MAGE

Generated by Doxygen 1.8.12

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Namespace Documentation

4.1 mage Namespace Reference

Classes

- struct AABB
- class ConditionVariable
- class DeviceEnumeration
- struct DisplayMode
- struct Edge
- class Engine
- struct EngineSetup
- struct Face
- struct GeneralConfiguration
- class GraphicsManager
- struct IndexedEdge
- struct IndexedFace
- class Input
- struct LVertex
- class MemoryArena
- class Mutex
- struct MutexLock
- class ProgressReporter
- class ReadWriteMutex
- struct ReadWriteMutexLock
- class Reference
- class ReferenceCounted
- class Resource
- class ResourceManager
- class Semaphore
- class Sound
- struct Sphere
- class State
- class StateManager
- class Task
- class Timer
- struct TLVertex
- struct Variable
- class VariableScript
- struct Vertex
- struct ViewerSetup

Enumerations

- enum ReadWriteMutexLockType { READ, WRITE }
- enum VariableType {
  BoolType, IntType, FloatType, Float3Type,
  Float4Type, ColourType, StringType, UnknownType }

Functions

- LRESULT CALLBACK WindowProc (HWND hwnd, UINT msg, WPARAM wparam, LPARAM lparam)
- static bool AttachConsole ()
- static void PrintConsoleHeader ()
- INT_PTR CALLBACK SettingsDialogProcDelegate (HWND hwndDlg, UINT uMsg, WPARAM wParam, LP↩ ARAM lParam)
- void ComboBoxAdd (HWND dialog, int id, const void *data, const wchar_t *desc)
- void ComboBoxSelect (HWND dialog, int id, int index)
- void ComboBoxSelect (HWND dialog, int id, const void *data)
- const void * ComboBoxSelected (HWND dialog, int id)
- bool ComboBoxSomethingSelected (HWND dialog, int id)
- int ComboBoxCount (HWND dialog, int id)
- bool ComboBoxContains (HWND dialog, int id, const wchar_t *desc)
- const char * FindWordEnd (const char *buffer)
- void ProcessError (const char *format, const va_list args, const string &error_type, int error_disposition)
- void Info (const char *format,...)
- void Warning (const char *format,...)
- void Error (const char *format,...)
- void Severe (const char *format,...)
- int TerminalWidth ()
- void * AllocAligned (size_t size)
- template<typename T >
  T * AllocAligned (uint32_t count)
- void FreeAligned (void *ptr)
- template<typename T >
  T * AtomicCompareAndSwapPointer (T **destination, T *exchange, T *comparand)
- int32_t AtomicAdd (AtomicInt32 *addend, int32_t value)
- int32_t AtomicCompareAndSwap (AtomicInt32 *destination, int32_t exchange, int32_t comparand)
- float AtomicAdd (volatile float *addend, float value)
- uint32_t NumberOfSystemCores ()
- static DWORD WINAPI task_entry (LPVOID lpParameter)
- void TasksInit ()
- void TasksCleanup ()
- void EnqueueTasks (const vector< Task *> &tasks)
- void WaitForAllTasks ()

Variables

- GeneralConfiguration general_configuration
- Engine * g_engine = NULL
- DeviceEnumeration * g_device_enumeration = NULL
- const D3D11_INPUT_ELEMENT_DESC vertex_input_element_desc [ ]
- const D3D11_INPUT_ELEMENT_DESC lvertex_input_element_desc [ ]
- const D3D11_INPUT_ELEMENT_DESC tlvertex_input_element_desc [ ]
- static HANDLE * threads
- static Mutex * task_queue_mutex = Mutex::Create()
- static vector< Task * > task_queue
- static Semaphore * worker_semaphore
- static uint32_t nb_unfinished_tasks
- static ConditionVariable * tasks_running_condition

4.1.1 Detailed Description

The namespace for all the MAGE functionality.

4.1.2 Enumeration Type Documentation

4.1.2.1 ReadWriteMutexLockType

```
enum mage::ReadWriteMutexLockType
```

Type of read write mutex locks.

Enumerator

READ	
WRITE	

4.1.2.2 VariableType

```
enum mage::VariableType
```

Enumeration of variable types.

Enumerator

BoolType	
IntType	
FloatType	
Float3Type	
Float4Type	
ColourType	
StringType	
UnknownType	

4.1.3 Function Documentation

4.1.3.1 AllocAligned() [1/2]

```
void* mage::AllocAligned (
            size_t size )
```

Allocates memory on an alignment boundary of 64 bytes of the given size.

Parameters

in	*size*	The requested size in bytes to allocate in memory.

Returns

NULL if the allocation failed.
A pointer to the memory block that was allocated. The pointer is a multiple of the alignment of 64 bytes.

4.1.3.2 AllocAligned() [2/2]

```
template<typename T >
T* mage::AllocAligned (
            uint32_t count )
```

Allocates memory on an alignment boundary of 64 bytes.

Template Parameters

T	The type of objects to allocate in memory.

Parameters

in	*count*	The number of objects of type T to allocate in memory.

Returns

NULL if the allocation failed.
A pointer to the memory block that was allocated. The pointer is a multiple of the alignment of 64 bytes.

4.1.3.3 AtomicAdd() [1/2]

```
int32_t mage::AtomicAdd (
            AtomicInt32 * addend,
            int32_t value )
```

Performs an atomic addition operation on the specified values.

Parameters

in,out	*addend*	A pointer to the first operand. This value will be replaced with the result of the operation.
in	*value*	The second operand.

Returns

The function returns the result of the operation.

4.1.3.4 AtomicAdd() [2/2]

```
float mage::AtomicAdd (
            volatile float * addend,
            float value )
```

Performs an atomic addition operation on the specified values.

Parameters

in,out	*addend*	A pointer to the first operand. This value will be replaced with the result of the operation.
in	*value*	The second operand.

Returns

The function returns the result of the operation.

4.1.3.5 AtomicCompareAndSwap()

```
int32_t mage::AtomicCompareAndSwap (
            AtomicInt32 * destination,
            int32_t exchange,
            int32_t comparand )
```

Performs an atomic compare-and-exchange operation on the specified values. The function compares the original value against a given comparand value and exchanges the original value with a given exchange value in case of equality.

Parameters

in,out	*destination*	
in	*exchange*	The exchange value.
in	*comparand*	The value to compare to *destination*.

Returns

The function returns the initial value of *destination*.

4.1.3.6 AtomicCompareAndSwapPointer()

```
template<typename T >
T* mage::AtomicCompareAndSwapPointer (
            T ** destination,
            T * exchange,
            T * comparand )
```

Performs an atomic compare-and-exchange operation on the specified pointers. The function compares the original pointer against a given comparand pointer and exchanges the original pointer with a given exchange pointer in case of equality.

Parameters

in,out	*destination*	
in	*exchange*	The exchange pointer.
in	*comparand*	The pointer to compare to *destination*.

Returns

The function returns the initial pointer of *destination*.

4.1.3.7 AttachConsole()

```
static bool mage::AttachConsole ( )  [static]
```

Allocates a console to the engine for basic io and redirects stdin, stdout and stderr to the allocated console.

Returns

`true` if a console is successfully attached. `false` otherwise.

4.1.3.8 ComboBoxAdd()

```
void mage::ComboBoxAdd (
            HWND dialog,
            int id,
            const void * data,
            const wchar_t * desc )
```

Adds an item associated with the given data and described with the given descriptor to a combo box.

Parameters

in	*dialog*	A handle to the dialog box that contains the control.
in	*id*	The identifier of the control to be retrieved.
in	*data*	A pointer to the data of the item to add.
in	*desc*	The description of the item to add.

4.1.3.9 ComboBoxContains()

```
bool mage::ComboBoxContains (
            HWND dialog,
            int id,
            const wchar_t * desc )
```

Checks whether a combo box contains the given descriptor.

Parameters

in	*dialog*	A handle to the dialog box that contains the control.
in	*id*	The identifier of the control to be retrieved.
in	*desc*	The string description to check.

Returns

`true` if the given description is contained in the combo box. `false` otherwise.

4.1.3.10 ComboBoxCount()

```
int mage::ComboBoxCount (
            HWND dialog,
            int id )
```

Returns the number of items in a combo box.

Parameters

in	*dialog*	A handle to the dialog box that contains the control.
in	*id*	The identifier of the control to be retrieved.

Returns

The number of items of a combo box.

