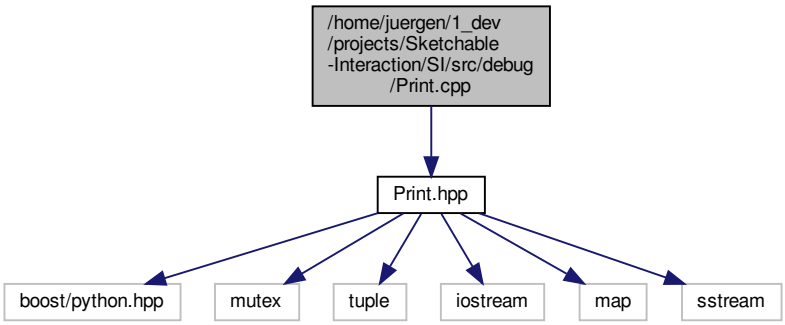
Chapter 5

File Documentation

5.1 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/debug/Print.cpp File Reference

```
#include "Print.hpp"
```

Include dependency graph for Print.cpp:

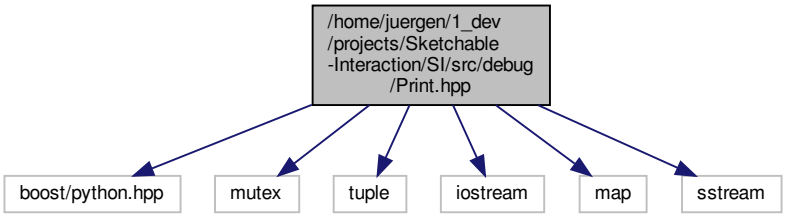


5.2 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/debug/Print.hpp File Reference

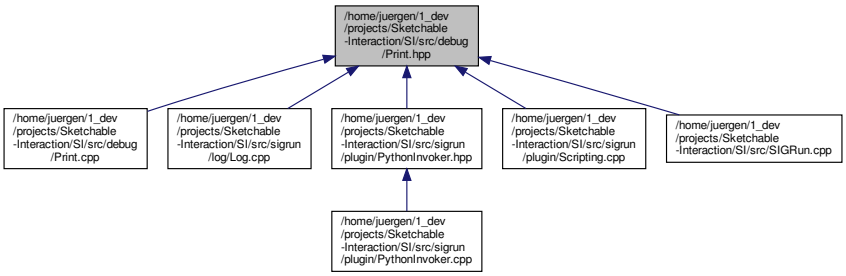
```
#include <boost/python.hpp>
#include <mutex>
#include <tuple>
#include <iostream>
#include <map>
```

```
#include <sstream>
```

Include dependency graph for Print.hpp:



This graph shows which files directly or indirectly include this file:



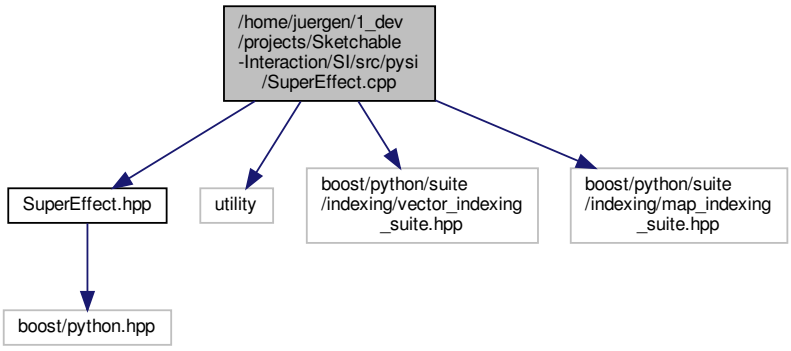
Classes

- class Print

5.3 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/pysi/SuperEffect.cpp File Reference

```
#include "SuperEffect.hpp"
#include <utility>
#include <boost/python/suite/indexing/vector_indexing_suite.hpp>
#include <boost/python/suite/indexing/map_indexing_suite.hpp>
```

Include dependency graph for SuperEffect.cpp:



Functions

- BOOST_PYTHON_MODULE (libPySI)

5.3.1 Function Documentation

5.3.1.1 BOOST_PYTHON_MODULE()

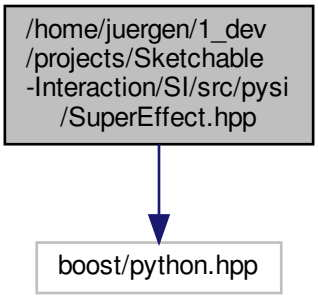
```
BOOST_PYTHON_MODULE (
              libPySI  )
```

Definition at line 62 of file SuperEffect.cpp.