4.1.3.11 ComboBoxSelect() [1/2]

```
void mage::ComboBoxSelect (
            HWND dialog,
            int id,
            int index )
```

Selects the item at the given index in a combo box.

Parameters

in	*dialog*	A handle to the dialog box that contains the control.
in	*id*	The identifier of the control to be retrieved.
in	*index*	The index of the item.

4.1.3.12 ComboBoxSelect() [2/2]

```
void mage::ComboBoxSelect (
            HWND dialog,
            int id,
            const void * data )
```

Selects the item associated with the given data in a combo box.

Parameters

in	*dialog*	A handle to the dialog box that contains the control.
in	*id*	The identifier of the control to be retrieved.
in	*data*	A pointer to the data of the item.

4.1.3.13 ComboBoxSelected()

```
const void * mage::ComboBoxSelected (
            HWND dialog,
            int id )
```

Returns the data associated with the selected item in a combo box.

Parameters

in	*dialog*	A handle to the dialog box that contains the control.
in	*id*	The identifier of the control to be retrieved.

Returns

`NULL` if the combo box has no items.
A pointer to the data associated with the selected item in the combo box.

4.1.3.14 ComboBoxSomethingSelected()

```
bool mage::ComboBoxSomethingSelected (
            HWND dialog,
            int id )
```

Checks whether a valid item is selected in a combo box.

Parameters

in	*dialog*	A handle to the dialog box that contains the control.
in	*id*	The identifier of the control to be retrieved.

Returns

`true` if a valid item is selected in the combo box. `false` otherwise.

4.1.3.15 EnqueueTasks()

```
void mage::EnqueueTasks (
            const vector< Task * > & tasks )
```

Enqueues the given tasks.

Parameters

in	*tasks*	The tasks.

4.1.3.16 Error()

```
void mage::Error (
            const char * format,
             ...  )
```

Notifies an error message.

Parameters

in	*format*	Pointer to the message format.

4.1.3.17 FindWordEnd()

```
const char* mage::FindWordEnd (
            const char * buffer )
```

Finds the end of a word.

Parameters

in	*buffer*	Pointer to the first character.

Returns

Pointer to the end of the word. This means the pointer points to a space or null-terminating character.

4.1.3.18 FreeAligned()

```
void mage::FreeAligned (
            void * ptr )
```

Frees a block of memory that was allocated with mage::AllocAligned(size_t) or mage::AllocAligned<T>(uint32_t).

Parameters

in	*ptr*	A pointer to the memory block that was allocated.

4.1.3.19 Info()

```
void mage::Info (
            const char * format,
             ...  )
```

Notifies an info message.

Parameters

in	*format*	Pointer to the message format.

4.1.3.20 NumberOfSystemCores()

```
uint32_t mage::NumberOfSystemCores ( )
```

Returns the number of system cores (i.e. logical processors).

Returns

The number of system cores (i.e. logical processors).

4.1.3.21 PrintConsoleHeader()

```
static void mage::PrintConsoleHeader ( )  [static]
```

Prints the header of the engine to the console.

4.1.3.22 ProcessError()

```
void mage::ProcessError (
            const char * format,
            const va_list args,
            const string & error_type,
            int error_disposition )
```

Process the given error.

Parameters

in	*format*	The format of the error string.
in	*args*	The arguments of the format string.
in	*error_type*	The type of the error.
in	*error_disposition*	Disposition of the error.

4.1.3.23 SettingsDialogProcDelegate()

```
INT_PTR CALLBACK mage::SettingsDialogProcDelegate (
            HWND hwndDlg,
            UINT uMsg,
            WPARAM wParam,
            LPARAM lParam )
```

Engine-defined callback function used with the CreateDialog for device enumeration.

Parameters

in	*hwndDlg*	A handle to the dialog box.
in	*uMsg*	The message.
in	*wParam*	Additional message-specific information.
in	*lParam*	Additional message-specific information.

4.1.3.24 Severe()

```
void mage::Severe (
            const char * format,
             ...  )
```

Notifies a severe message.

Parameters

in	*format*	Pointer to the message format.

4.1.3.25 task_entry()

```
static DWORD WINAPI mage::task_entry (
            LPVOID lpParameter )  [static]
```

An application-defined function that serves as the starting address for a thread.

Parameters

in	*lpParameter*	The thread data passed to the function using the `lpParameter` parameter of CreateThread.

Returns

A value indicating success or failure.

4.1.3.26 TasksCleanup()

```
void mage::TasksCleanup ( )
```

Clean the tasks.

4.1.3.27 TasksInit()

```
void mage::TasksInit ( )
```

Initialize the tasks.

4.1.3.28 TerminalWidth()

```
int mage::TerminalWidth ( )
```

Returns the fixed terminal width.

Returns

The fixed terminal width.

4.1.3.29 WaitForAllTasks()

```
void mage::WaitForAllTasks ( )
```

Waits for all the tasks to finish.

4.1.3.30 Warning()

```
void mage::Warning (
            const char * format,
             ...  )
```

Notifies a warning message.

Parameters

in	*format*	Pointer to the message format.

4.1.3.31 WindowProc()

```
LRESULT CALLBACK mage::WindowProc (
            HWND hwnd,
            UINT msg,
            WPARAM wparam,
            LPARAM lparam )
```

The application-defined function that processes messages sent to the engine window. The WindowProc type defines a pointer to this callback function.

Parameters

in	*hwnd*	A handle to the window.
in	*msg*	The message.
in	*wparam*	Additional message information. The contents of this parameter depend on the value of *msg*.
in	*lparam*	Additional message information. The contents of this parameter depend on the value of *msg*.

Returns

The return value is the result of the message processing and depends on the message sent.

4.1.4 Variable Documentation

4.1.4.1 g_device_enumeration

```
DeviceEnumeration * mage::g_device_enumeration = NULL
```

A (global) pointer to the device enumeration.

4.1.4.2 g_engine

```
Engine * mage::g_engine = NULL
```

The engine used by the user.

4.1.4.3 general_configuration

```
GeneralConfiguration mage::general_configuration
```

The general configuration defined by the user and used by the engine.

4.1.4.4 lvertex_input_element_desc

```
const D3D11_INPUT_ELEMENT_DESC mage::lvertex_input_element_desc[]
```

Initial value:

```
= {
        { "POSITION", 0, DXGI_FORMAT_R32G32B32_FLOAT, 0, UINT(offsetof(LVertex, p)),
      D3D11_INPUT_PER_VERTEX_DATA, 0 },
        { "DIFFUSE", 0, DXGI_FORMAT_R32G32B32A32_FLOAT, 0, UINT(offsetof(LVertex, diffuse)),
      D3D11_INPUT_PER_VERTEX_DATA, 0 },
        { "UV", 0, DXGI_FORMAT_R32G32_FLOAT, 0, UINT(offsetof(LVertex, tu)), D3D11_INPUT_PER_VERTEX_DATA, 0
       }
    }
```

Input element descriptor for a LVertex.

4.1.4.5 nb_unfinished_tasks

```
uint32_t mage::nb_unfinished_tasks  [static]
```

The number of unfinished tasks.

4.1.4.6 task_queue

```
vector<Task *> mage::task_queue  [static]
```

The task queue.

4.1.4.7 task_queue_mutex

```
Mutex* mage::task_queue_mutex = Mutex::Create()  [static]
```

The mutex for exclusive access to the task queue.

4.1.4.8 tasks_running_condition

```
ConditionVariable* mage::tasks_running_condition  [static]
```

The running condition variable for exclusive access to the number of unfinished tasks and for signaling on updates.

4.1.4.9 threads

```
HANDLE* mage::threads  [static]
```

The thread handles.