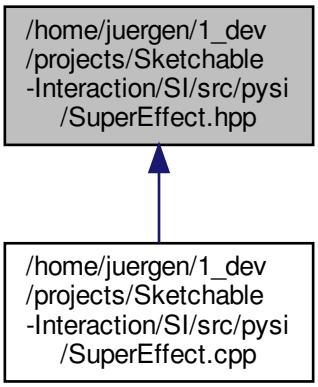
5.4 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/pysi/SuperEffect.hpp File Reference

```
#include <boost/python.hpp>
```

Include dependency graph for SuperEffect.hpp:



This graph shows which files directly or indirectly include this file:



Classes

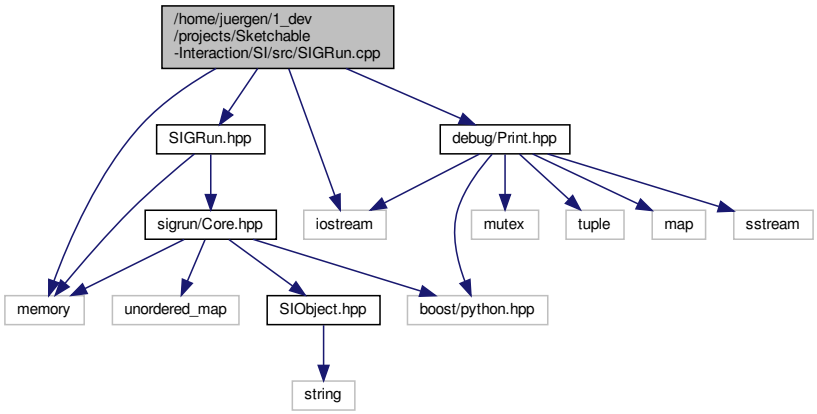
- class IterableConverter
- class SuperEffect
- class PySIEffect

5.5 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/SIGRun.cpp File Reference

```
#include <memory>
#include <iostream>
```

```
#include "SIGRun.hpp"
#include "debug/Print.hpp"
```

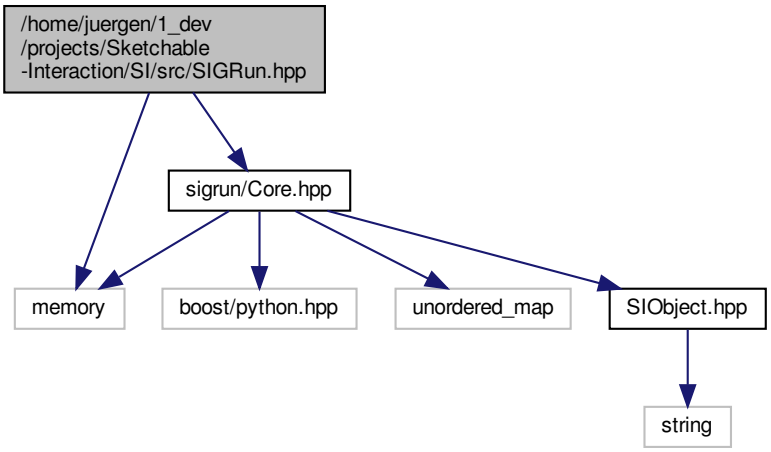
Include dependency graph for SIGRun.cpp:



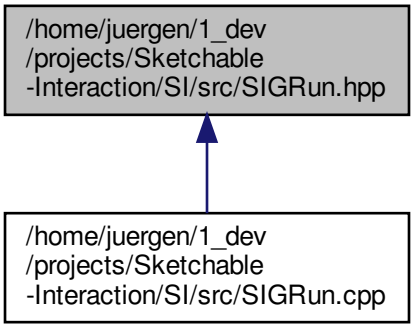
5.6 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/SIGRun.hpp File Reference

```
#include <memory>
#include "sigrun/Core.hpp"
```

Include dependency graph for SIGRun.hpp:



This graph shows which files directly or indirectly include this file:



Classes

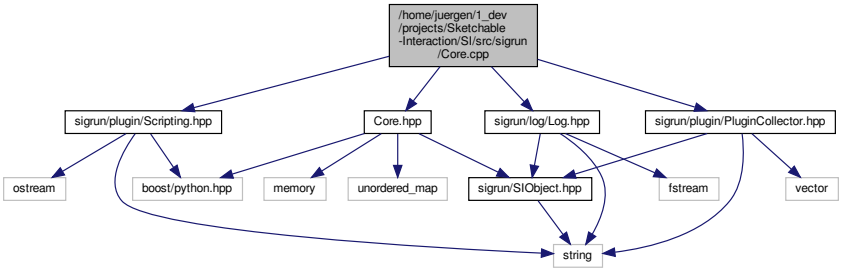
- class SIGRun

  SIGRun class serving as entry point of an SI environment.

5.7 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/Core.cpp File Reference

```
#include <sigrun/log/Log.hpp>
#include "Core.hpp"
#include "sigrun/plugin/Scripting.hpp"
#include "sigrun/plugin/PluginCollector.hpp"
```

Include dependency graph for Core.cpp:

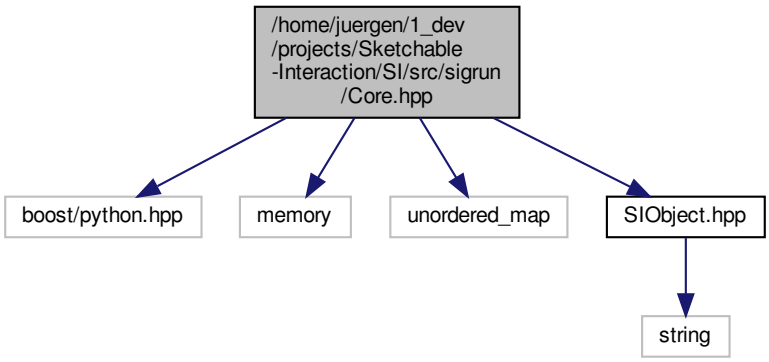


5.8 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/Core.hpp File Reference

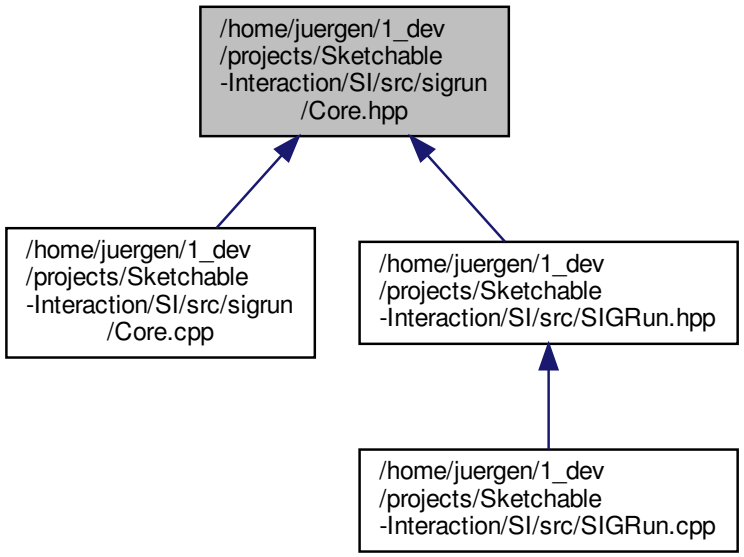
```
#include <boost/python.hpp>
#include <memory>
```

```
#include <unordered_map>
#include "SIObject.hpp"
```

Include dependency graph for Core.hpp:



This graph shows which files directly or indirectly include this file:



Classes

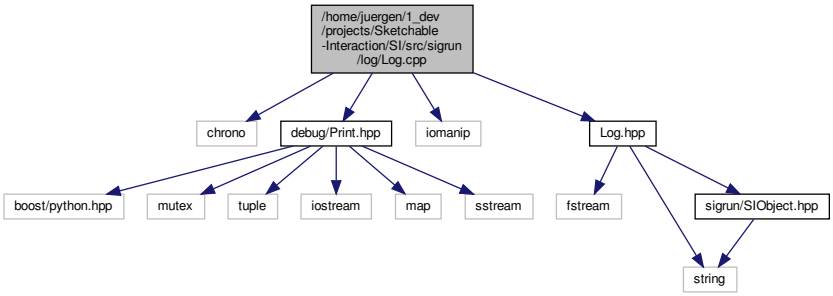
- class Core

  namespace shortening for python object integration

5.9 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/log/Log.cpp File Reference

```
#include <chrono>
#include <debug/Print.hpp>
#include <iomanip>
#include "Log.hpp"
```

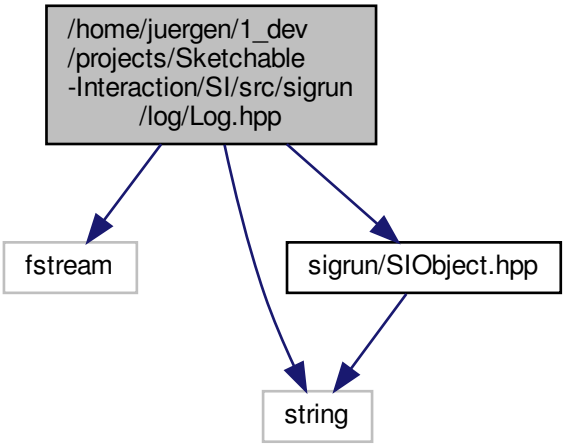
Include dependency graph for Log.cpp:



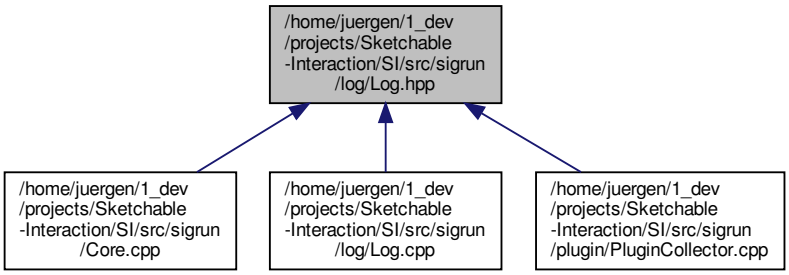
5.10 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/log/Log.hpp File Reference

```
#include <fstream>
#include <string>
#include "sigrun/SIObject.hpp"
```

Include dependency graph for Log.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Log

  Log class serving as central logging functionality for easy logging data output.