4.1.4.10 tlvertex_input_element_desc

```
const D3D11_INPUT_ELEMENT_DESC mage::tlvertex_input_element_desc[]
```

Initial value:

```
= {
        { "POSITION", 0, DXGI_FORMAT_R32G32B32A32_FLOAT, 0, UINT(offsetof(TLVertex, p)),
      D3D11_INPUT_PER_VERTEX_DATA, 0 },
        { "DIFFUSE", 0, DXGI_FORMAT_R32G32B32A32_FLOAT, 0, UINT(offsetof(TLVertex, diffuse)),
      D3D11_INPUT_PER_VERTEX_DATA, 0 },
        { "UV", 0, DXGI_FORMAT_R32G32_FLOAT, 0, UINT(offsetof(TLVertex, tu)), D3D11_INPUT_PER_VERTEX_DATA,
      0 }
    }
```

Input element descriptor for a TLVertex

4.1.4.11 vertex_input_element_desc

```
const D3D11_INPUT_ELEMENT_DESC mage::vertex_input_element_desc[]
```

Initial value:

```
= {
        { "POSITION", 0, DXGI_FORMAT_R32G32B32_FLOAT, 0, UINT(offsetof(Vertex, p)),
      D3D11_INPUT_PER_VERTEX_DATA, 0 },
        { "NORMAL", 0, DXGI_FORMAT_R32G32B32_FLOAT, 0, UINT(offsetof(Vertex, n)),
      D3D11_INPUT_PER_VERTEX_DATA, 0 },
        { "UV", 0, DXGI_FORMAT_R32G32_FLOAT, 0, UINT(offsetof(Vertex, tu)), D3D11_INPUT_PER_VERTEX_DATA, 0
      }
    }
```

Input element descriptor for a Vertex.

4.1.4.12 worker_semaphore

```
Semaphore* mage::worker_semaphore  [static]
```

The worker semaphore for being able to work.

Chapter 5

Class Documentation

5.1 mage::AABB Struct Reference

`#include <geometry.hpp>`

Public Member Functions

- AABB ()
- AABB (XMFLOAT3 p_min, XMFLOAT3 p_max)
- bool Encloses (const AABB &aabb) const
- bool Encloses (const Face &face) const
- bool EnclosedBy (const list< XMFLOAT4 > &planes) const

Public Attributes

- XMFLOAT3 p_min
- XMFLOAT3 p_max

5.1.1 Detailed Description

A struct of Axis-Aligned Bounding Boxes (AABBs).

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AABB() [1/2]

```
mage::AABB::AABB ( )
```

Constructs an AABB.

5.1.2.2 AABB() [2/2]

```
mage::AABB::AABB (
            XMFLOAT3 p_min,
            XMFLOAT3 p_max )
```

Constructs an AABB.

Parameters

in	*p_min*	The minimum extents.
in	*p_max*	The maximum extents.

5.1.3 Member Function Documentation

5.1.3.1 EnclosedBy()

```
bool mage::AABB::EnclosedBy (
            const list< XMFLOAT4 > & planes ) const
```

Checks whether this AABB is completely enclosed by the given (closed) volume.

Parameters

in	*planes*	A reference to a linked list containing the planes of the volume (each plane's coefficients are represented as a `XMFLOAT4`).

Returns

`true` if this AABB is completely enclosed by *planes*. `false` otherwise.

5.1.3.2 Encloses() [1/2]

```
bool mage::AABB::Encloses (
            const AABB & aabb ) const
```

Checks whether this AABB completely encloses the given AABB.

Parameters

in	*aabb*	A reference to the AABB.

Returns

`true` if this AABB completely encloses *aabb*. `false` otherwise.

5.1.3.3 Encloses() [2/2]

```
bool mage::AABB::Encloses (
            const Face & face ) const
```

Checks whether this AABB completely encloses the given face.

Parameters

`in`	*face*	A reference to the face.

Returns

`true` if this AABB completely encloses *face*. `false` otherwise.

5.1.4 Member Data Documentation

5.1.4.1 p_max

```
XMFLOAT3 mage::AABB::p_max
```

The maximum extents of this AABB.

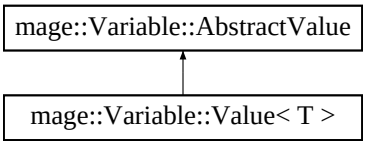
5.1.4.2 p_min

```
XMFLOAT3 mage::AABB::p_min
```

The minimum extents of this AABB.

5.2 mage::Variable::AbstractValue Struct Reference

Inheritance diagram for mage::Variable::AbstractValue:



Public Member Functions

- virtual ~AbstractValue ()
- virtual const void * GetValue () const =0

5.2.1 Detailed Description

A struct of abstract values.

Note

This is an example of the Type Erasure pattern for templates. We need to keep the original type to ensure the right destructor can be called in case of non-primitive types.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 ~AbstractValue()

```
virtual mage::Variable::AbstractValue::~AbstractValue ( )  [virtual]
```

Destructs this value.

5.2.3 Member Function Documentation

5.2.3.1 GetValue()

```
virtual const void* mage::Variable::AbstractValue::GetValue ( ) const  [pure virtual]
```

Returns the value of this value.

Returns

A pointer to the value of this value.

Implemented in mage::Variable::Value< T >.

5.3 mage::ConditionVariable Class Reference

```
#include <lock.hpp>
```

Public Member Functions

- ConditionVariable ()
- ~ConditionVariable ()
- void Lock ()
- void Unlock ()
- void Wait ()
- void Signal ()

Private Types

- enum { SIGNAL = 0, BROADCAST = 1, NUM_EVENTS = 2 }

Private Attributes

- uint32_t m_nb_waiters
- CRITICAL_SECTION m_nb_waiters_mutex
- CRITICAL_SECTION m_condition_mutex
- HANDLE m_events [NUM_EVENTS]

5.3.1 Detailed Description

A class of condition variables.

5.3.2 Member Enumeration Documentation

5.3.2.1 anonymous enum

```
anonymous enum  [private]
```

Type of events (indices).

Enumerator

SIGNAL	
BROADCAST	
NUM_EVENTS	

5.3.3 Constructor & Destructor Documentation

5.3.3.1 ConditionVariable()

```
mage::ConditionVariable::ConditionVariable ( )
```

Constructs a condition variable.

5.3.3.2 ~ConditionVariable()

```
mage::ConditionVariable::~ConditionVariable ( )
```

Destructs this condition variable.

5.3.4 Member Function Documentation

5.3.4.1 Lock()

```
void mage::ConditionVariable::Lock ( )
```

Locks this condition variable.

5.3.4.2 Signal()

```
void mage::ConditionVariable::Signal ( )
```

Signal a condition change.

5.3.4.3 Unlock()

```
void mage::ConditionVariable::Unlock ( )
```

Unlocks this condition variable.

5.3.4.4 Wait()

```
void mage::ConditionVariable::Wait ( )
```

Wait for a signal indicating a condition change.

5.3.5 Member Data Documentation

5.3.5.1 m_condition_mutex

```
CRITICAL_SECTION mage::ConditionVariable::m_condition_mutex  [private]
```

The critical section object for the mutex guarding the condition of this condition variable.

5.3.5.2 m_events

```
HANDLE mage::ConditionVariable::m_events[NUM_EVENTS]  [private]
```

Signal and broadcast event handles of this condition variable.

5.3.5.3 m_nb_waiters

```
uint32_t mage::ConditionVariable::m_nb_waiters  [private]
```

The number of waiters of this condition variable.

5.3.5.4 m_nb_waiters_mutex

```
CRITICAL_SECTION mage::ConditionVariable::m_nb_waiters_mutex  [private]
```

The critical section object for the mutex guarding `m_nb_waiters` of this condition variable.

5.4 mage::DeviceEnumeration Class Reference

```
#include <device_enumeration.hpp>
```

Public Member Functions

- DeviceEnumeration ()
- virtual ~DeviceEnumeration ()
- INT_PTR Enumerate (IDXGIDevice3 *device)
- INT_PTR SettingsDialogProc (HWND hwndDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
- const DXGI_MODE_DESC1 * GetSelectedDisplayMode () const
- bool IsWindowed () const
- bool IsVSynced () const

Private Attributes

- DXGI_ADAPTER_DESC2 m_adapter_desc
- VariableScript * m_settings_script
- list< DisplayMode > m_display_modes
- DXGI_MODE_DESC1 m_selected_diplay_mode
- bool m_windowed
- bool m_vsync

5.4.1 Detailed Description

A device enumeration.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 DeviceEnumeration()

```
mage::DeviceEnumeration::DeviceEnumeration ( )
```

Constructs a device enumeration.

5.4.2.2 ~DeviceEnumeration()

```
virtual mage::DeviceEnumeration::~DeviceEnumeration ( )  [virtual]
```

Destructs this device enumeration.

5.4.3 Member Function Documentation

5.4.3.1 Enumerate()

```
INT_PTR mage::DeviceEnumeration::Enumerate (
            IDXGIDevice3 * device )
```

Enumerates the available display modes on the default adapter output of the physical adapter associated with the given device.