Macros

- #define ERROR_COLOR(x) ("\033[31m" + x + "\033[0m")

  red coloring for console output
- #define UNDEFINED_COLOR(x) ("\033[1;31m" + x + "\033[0m")

  bold red coloring for console output
- #define INFO_COLOR(x) ("\033[32m" + x + "\033[0m")

  green coloring for console output
- #define WARN_COLOR(x) ("\033[33m" + x + "\033[0m")

  yellow coloring for console output
- #define DEBUG_COLOR(x) ("\033[37m" + x + "\033[0m")

  white/gray coloring for console output
- #define __FILENAME__ (strrchr(__FILE__, '/') ? strrchr(__FILE__, '/') + 1 : __FILE__)

  file name and extension without full path
- #define LOG_NONE Log::MODE::NONE

  disable logging output
- #define LOG_CONSOLE Log::MODE::CONSOLE

  output logging data to stdout
- #define LOG_FILE Log::MODE::FILE

  output logging data to file
- #define LOG_SHOW_NONE Log::SHOW_TYPE::HIDDEN

  disable logging except for errors and undefined behaviour
- #define LOG_SHOW_INFO Log::SHOW_TYPE::INFO

  enable logging of data tagged as INFO (information) additionally to errors and undefined behaviour
- #define LOG_SHOW_WARN Log::SHOW_TYPE::WARN

  enable logging of data tagged as WARN (warning) additionally to errors and undefined behaviour
- #define LOG_SHOW_ERROR Log::SHOW_TYPE::ERROR

  enable logging of data tagged as ERROR (error) however this per default enabled and cannot be disabled
- #define LOG_SHOW_DEBUG Log::SHOW_TYPE::DEBUG

enable logging of data tagged as DEBUG (debugging information) additionally to errors and undefined behaviour

- #define LOG_SHOW_ALL Log::SHOW_TYPE::INFO | Log::SHOW_TYPE::WARN | Log::SHOW_TYPE::ERROR | Log::SHOW_TYPE::DEBUG

  enable logging of any tagged data

- #define DEBUG(what, log_mode) Log::log(what, Log::LOG_LEVEL::DEBUG_LEVEL, log_mode, meta_type(),__FILENAME__, __FUNCTION__, std::to_string(__LINE__))

  perform logging of data with the DEBUG tag

- #define INFO(what, log_mode) Log::log(what, Log::LOG_LEVEL::INFO_LEVEL, log_mode, meta_type())

  perform logging of data with the INFO tag

- #define ERROR(what, log_mode) Log::log(what, Log::LOG_LEVEL::ERROR_LEVEL, log_mode, meta_type(),__FILENAME__, __FUNCTION__, std::to_string(__LINE__))

  perform logging of data with the ERROR tag

- #define WARN(what, log_mode) Log::log(what, Log::LOG_LEVEL::WARN_LEVEL, log_mode, meta_type())

  perform logging of data with the WARN tag

- #define UNDEFINED(what, log_mode) Log::log(what, Log::LOG_LEVEL::UNDEFINED_LEVEL, log_mode, meta_type(), __FILENAME__, __FUNCTION__, std::to_string(__LINE__))

  perform logging of data with the UNDEFINED tag

5.10.1 Macro Definition Documentation

5.10.1.1 __FILENAME__

```
#define __FILENAME__ (strrchr(__FILE__, '/') ?  strrchr(__FILE__, '/') + 1 :  __FILE__)
```

file name and extension without full path

file name and extension without full path

Definition at line 61 of file Log.hpp.

5.10.1.2 DEBUG

```
#define DEBUG(
              what,
              log_mode ) Log::log(what, Log::LOG_LEVEL::DEBUG_LEVEL, log_mode, meta_type(),__FILENAME__,
__FUNCTION__, std::to_string(__LINE__))
```

perform logging of data with the DEBUG tag

Shortcut macro for logging of data with the DEBUG tag which uses static access of log() function of Log class

Parameters

what	the message to be logged
log_mode	the description where the message is outputted (

See also

Log::MODE::CONSOLE or
Log::MODE::FILE or both)
Log::log()

Definition at line 118 of file Log.hpp.

5.10.1.3 DEBUG_COLOR

```
#define DEBUG_COLOR(
              x ) ("\033[37m" + x + "\033[0m")
```

white/gray coloring for console output

Coloring for console output. Unused for file output. See table of codes here: https://en.wikipedia.org/wiki/ANSI_escape_code#graphics

Definition at line 54 of file Log.hpp.

5.10.1.4 ERROR

```
#define ERROR(
              what,
              log_mode ) Log::log(what, Log::LOG_LEVEL::ERROR_LEVEL, log_mode, meta_type(),__FILENAME__,
__FUNCTION__, std::to_string(__LINE__))
```

perform logging of data with the ERROR tag

Shortcut macro for logging of data with the ERROR tag which uses static access of log() function of Log class

Parameters

what	the message to be logged
log_mode	the description where the message is outputted (

See also

Log::MODE::CONSOLE or
Log::MODE::FILE or both)
Log::log()

Definition at line 142 of file Log.hpp.

5.10.1.5 ERROR_COLOR

```
#define ERROR_COLOR(
              x ) ("\033[31m" + x + "\033[0m")
```

red coloring for console output

Coloring for console output. Unused for file output. See table of codes here: https://en.wikipedia.org/wiki/ANSI_escape_code#graphics

Definition at line 18 of file Log.hpp.

5.10.1.6 INFO

```
#define INFO(
              what,
              log_mode ) Log::log(what, Log::LOG_LEVEL::INFO_LEVEL, log_mode, meta_type())
```

perform logging of data with the INFO tag

Shortcut macro for logging of data with the INFO tag which uses static access of log() function of Log class

Parameters

what	the message to be logged
log_mode	the description where the message is outputted (

See also

Log::MODE::CONSOLE or
Log::MODE::FILE or both)
Log::log()

Definition at line 130 of file Log.hpp.