Parameters

in	*device*	A pointer to the device.

5.4.3.2 GetSelectedDisplayMode()

```
const DXGI_MODE_DESC1* mage::DeviceEnumeration::GetSelectedDisplayMode ( ) const
```

Returns the selected display mode by the user.

Returns

A pointer to the selected display mode.

5.4.3.3 IsVSynced()

```
bool mage::DeviceEnumeration::IsVSynced ( ) const
```

Check whether v-sync should be enabled.

Returns

`true` if v-sync should be enabled. `false` otherwise.

5.4.3.4 IsWindowed()

```
bool mage::DeviceEnumeration::IsWindowed ( ) const
```

Check whether the application should run in windowed mode.

Returns

`true` if the application should run in windowed mode. `false` otherwise.

5.4.3.5 SettingsDialogProc()

```
INT_PTR mage::DeviceEnumeration::SettingsDialogProc (
            HWND hwndDlg,
            UINT uMsg,
            WPARAM wParam,
            LPARAM lParam )
```

Engine-defined callback function used with the CreateDialog for device enumeration.

Parameters

in	*hwndDlg*	A handle to the dialog box.
in	*uMsg*	The message.
in	*wParam*	Additional message-specific information.
in	*lParam*	Additional message-specific information.

5.4.4 Member Data Documentation

5.4.4.1 m_adapter_desc

```
DXGI_ADAPTER_DESC2 mage::DeviceEnumeration::m_adapter_desc  [private]
```

The description of the adapter (or video card).

5.4.4.2 m_display_modes

```
list< DisplayMode > mage::DeviceEnumeration::m_display_modes  [private]
```

The linked list of enumerated display modes.

5.4.4.3 m_selected_diplay_mode

```
DXGI_MODE_DESC1 mage::DeviceEnumeration::m_selected_diplay_mode  [private]
```

The selected display mode by the user.

5.4.4.4 m_settings_script

```
VariableScript* mage::DeviceEnumeration::m_settings_script  [private]
```

A pointer to the script which stores the device configuration.

5.4.4.5 m_vsync

```
bool mage::DeviceEnumeration::m_vsync  [private]
```

Flag indicating whether v-sync should be enabled.

5.4.4.6 m_windowed

```
bool mage::DeviceEnumeration::m_windowed  [private]
```

Flag indicating whether the application should run in windowed mode.

5.5 mage::DisplayMode Struct Reference

```
#include <device_enumeration.hpp>
```

Public Attributes

- DXGI_MODE_DESC1 mode
- wchar_t bpp [MAGE_DISPLAYMODE_BPP_COUNT]

5.5.1 Detailed Description

A display mode.

5.5.2 Member Data Documentation

5.5.2.1 bpp

```
wchar_t mage::DisplayMode::bpp[MAGE_DISPLAYMODE_BPP_COUNT]
```

The colour depth expressed as a character string for the display mode.

5.5.2.2 mode

```
DXGI_MODE_DESC1 mage::DisplayMode::mode
```

The display mode descriptor of this display mode.

5.6 mage::Edge Struct Reference

```
#include <geometry.hpp>
```

Public Member Functions

- Edge (Vertex *v0, Vertex *v1)

Public Attributes

- Vertex * v0
- Vertex * v1

5.6.1 Detailed Description

A struct of edges.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Edge()

```
mage::Edge::Edge (
            Vertex * v0,
            Vertex * v1 )
```

Constructs an edge between the two given vertices.

Parameters

in	*v0*	A pointer to the first vertex.
in	*v1*	A pointer to the second vertex.

5.6.3 Member Data Documentation

5.6.3.1 v0

```
Vertex* mage::Edge::v0
```

The first vertex of this edge.

5.6.3.2 v1

```
Vertex* mage::Edge::v1
```

The second vertex of this edge.

5.7 mage::Engine Class Reference

```
#include <engine.hpp>
```

Public Member Functions

- Engine (const EngineSetup *setup=NULL)
- virtual ~Engine ()
- void Run ()
- HWND GetWindow () const
- void SetDeactiveFlag (bool deactive)
- GraphicsManager * GetGraphicsManager () const
- StateManager * GetStateManager () const
- ResourceManager< VariableScript > * GetScriptManager () const
- Input * GetInput () const

Private Attributes

- EngineSetup * m_setup
- bool m_loaded
- HWND m_hwindow
- bool m_deactive
- GraphicsManager * m_graphics_manager
- StateManager * m_state_manager
- ResourceManager< VariableScript > * m_script_manager
- Input * m_input

5.7.1 Detailed Description

A class of engines.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Engine()

```
mage::Engine::Engine (
            const EngineSetup * setup = NULL )
```

Constructs an engine from the given engine setup.

Parameters

in	*setup*	A pointer to an engine setup.

5.7.2.2 ~Engine()

```
mage::Engine::~Engine ( )  [virtual]
```

Destructs this engine.

5.7.3 Member Function Documentation

5.7.3.1 GetGraphicsManager()

```
GraphicsManager* mage::Engine::GetGraphicsManager ( ) const
```

Returns the graphics manager of this engine.

Returns

A pointer to the graphics manager of this engine

5.7.3.2 GetInput()

```
Input* mage::Engine::GetInput ( ) const
```

Returns the input object of this engine.

Returns

A pointer to the input object of this engine

5.7.3.3 GetScriptManager()

```
ResourceManager< VariableScript >* mage::Engine::GetScriptManager ( ) const
```

Returns the script manager of this engine.

Returns

A pointer to the script manager of this engine

5.7.3.4 GetStateManager()

```
StateManager* mage::Engine::GetStateManager ( ) const
```

Returns the state manager of this engine.

Returns

A pointer to the state manager of this engine

5.7.3.5 GetWindow()

```
HWND mage::Engine::GetWindow ( ) const
```

Returns a handle to the window of this engine.

5.7.3.6 Run()

```
void mage::Engine::Run ( )
```

Runs the engine setup.

5.7.3.7 SetDeactiveFlag()

```
void mage::Engine::SetDeactiveFlag (
            bool deactive )
```

Sets the deactive flag of this engine to the given value.

Parameters

in	*deactive*	The new value for the deactive flag.

5.7.4 Member Data Documentation

5.7.4.1 m_deactive

```
bool mage::Engine::m_deactive  [private]
```

Flag indicating whether the application is active or not.

5.7.4.2 m_graphics_manager

```
GraphicsManager* mage::Engine::m_graphics_manager  [private]
```

A pointer to the graphics manager of this engine.

5.7.4.3 m_hwindow

```
HWND mage::Engine::m_hwindow  [private]
```

Main window handle of this engine.

5.7.4.4 m_input

```
Input* mage::Engine::m_input  [private]
```

A pointer to the input object of this engine.

5.7.4.5 m_loaded

```
bool mage::Engine::m_loaded  [private]
```

Flag indicating whether this engine is loaded.

5.7.4.6 m_script_manager

```
ResourceManager< VariableScript >* mage::Engine::m_script_manager  [private]
```

A pointer the script manager of this engine

5.7.4.7 m_setup

```
EngineSetup* mage::Engine::m_setup  [private]
```

Pointer to a copy of the engine setup structure.

5.7.4.8 m_state_manager

```
StateManager* mage::Engine::m_state_manager  [private]
```

A pointer to the state manager of this engine.

5.8 mage::EngineSetup Struct Reference

```
#include <engine.hpp>
```

Public Member Functions

- EngineSetup (const wstring &name=L"Application")
- EngineSetup (const EngineSetup *setup)

Public Attributes

- HINSTANCE m_hinstance
- wstring m_name
- void(* StateSetup)()

5.8.1 Detailed Description

A struct of engine setups.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 EngineSetup() [1/2]

```
mage::EngineSetup::EngineSetup (
            const wstring & name = L"Application" )
```

Constructs an engine setup with the given application name.

Parameters

in	*name*	A reference to the name of the application.

5.8.2.2 EngineSetup() [2/2]

```
mage::EngineSetup::EngineSetup (
            const EngineSetup * setup )
```

Constructs an engine setup from the given engine setup.

Precondition

setup does not point to NULL.

Parameters

in	*setup*	A pointer to the engine setup.

5.8.3 Member Data Documentation

5.8.3.1 m_hinstance

```
HINSTANCE mage::EngineSetup::m_hinstance
```

Application instance handle.

5.8.3.2 m_name

```
wstring mage::EngineSetup::m_name
```

Name of the application.