5.10.1.7 INFO_COLOR

```
#define INFO_COLOR(
              x ) ("\033[32m" + x + "\033[0m")
```

green coloring for console output

Coloring for console output. Unused for file output. See table of codes here: https://en.wikipedia.org/wiki/ANSI_escape_code#graphics

Definition at line 36 of file Log.hpp.

5.10.1.8 LOG_CONSOLE

```
#define LOG_CONSOLE Log::MODE::CONSOLE
```

output logging data to stdout

Definition at line 71 of file Log.hpp.

5.10.1.9 LOG_FILE

```
#define LOG_FILE Log::MODE::FILE
```

output logging data to file

Definition at line 76 of file Log.hpp.

5.10.1.10 LOG_NONE

```
#define LOG_NONE Log::MODE::NONE
```

disable logging output

Definition at line 66 of file Log.hpp.

5.10.1.11 LOG_SHOW_ALL

```
#define LOG_SHOW_ALL Log::SHOW_TYPE::INFO | Log::SHOW_TYPE::WARN | Log::SHOW_TYPE::ERROR | Log::SHOW_TYPE::DEBU
```

enable logging of any tagged data

Definition at line 106 of file Log.hpp.

5.10.1.12 LOG_SHOW_DEBUG

```
#define LOG_SHOW_DEBUG Log::SHOW_TYPE::DEBUG
```

enable logging of data tagged as DEBUG (debugging information) additionally to errors and undefined behaviour

Definition at line 101 of file Log.hpp.

5.10.1.13 LOG_SHOW_ERROR

```
#define LOG_SHOW_ERROR Log::SHOW_TYPE::ERROR
```

enable logging of data tagged as ERROR (error) however this per default enabled and cannot be disabled

Definition at line 96 of file Log.hpp.

5.10.1.14 LOG_SHOW_INFO

```
#define LOG_SHOW_INFO Log::SHOW_TYPE::INFO
```

enable logging of data tagged as INFO (information) additionally to errors and undefined behaviour

Definition at line 86 of file Log.hpp.

5.10.1.15 LOG_SHOW_NONE

```
#define LOG_SHOW_NONE Log::SHOW_TYPE::HIDDEN
```

disable logging except for errors and undefined behaviour

Definition at line 81 of file Log.hpp.

5.10.1.16 LOG_SHOW_WARN

```
#define LOG_SHOW_WARN Log::SHOW_TYPE::WARN
```

enable logging of data tagged as WARN (warning) additionally to errors and undefined behaviour

Definition at line 91 of file Log.hpp.

5.10.1.17 UNDEFINED

```
#define UNDEFINED(
              what,
              log_mode ) Log::log(what, Log::LOG_LEVEL::UNDEFINED_LEVEL, log_mode, meta_↩
type(), __FILENAME__, __FUNCTION__, std::to_string(__LINE__))
```

perform logging of data with the UNDEFINED tag

Shortcut macro for logging of data with the UNDEFINED tag which uses static access of log() function of Log class

Parameters

what	the message to be logged
log_mode	the description where the message is outputted (

See also

Log::MODE::CONSOLE or
Log::MODE::FILE or both)
Log::log()

Definition at line 166 of file Log.hpp.

5.10.1.18 UNDEFINED_COLOR

```
#define UNDEFINED_COLOR(
              x ) ("\033[1;31m" + x + "\033[0m")
```

bold red coloring for console output

Coloring for console output. Unsused for file output. See table of codes here: https://en.wikipedia.org/wiki/ANSI_escape_code#graphics

Definition at line 27 of file Log.hpp.

5.10.1.19 WARN

```
#define WARN(
              what,
              log_mode ) Log::log(what, Log::LOG_LEVEL::WARN_LEVEL, log_mode, meta_type())
```

perform logging of data with the WARN tag

Shortcut macro for logging of data with the WARN tag which uses static access of log() function of Log class

Parameters

what	the message to be logged
log_mode	the description where the message is outputted (

See also

Log::MODE::CONSOLE or
Log::MODE::FILE or both)
Log::log()

Definition at line 154 of file Log.hpp.

5.10.1.20 WARN_COLOR

```
#define WARN_COLOR(
              x ) ("\033[33m" + x + "\033[0m")
```

yellow coloring for console output

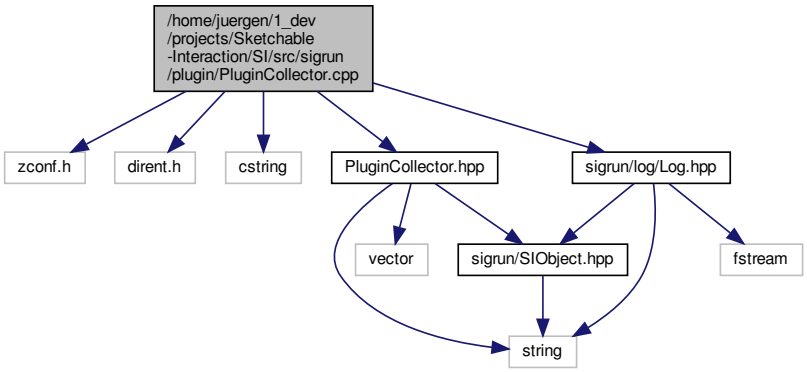
Coloring for console output. Unsused for file output. See table of codes here: https://en.wikipedia.org/wiki/ANSI_escape_code#graphics

Definition at line 45 of file Log.hpp.