5.8.3.3 StateSetup

```
void(* mage::EngineSetup::StateSetup) ()
```

The state setup function.

5.9 mage::Face Struct Reference

```
#include <geometry.hpp>
```

Public Member Functions

- Face (Vertex *v0, Vertex *v1, Vertex *v2)

Public Attributes

- Vertex * v0
- Vertex * v1
- Vertex * v2

5.9.1 Detailed Description

A struct of faces.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Face()

```
mage::Face::Face (
            Vertex * v0,
            Vertex * v1,
            Vertex * v2 )
```

Constructs a face for the three given vertices.

Parameters

in	*v0*	A pointer to the first vertex.
in	*v1*	A pointer to the second vertex.
in	*v2*	A pointer to the third vertex.

5.9.3 Member Data Documentation

5.9.3.1 v0

```
Vertex* mage::Face::v0
```

The first vertex of this face.

5.9.3.2 v1

```
Vertex* mage::Face::v1
```

The second vertex of this face.

5.9.3.3 v2

```
Vertex* mage::Face::v2
```

The third vertex of this face.

5.10 mage::GeneralConfiguration Struct Reference

```
#include <engine.hpp>
```

Public Member Functions

- GeneralConfiguration ()
- bool IsQuiet () const
- bool IsVerbose () const

Public Attributes

- bool m_quiet
- bool m_verbose

5.10.1 Detailed Description

A struct of general configurations (of the logging) of the engine processing.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 GeneralConfiguration()

```
mage::GeneralConfiguration::GeneralConfiguration ( )
```

Constructs a new general configuration.

5.10.3 Member Function Documentation

5.10.3.1 IsQuiet()

```
bool mage::GeneralConfiguration::IsQuiet ( ) const
```

Checks whether the logging of the engine processing is quiet.

Returns

`true` if the logging of the engine processing is quiet. `false` otherwise.

5.10.3.2 IsVerbose()

```
bool mage::GeneralConfiguration::IsVerbose ( ) const
```

Checks wheter the logging of the engine processing is verbose.

Returns

`true` if the logging of the engine processing is verbose. `false` otherwise.

5.10.4 Member Data Documentation

5.10.4.1 m_quiet

```
bool mage::GeneralConfiguration::m_quiet
```

Flag indicating the logging of the engine processing is quiet.

5.10.4.2 m_verbose

```
bool mage::GeneralConfiguration::m_verbose
```

Flag indicating the logging of the engine processing is verbose.

5.11 mage::GraphicsManager Class Reference

```
#include <graphics_manager.hpp>
```

Public Member Functions

- GraphicsManager ()
- virtual ~GraphicsManager ()
- IDXGIDevice3 * GetDevice () const
- DXGI_MODE_DESC1 GetDisplayMode () const

Private Attributes

- IDXGIDevice3 * m_device
- DXGI_MODE_DESC1 m_display_mode

5.11.1 Constructor & Destructor Documentation

5.11.1.1 GraphicsManager()

```
mage::GraphicsManager::GraphicsManager ( )
```

5.11.1.2 ~GraphicsManager()

```
virtual mage::GraphicsManager::~GraphicsManager ( )  [virtual]
```

5.11.2 Member Function Documentation

5.11.2.1 GetDevice()

```
IDXGIDevice3* mage::GraphicsManager::GetDevice ( ) const
```

5.11.2.2 GetDisplayMode()

```
DXGI_MODE_DESC1 mage::GraphicsManager::GetDisplayMode ( ) const
```

5.11.3 Member Data Documentation

5.11.3.1 m_device

```
IDXGIDevice3* mage::GraphicsManager::m_device  [private]
```

5.11.3.2 m_display_mode

```
DXGI_MODE_DESC1 mage::GraphicsManager::m_display_mode  [private]
```

5.12 mage::IndexedEdge Struct Reference

```
#include <geometry.hpp>
```

Public Attributes

- uint16_t iv0
- uint16_t iv1

5.12.1 Detailed Description

A struct of indexed edges.

5.12.2 Member Data Documentation

5.12.2.1 iv0

```
uint16_t mage::IndexedEdge::iv0
```

The index of the edge's first vertex.

5.12.2.2 iv1

```
uint16_t mage::IndexedEdge::iv1
```

The index of the edge's second vertex.

5.13 mage::IndexedFace Struct Reference

```
#include <geometry.hpp>
```

Public Attributes

- uint16_t iv0
- uint16_t iv1
- uint16_t iv2

5.13.1 Detailed Description

A struct of indexed faces.

5.13.2 Member Data Documentation

5.13.2.1 iv0

```
uint16_t mage::IndexedFace::iv0
```

Index of the face's first vertex.

5.13.2.2 iv1

```
uint16_t mage::IndexedFace::iv1
```

Index of the face's second vertex.

5.13.2.3 iv2

```
uint16_t mage::IndexedFace::iv2
```

Index of the face's third vertex.

5.14 mage::Input Class Reference

```
#include <input.hpp>
```

Public Member Functions

- Input (HWND hwindow)
- virtual ~Input ()
- void Update ()
- bool GetKeyPress (char key, bool ignore_press_stamp=false)
- bool GetMouseButtonPress (char mouse_button, bool ignore_press_stamp=false)
- long GetPosX () const
- long GetPosY () const
- long GetDeltaX () const
- long GetDeltaY () const
- long GetDeltaWheel () const

Private Attributes

- HWND m_hwindow
- IDirectInput8 * m_di
- uint64_t m_press_stamp
- IDirectInputDevice8 * m_keyboard
- char m_key_state [256]
- uint64_t m_key_press_stamp [256]
- IDirectInputDevice8 * m_mouse
- DIMOUSESTATE m_mouse_state
- uint64_t m_mouse_button_press_stamp [3]
- POINT m_mouse_position

5.14.1 Detailed Description

A class of input objects.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 Input()

```
mage::Input::Input (
            HWND hwindow )
```

Constructs an input for the given window handle.

Parameters

in	*hwindow*	The handle of the parent window.

5.14.2.2 ~Input()

```
virtual mage::Input::~Input ( )  [virtual]
```

Destructs this input object.

5.14.3 Member Function Documentation

5.14.3.1 GetDeltaWheel()

```
long mage::Input::GetDeltaWheel ( ) const
```

Returns the change in the mouse's scroll wheel.

Returns

The change in the mouse's mouse's scroll wheel.

5.14.3.2 GetDeltaX()

```
long mage::Input::GetDeltaX ( ) const
```

Returns the change in the mouse's horizontal coordinate.

Returns

The change in the mouse's horizontal coordinate.

5.14.3.3 GetDeltaY()

```
long mage::Input::GetDeltaY ( ) const
```

Returns the change in the mouse's vertical coordinate.

Returns

The change in the mouse's vertical coordinate.

5.14.3.4 GetKeyPress()

```
bool mage::Input::GetKeyPress (
            char key,
            bool ignore_press_stamp = false )
```

Checks whether the given key is pressed.

Parameters

in	*key*	The key.
in	*ignore_press_stamp*	Flag indicating whether press stamps should be ignored. Consistent presses will return false when using the press stamp.

Returns

`true` if the given key is pressed. `false` otherwise.

5.14.3.5 GetMouseButtonPress()

```
bool mage::Input::GetMouseButtonPress (
            char mouse_button,
            bool ignore_press_stamp = false )
```

Checks whether the given mouse button is pressed.

Parameters

in	*mouse_button*	The mouse button.
in	*ignore_press_stamp*	Flag indicating whether press stamps should be ignored. Consistent presses will return false when using the press stamp.

Returns

`true` if the given mouse button is pressed. `false` otherwise.

5.14.3.6 GetPosX()

```
long mage::Input::GetPosX ( ) const
```

Returns the horizontal position of the mouse.

Returns

The horizontal position of the mouse.

5.14.3.7 GetPosY()

```
long mage::Input::GetPosY ( ) const
```

Returns the vertical position of the mouse.

Returns

The vertical position of the mouse.

5.14.3.8 Update()

```
void mage::Input::Update ( )
```

Updates the state of both the keyboard and mouse device of this input object.

5.14.4 Member Data Documentation

5.14.4.1 m_di

```
IDirectInput8* mage::Input::m_di  [private]
```

The DirectInput object.

The methods of the IDirectInput8 interface are used to enumerate, create, and retrieve the status of Microsoft DirectInput device.