5.11 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/PluginCollector.cpp File Reference

```
#include <zconf.h>
#include <dirent.h>
#include <cstring>
#include "PluginCollector.hpp"
#include "sigrun/log/Log.hpp"
```

Include dependency graph for PluginCollector.cpp:

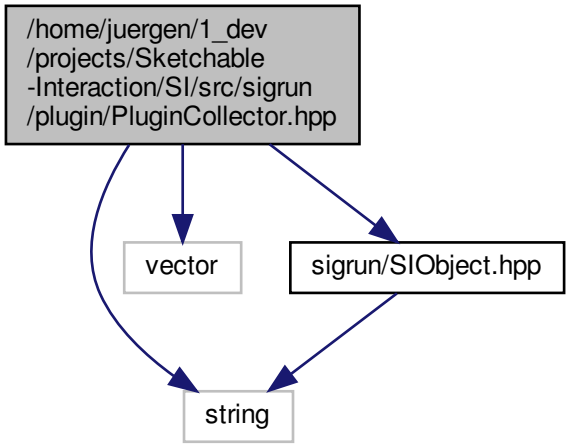


5.12 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/PluginCollector.hpp File Reference

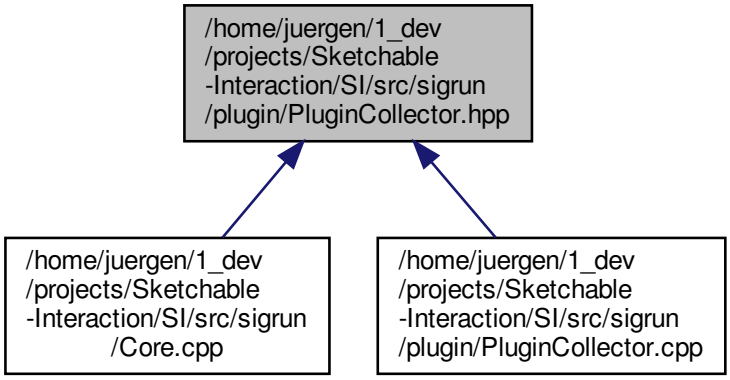
```
#include <string>
#include <vector>
```

```
#include <sigrun/SIObject.hpp>
```

Include dependency graph for PluginCollector.hpp:



This graph shows which files directly or indirectly include this file:



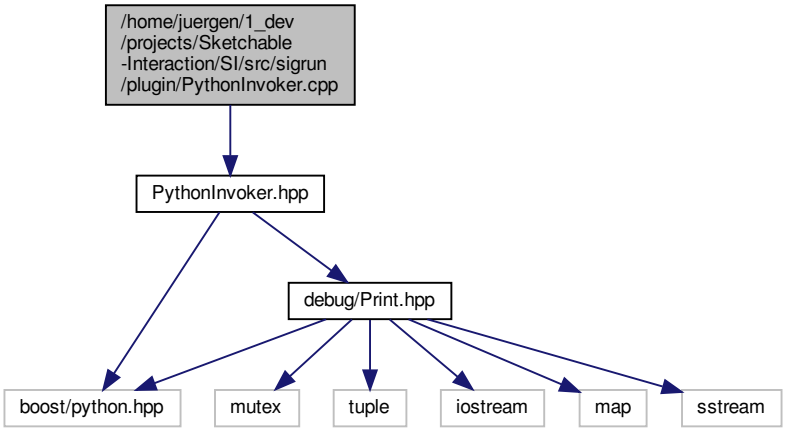
Classes

- class PluginCollector

5.13 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/PythonInvoker.cpp File Reference

```
#include "PythonInvoker.hpp"
```

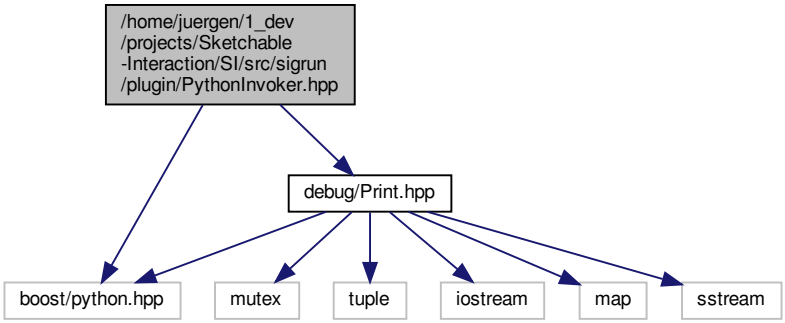
Include dependency graph for PythonInvoker.cpp:



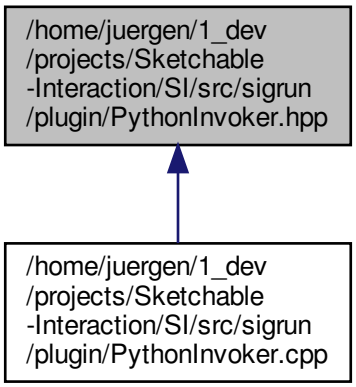
5.14 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/PythonInvoker.hpp File Reference

```
#include <boost/python.hpp>
#include "debug/Print.hpp"
```

Include dependency graph for PythonInvoker.hpp:



This graph shows which files directly or indirectly include this file:



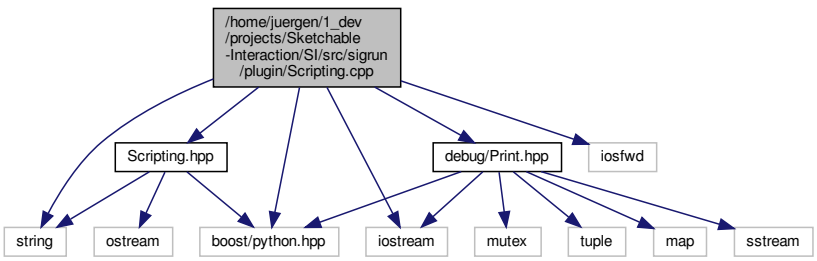
Classes

- class PythonInvoker

5.15 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/Scripting.cpp File Reference

```
#include "Scripting.hpp"
#include <iostream>
#include <string>
#include <iosfwd>
#include <boost/python.hpp>
#include <debug/Print.hpp>
```

Include dependency graph for Scripting.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const Scripting &scripting)

5.15.1 Function Documentation

5.15.1.1 operator<<()

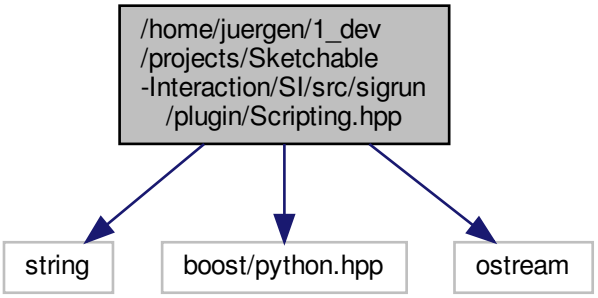
```
std::ostream& operator<< (
            std::ostream & os,
            const Scripting & scripting )
```

Definition at line 110 of file Scripting.cpp.

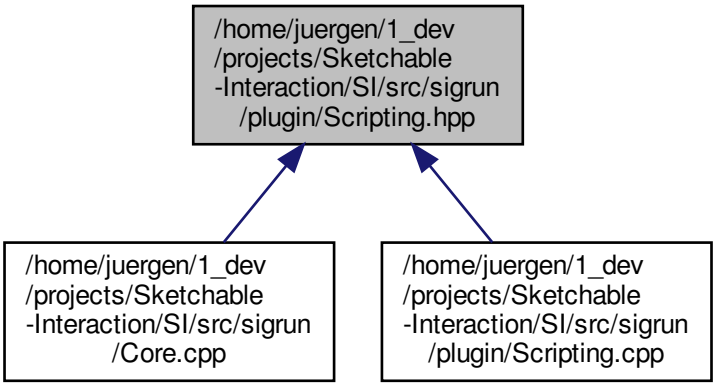
5.16 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/plugin/Scripting.hpp File Reference

```
#include <string>
#include <boost/python.hpp>
#include <ostream>
```

Include dependency graph for Scripting.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Scripting

Functions

- PyObject * PyInit_libPySI (void)

5.16.1 Function Documentation

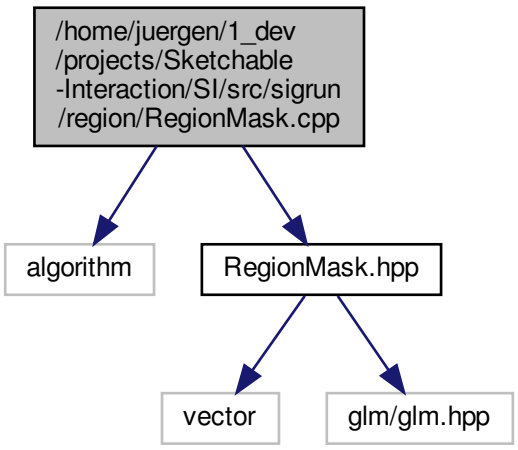
5.16.1.1 PyInit_libPySI()

```
PyObject* PyInit_libPySI (
            void  )
```

5.17 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/region/RegionMask.cpp File Reference

```
#include <algorithm>
#include "RegionMask.hpp"
```

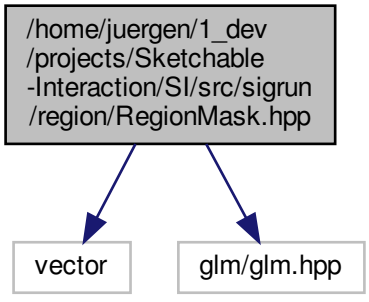
Include dependency graph for RegionMask.cpp:



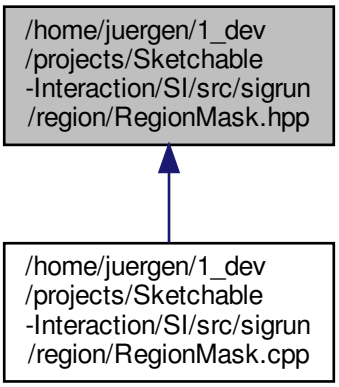
5.18 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/region/RegionMask.hpp File Reference

```
#include <vector>
#include <glm/glm.hpp>
```