5.14.4.2 m_hwindow

```
HWND mage::Input::m_hwindow  [private]
```

The handle of the parent window.

5.14.4.3 m_key_press_stamp

```
uint64_t mage::Input::m_key_press_stamp[256]  [private]
```

Stamps the keys pressed in the last frame.

5.14.4.4 m_key_state

```
char mage::Input::m_key_state[256]  [private]
```

State of the keys.

5.14.4.5 m_keyboard

```
IDirectInputDevice8* mage::Input::m_keyboard  [private]
```

The DirectInput keyboard device.

The methods of the IDirectInputDevice8 interface are used to gain and release access to Microsoft DirectInput devices, manage device properties and information, set behavior, perform initialization, create and play force-feedback effects, and invoke a device's control panel.

5.14.4.6 m_mouse

```
IDirectInputDevice8* mage::Input::m_mouse  [private]
```

DirectInput mouse device.

The methods of the IDirectInputDevice8 interface are used to gain and release access to Microsoft DirectInput devices, manage device properties and information, set behavior, perform initialization, create and play force-feedback effects, and invoke a device's control panel.

5.14.4.7 m_mouse_button_press_stamp

```
uint64_t mage::Input::m_mouse_button_press_stamp[3]  [private]
```

Stamps the mouse buttons pressed in the last frame.

5.14.4.8 m_mouse_position

```
POINT mage::Input::m_mouse_position  [private]
```

The position of the mouse cursor on the screen.

5.14.4.9 m_mouse_state

```
DIMOUSESTATE mage::Input::m_mouse_state  [private]
```

State of the mouse buttons.

Describes the state of a mouse device that has up to four buttons, or another device that is being accessed as if it were a mouse device.

5.14.4.10 m_press_stamp

```
uint64_t mage::Input::m_press_stamp  [private]
```

The current press stamp (incremented every frame).

5.15 mage::LVertex Struct Reference

```
#include <geometry.hpp>
```

Public Member Functions

- LVertex ()
- LVertex (XMFLOAT3 p, XMFLOAT4 diffuse, float tu, float tv)

Public Attributes

- XMFLOAT3 p
- XMFLOAT4 diffuse
- float tu
- float tv

5.15.1 Detailed Description

A struct of lit vertices.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 LVertex() [1/2]

```
mage::LVertex::LVertex ( )
```

Constructs a lit vertex.

5.15.2.2 LVertex() [2/2]

```
mage::LVertex::LVertex (
            XMFLOAT3 p,
            XMFLOAT4 diffuse,
            float tu,
            float tv )
```

Constructs a lit vertex.

Parameters

in	*p*	Position of the lit vertex (in world space).
in	*diffuse*	Diffuse colour of the lit vertex.
in	*tu*	Texture u coordinate of the lit vertex.
in	*tv*	Texture v coordinate of the lit vertex.

5.15.3 Member Data Documentation

5.15.3.1 diffuse

```
XMFLOAT4 mage::LVertex::diffuse
```

Diffuse colour of this lit vertex.

5.15.3.2 p

```
XMFLOAT3 mage::LVertex::p
```

Position of this lit vertex (in world space).

5.15.3.3 tu

```
float mage::LVertex::tu
```

Texture u coordinate of this lit vertex.

5.15.3.4 tv

```
float mage::LVertex::tv
```

Texture v coordinate of this lit vertex.

5.16 mage::MemoryArena Class Reference

```
#include <arena.hpp>
```

Public Member Functions

- MemoryArena (uint32_t block_size=32768)
- ~MemoryArena ()
- void FreeAll ()
- void * Alloc (uint32_t size)
- template<typename T >
  T * Alloc (uint32_t count=1)

Private Attributes

- uint32_t m_current_block_pos
- const uint32_t m_block_size
- char * m_current_block
- vector< char * > m_used_blocks
- vector< char * > m_available_blocks

5.16.1 Detailed Description

A class of memory arena's.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 MemoryArena()

```
mage::MemoryArena::MemoryArena (
            uint32_t block_size = 32768 )
```

Constructs a memory arena with given block size.

Parameters

in	*block_size*	The block size in bytes.

5.16.2.2 ~MemoryArena()

```
mage::MemoryArena::~MemoryArena ( )
```

Destructs the given memory arena.

5.16.3 Member Function Documentation

5.16.3.1 Alloc() [1/2]

```
void* mage::MemoryArena::Alloc (
            uint32_t size )
```

Allocates a block of memory of the given size.

Parameters

in	*size*	The requested size in bytes to allocate in memory.

Returns

NULL if the allocation failed.
A pointer to the memory block that was allocated.

5.16.3.2 Alloc() [2/2]

```
template<typename T >
T* mage::MemoryArena::Alloc (
            uint32_t count = 1 )
```

Allocates a block of memory.

Template Parameters

T	The type of objects to allocate in memory.

Parameters

in	*count*	The number of objects of type T to allocate in memory.

Returns

`NULL` if the allocation failed.
A pointer to the memory block that was allocated.

Note

The objects will be constructed with their default empty constructor.

5.16.3.3 FreeAll()

```
void mage::MemoryArena::FreeAll ( )
```

Frees all blocks of this memory arena.

5.16.4 Member Data Documentation

5.16.4.1 m_available_blocks

```
vector<char *> mage::MemoryArena::m_available_blocks  [private]
```

Pointers to the available blocks of this memory arena.

5.16.4.2 m_block_size

```
const uint32_t mage::MemoryArena::m_block_size  [private]
```

The fixed block size of this memory arena.

5.16.4.3 m_current_block

```
char* mage::MemoryArena::m_current_block  [private]
```

A pointer to the current block of this memory arena.

5.16.4.4 m_current_block_pos

```
uint32_t mage::MemoryArena::m_current_block_pos  [private]
```

The current block position of this memory arena.

5.16.4.5 m_used_blocks

```
vector<char *> mage::MemoryArena::m_used_blocks  [private]
```

Pointers to the used blocks of this memory arena.

5.17 mage::Mutex Class Reference

`#include <lock.hpp>`

Static Public Member Functions

- static Mutex * Create ()
- static void Destroy (Mutex *mutex)

Private Member Functions

- Mutex ()
- Mutex (Mutex &mutex)
- ~Mutex ()
- Mutex & operator= (const Mutex &mutex)

Private Attributes

- CRITICAL_SECTION m_critical_section

Friends

- struct MutexLock

5.17.1 Detailed Description

A class of mutexes.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 Mutex() [1/2]

```
mage::Mutex::Mutex ( )  [private]
```

Constructs a mutex.

5.17.2.2 Mutex() [2/2]

```
mage::Mutex::Mutex (
            Mutex & mutex )  [private]
```

Constructs a mutex from the given mutex.

Parameters

in	*mutex*	A reference to a mutex.

5.17.2.3 ~Mutex()

```
mage::Mutex::~Mutex ( )  [private]
```

Destructs this mutex.

5.17.3 Member Function Documentation

5.17.3.1 Create()

```
static Mutex* mage::Mutex::Create ( )  [static]
```

Creates a mutex.

5.17.3.2 Destroy()

```
static void mage::Mutex::Destroy (
            Mutex * mutex )  [static]
```

Destroys a given mutex.

Parameters

in	*mutex*	The mutex to destroy.

5.17.3.3 operator=()

```
Mutex& mage::Mutex::operator= (
            const Mutex & mutex )  [private]
```

Copies the given mutex to this mutex.

Parameters

in	*mutex*	A reference to a mutex.

Returns

A reference to the copy of *mutex*.

5.17.4 Friends And Related Function Documentation

5.17.4.1 MutexLock

```
friend struct MutexLock  [friend]
```

5.17.5 Member Data Documentation

5.17.5.1 m_critical_section

```
CRITICAL_SECTION mage::Mutex::m_critical_section  [private]
```

The critical section object of this mutex.

5.18 mage::MutexLock Struct Reference

```
#include <lock.hpp>
```

Public Member Functions

- MutexLock (Mutex &mutex)
- ~MutexLock ()

Private Member Functions

- MutexLock (const MutexLock &mutex_lock)
- MutexLock & operator= (const MutexLock &mutex_lock)

Private Attributes

- Mutex & m_mutex

5.18.1 Detailed Description

A struct of mutex locks.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 MutexLock() [1/2]

```
mage::MutexLock::MutexLock (
            Mutex & mutex )
```

Constructs a mutex lock for the given mutex.