Include dependency graph for RegionMask.hpp:



This graph shows which files directly or indirectly include this file:



Classes

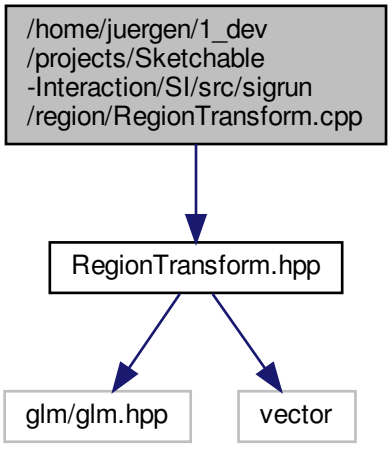
- class RegionMask

  RegionMask class which stores a bit array used for true collision testing.

5.19 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/region/RegionTransform.cpp File Reference

```
#include "RegionTransform.hpp"
```

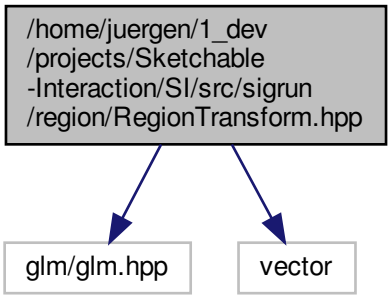
Include dependency graph for RegionTransform.cpp:



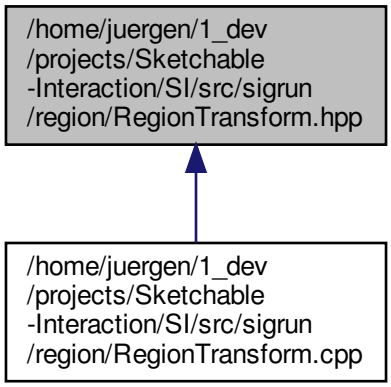
5.20 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/region/RegionTransform.hpp File Reference

```
#include <glm/glm.hpp>
#include <vector>
```

Include dependency graph for RegionTransform.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class RegionTransform

  RegionTransform class storing the relative translation, rotation and scale of a contour.

Macros

- #define PI_DIV_180 (float) 0.0174532925199

  quivalent to M_PI / 180.0

5.20.1 Macro Definition Documentation

5.20.1.1 PI_DIV_180

```
#define PI_DIV_180 (float) 0.0174532925199
```

quivalent to M_PI / 180.0

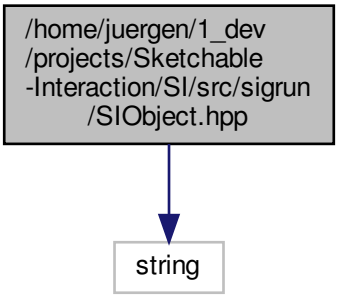
Equivalent to M_PI / 180.0. Can be used to convert angles given in degrees to radians.

Definition at line 15 of file RegionTransform.hpp.

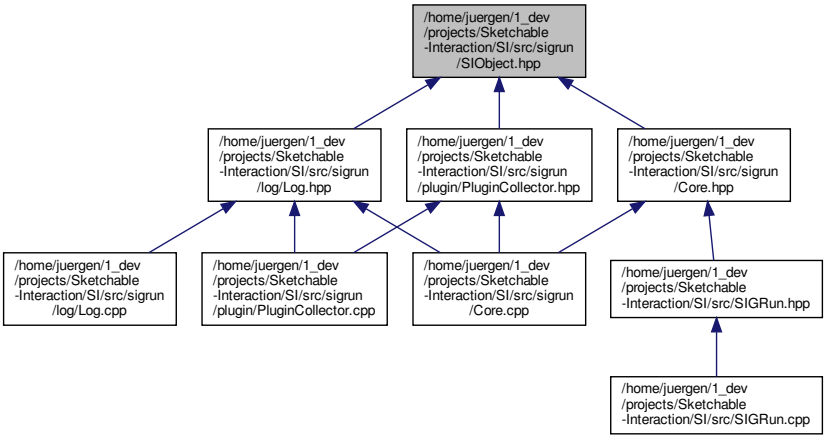
5.21 /home/juergen/1_dev/projects/Sketchable-Interaction/SI/src/sigrun/SIObject.hpp File Reference

```
#include <string>
```

Include dependency graph for SIObject.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class SIObject

  A meta class from which other classes are derived from to register them as SIObject meta types.

Macros

- #define SIOBJECT(type) (d_meta_type = type);

  macro for registering another class as SIObject

5.21.1 Macro Definition Documentation

5.21.1.1 SIOBJECT

```
#define SIOBJECT(
              type ) (d_meta_type = type);
```

macro for registering another class as SIObject

The macro is a shortcut for registering other classes which are derived from SIObject as such a SIObject. Syntax: class A: public SIObject {SIOBJECT("A") ... };

Parameters

type	a std::string containing the type name of a class to be registered as SIObject.

Definition at line 19 of file SIObject.hpp.

Index