Parameters

in	*mutex*	A reference to a mutex.

5.18.2.2 ~MutexLock()

```
mage::MutexLock::~MutexLock ( )
```

Destructs this mutex lock.

5.18.2.3 MutexLock() [2/2]

```
mage::MutexLock::MutexLock (
            const MutexLock & mutex_lock )  [private]
```

Constructs a mutex lock from the given mutex lock.

Parameters

in	*mutex_lock*	A reference to a mutex lock.

5.18.3 Member Function Documentation

5.18.3.1 operator=()

```
MutexLock& mage::MutexLock::operator= (
            const MutexLock & mutex_lock )  [private]
```

Copies the given mutex lock to this mutex lock.

Parameters

in	*mutex_lock*	A reference to a mutex lock.

Returns

A reference to the copy of *mutex_lock*.

5.18.4 Member Data Documentation

5.18.4.1 m_mutex

```
Mutex& mage::MutexLock::m_mutex  [private]
```

The mutex of this mutex lock.

5.19 mage::ProgressReporter Class Reference

`#include <progressreporter.hpp>`

Public Member Functions

- ProgressReporter (uint32_t nb_work, const string &title, uint32_t bar_length=0)
- virtual ~ProgressReporter ()
- void Update (uint32_t nb_work=1)
- void Done ()

Private Attributes

- const uint32_t m_nb_work_total
- uint32_t m_nb_work_done
- uint32_t m_nb_plusses_total
- uint32_t m_nb_plusses_printed
- Timer * m_timer
- FILE * m_fout
- char * m_buffer
- char * m_current_pos
- Mutex * m_mutex

5.19.1 Detailed Description

A class of progress reporters.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 ProgressReporter()

```
mage::ProgressReporter::ProgressReporter (
            uint32_t nb_work,
            const string & title,
            uint32_t bar_length = 0 )
```

Constructs a progress reporter.

Parameters

in	*nb_work*	The number of parts of the total work.
in	*title*	A reference to the title.
in	*bar_length*	The length of the progress bar. If 0 the default length will be chosen.

5.19.2.2 ~ProgressReporter()

```
mage::ProgressReporter::~ProgressReporter ( )  [virtual]
```

Destructs this progress reporter.

5.19.3 Member Function Documentation

5.19.3.1 Done()

```
void mage::ProgressReporter::Done ( )
```

Finishes this progress reporter.

5.19.3.2 Update()

```
void mage::ProgressReporter::Update (
            uint32_t nb_work = 1 )
```

Updates this progress reporter.

Parameters

in	*nb_work*	The number of parts of the total work that are done.

5.19.4 Member Data Documentation

5.19.4.1 m_buffer

```
char* mage::ProgressReporter::m_buffer  [private]
```

The output buffer of this progress reporter.

5.19.4.2 m_current_pos

```
char* mage::ProgressReporter::m_current_pos  [private]
```

The current (output) position of this progress reporter.

5.19.4.3 m_fout

```
FILE* mage::ProgressReporter::m_fout  [private]
```

The output file stream of this progress reporter.

5.19.4.4 m_mutex

```
Mutex* mage::ProgressReporter::m_mutex  [private]
```

The mutex needed for updating this progress reporter.

5.19.4.5 m_nb_plusses_printed

```
uint32_t mage::ProgressReporter::m_nb_plusses_printed  [private]
```

The total number of plusses that are already outputted.

5.19.4.6 m_nb_plusses_total

```
uint32_t mage::ProgressReporter::m_nb_plusses_total  [private]
```

The total number of plusses to output.

5.19.4.7 m_nb_work_done

```
uint32_t mage::ProgressReporter::m_nb_work_done  [private]
```

The number of parts of the total work that are already done.

5.19.4.8 m_nb_work_total

```
const uint32_t mage::ProgressReporter::m_nb_work_total  [private]
```

The number of parts of the total work.

5.19.4.9 m_timer

```
Timer* mage::ProgressReporter::m_timer  [private]
```

The timer of this progress reporter.

5.20 mage::ReadWriteMutex Class Reference

```
#include <lock.hpp>
```

Static Public Member Functions

- static ReadWriteMutex * Create ()
- static void Destroy (ReadWriteMutex *mutex)

Private Member Functions

- ReadWriteMutex ()
- ReadWriteMutex (ReadWriteMutex &mutex)
- ~ReadWriteMutex ()
- ReadWriteMutex & operator= (const ReadWriteMutex &mutex)
- void AcquireRead ()
- void ReleaseRead ()
- void AcquireWrite ()
- void ReleaseWrite ()

Private Attributes

- LONG m_nb_writers_waiting
- LONG m_nb_readers_waiting
- DWORD m_active_writer_readers
- HANDLE m_ready_to_read_handle
- HANDLE m_ready_to_write_handle
- CRITICAL_SECTION m_critical_section

Friends

- struct ReadWriteMutexLock

5.20.1 Detailed Description

A class of read write mutexes.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 ReadWriteMutex() [1/2]

```
mage::ReadWriteMutex::ReadWriteMutex ( )  [private]
```

Constructs a read write mutex.

5.20.2.2 ReadWriteMutex() [2/2]

```
mage::ReadWriteMutex::ReadWriteMutex (
            ReadWriteMutex & mutex )  [private]
```

Constructs a read write mutex from the given read write mutex.

Parameters

in	*mutex*	The read write mutex.

5.20.2.3 ~ReadWriteMutex()

```
mage::ReadWriteMutex::~ReadWriteMutex ( )  [private]
```

Destructs this read write mutex.

5.20.3 Member Function Documentation

5.20.3.1 AcquireRead()

```
void mage::ReadWriteMutex::AcquireRead ( )  [private]
```

Acquires a read.

5.20.3.2 AcquireWrite()

```
void mage::ReadWriteMutex::AcquireWrite ( )  [private]
```

Acquires a write.

5.20.3.3 Create()

```
static ReadWriteMutex* mage::ReadWriteMutex::Create ( )  [static]
```

Creates a mutex.

5.20.3.4 Destroy()

```
static void mage::ReadWriteMutex::Destroy (
            ReadWriteMutex * mutex )  [static]
```

Destroys a given read write mutex.

Parameters

in	*mutex*	The read write mutex to destroy.

5.20.3.5 operator=()

```
ReadWriteMutex& mage::ReadWriteMutex::operator= (
            const ReadWriteMutex & mutex )  [private]
```

Copies the given read write mutex to this read write mutex.

Parameters

in	*mutex*	A reference to a read write mutex.

Returns

A reference to the copy of *mutex*.

5.20.3.6 ReleaseRead()

```
void mage::ReadWriteMutex::ReleaseRead ( )  [private]
```

Release a read.

5.20.3.7 ReleaseWrite()

```
void mage::ReadWriteMutex::ReleaseWrite ( )  [private]
```

Release a write.

5.20.4 Friends And Related Function Documentation

5.20.4.1 ReadWriteMutexLock

```
friend struct ReadWriteMutexLock  [friend]
```

5.20.5 Member Data Documentation

5.20.5.1 m_active_writer_readers

```
DWORD mage::ReadWriteMutex::m_active_writer_readers  [private]
```

The active group of this read write mutex lock.

HIWORD is the flag indicating a writer is active. LOWORD is the number of active readers.

5.20.5.2 m_critical_section

```
CRITICAL_SECTION mage::ReadWriteMutex::m_critical_section  [private]
```

The critical section object of this read write mutex.

5.20.5.3 m_nb_readers_waiting

```
LONG mage::ReadWriteMutex::m_nb_readers_waiting  [private]
```

The number of readers waiting for this read write mutex lock.

5.20.5.4 m_nb_writers_waiting

```
LONG mage::ReadWriteMutex::m_nb_writers_waiting  [private]
```

The number of writers waiting for this read write mutex lock.

5.20.5.5 m_ready_to_read_handle

```
HANDLE mage::ReadWriteMutex::m_ready_to_read_handle  [private]
```

The handle of this read write mutex lock if ready for reading.

5.20.5.6 m_ready_to_write_handle

```
HANDLE mage::ReadWriteMutex::m_ready_to_write_handle  [private]
```

The handle of this read write mutex lock if ready for writing.

5.21 mage::ReadWriteMutexLock Struct Reference

```
#include <lock.hpp>
```

Public Member Functions

- ReadWriteMutexLock (ReadWriteMutex &mutex, ReadWriteMutexLockType lock_type)
- ~ReadWriteMutexLock ()
- void UpgradeToWrite ()
- void DowngradeToRead ()

Private Member Functions

- ReadWriteMutexLock (const ReadWriteMutexLock &mutex_lock)
- ReadWriteMutexLock & operator= (const ReadWriteMutexLock &mutex_lock)

Private Attributes

- ReadWriteMutexLockType m_type
- ReadWriteMutex & m_mutex

5.21.1 Detailed Description

A struct of read write mutex locks.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 ReadWriteMutexLock() [1/2]

```
mage::ReadWriteMutexLock::ReadWriteMutexLock (
            ReadWriteMutex & mutex,
            ReadWriteMutexLockType lock_type )
```

Constructs a read write mutex lock for the given read write mutex and lock type.

Parameters

in	*mutex*	A reference to a read write mutex.
in	*lock_type*	The lock type.

5.21.2.2 ~ReadWriteMutexLock()

```
mage::ReadWriteMutexLock::~ReadWriteMutexLock ( )
```

Destructs this read write mutex lock.

5.21.2.3 ReadWriteMutexLock() [2/2]

```
mage::ReadWriteMutexLock::ReadWriteMutexLock (
            const ReadWriteMutexLock & mutex_lock )  [private]
```

Constructs a read write mutex lock from the given read write mutex lock.

Parameters

in	*mutex_lock*	A reference to a read write mutex lock.

5.21.3 Member Function Documentation

5.21.3.1 DowngradeToRead()

```
void mage::ReadWriteMutexLock::DowngradeToRead ( )
```

Downgrades this read write lock to read.

5.21.3.2 operator=()

```
ReadWriteMutexLock& mage::ReadWriteMutexLock::operator= (
            const ReadWriteMutexLock & mutex_lock )  [private]
```

Copies the given read write mutex lock to this read write mutex lock.

Parameters

in	*mutex_lock*	A reference to a read write mutex lock.

Returns

A reference to the copy of *mutex_lock*.

5.21.3.3 UpgradeToWrite()

```
void mage::ReadWriteMutexLock::UpgradeToWrite ( )
```

Upgrades this read write lock to write.

5.21.4 Member Data Documentation

5.21.4.1 m_mutex

```
ReadWriteMutex& mage::ReadWriteMutexLock::m_mutex  [private]
```

The read write mutex of this read write mutex lock.

5.21.4.2 m_type

```
ReadWriteMutexLockType mage::ReadWriteMutexLock::m_type  [private]
```

The lock type of this read write mutex lock.

5.22 mage::Reference< T > Class Template Reference

```
#include <reference.hpp>
```

Public Member Functions

- Reference (T *ptr=NULL)
- Reference (const Reference< T > &reference)
- virtual ~Reference ()
- Reference & operator= (T *ptr)
- Reference & operator= (const Reference< T > &reference)
- T * operator-> ()
- const T * operator-> () const
- const T * GetPtr () const
- operator bool () const

Private Attributes

- T * m_ptr

5.22.1 Detailed Description

template<typename T>
class mage::Reference< T >

A class of references.

Template Parameters

T	The type of reference.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 Reference() [1/2]

```
template<typename T>
mage::Reference< T >::Reference (
            T * ptr = NULL )
```

Constructs a reference for the given pointer.

Parameters

in	*ptr*	The pointer.

5.22.2.2 Reference() [2/2]

```
template<typename T>
mage::Reference< T >::Reference (
            const Reference< T > & reference )
```

Constructs a reference from the given reference.

Parameters

in	*reference*	The reference.

5.22.2.3 ~Reference()

```
template<typename T>
virtual mage::Reference< T >::~Reference ( )  [virtual]
```

Destructs this reference.

5.22.3 Member Function Documentation

5.22.3.1 GetPtr()

```
template<typename T>
const T* mage::Reference< T >::GetPtr ( ) const
```

Returns the pointer of this reference.

Returns

The pointer of this reference.

5.22.3.2 operator bool()

```
template<typename T>
mage::Reference< T >::operator bool ( ) const
```

Checks whether the pointer of this reference does not point to `NULL`.

Returns

`true` if the pointer of this reference does not point to `NULL`. `false` otherwise.

5.22.3.3 operator->() [1/2]

```
template<typename T>
T* mage::Reference< T >::operator-> ( )
```

Dereferences this reference.

Returns

The pointer of this reference.

5.22.3.4 operator->() [2/2]

```
template<typename T>
const T* mage::Reference< T >::operator-> ( ) const
```

Dereferences this reference.

Returns

The pointer of this reference.

5.22.3.5 operator=() [1/2]

```
template<typename T>
Reference& mage::Reference< T >::operator= (
            T * ptr )
```

Copies the given pointer into a reference.

Parameters

in	*ptr*	The pointer.

Returns

A reference for *ptr*.

5.22.3.6 operator=() [2/2]

```
template<typename T>
Reference& mage::Reference< T >::operator= (
            const Reference< T > & reference )
```

Copies the given reference into a reference.

Parameters

in	*reference*	The reference.

Returns

A reference for *reference*.

5.22.4 Member Data Documentation

5.22.4.1 m_ptr

```
template<typename T>
T* mage::Reference< T >::m_ptr  [private]
```

The pointer of this reference.

5.23 mage::ReferenceCounted Class Reference

```
#include <reference.hpp>
```

Public Member Functions

- uint32_t IncrementReferenceCount ()
- uint32_t DecrementReferenceCount ()

Protected Member Functions

- ReferenceCounted ()

Private Attributes

- AtomicInt32 m_reference_count

5.23.1 Detailed Description

A class of reference counted objects.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 ReferenceCounted()

```
mage::ReferenceCounted::ReferenceCounted ( )  [protected]
```

Constructs a reference counted object.

5.23.3 Member Function Documentation

5.23.3.1 DecrementReferenceCount()

```
uint32_t mage::ReferenceCounted::DecrementReferenceCount ( )
```

Decrements the reference count of this reference counted object.

Returns

The final reference count of this reference counted object.

5.23.3.2 IncrementReferenceCount()

```
uint32_t mage::ReferenceCounted::IncrementReferenceCount ( )
```

Increments the reference count of this reference counted object.

Returns

The final reference count of this reference counted object.

5.23.4 Member Data Documentation

5.23.4.1 m_reference_count

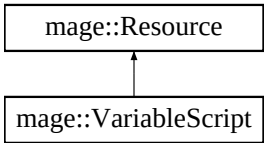
```
AtomicInt32 mage::ReferenceCounted::m_reference_count  [private]
```

The reference count of this reference counted object.

5.24 mage::Resource Class Reference

`#include <resource.hpp>`

Inheritance diagram for mage::Resource:



Public Member Functions

- Resource (const string &name, const string &path="./")
- virtual ~Resource ()
- const string & GetName () const
- const string & GetPath () const
- const string GetFilename () const

Private Member Functions

- uint32_t IncrementResourceReferenceCount ()
- uint32_t DecrementResourceReferenceCount ()

Private Attributes

- const string m_name
- const string m_path
- AtomicInt32 m_resource_reference_count

Friends

- template<typename T >
  class ResourceManager

5.24.1 Detailed Description

A class of resources.

5.24.2 Constructor & Destructor Documentation

5.24.2.1 Resource()

```
mage::Resource::Resource (
            const string & name,
            const string & path = "./" )
```

Constructs a resource with a given name and path.

Parameters

in	*name*	A reference to the name.
in	*path*	A reference to the path.

5.24.2.2 ~Resource()

```
virtual mage::Resource::~Resource ( )  [virtual]
```

5.24.3 Member Function Documentation

5.24.3.1 DecrementResourceReferenceCount()

```
uint32_t mage::Resource::DecrementResourceReferenceCount ( )  [private]
```

Decrements the resource reference count of this reference counted object.

Returns

The final resource reference count of this reference counted object.

5.24.3.2 GetFilename()

```
const string mage::Resource::GetFilename ( ) const
```

Returns the filename of this resource.

Returns

The filename of this resource.

5.24.3.3 GetName()

```
const string& mage::Resource::GetName ( ) const
```

Returns the name of this resource.

Returns

A reference to the name of this resource.

5.24.3.4 GetPath()

```
const string& mage::Resource::GetPath ( ) const
```

Returns the path of this resource.

Returns

A reference to the path of this resource.

5.24.3.5 IncrementResourceReferenceCount()

```
uint32_t mage::Resource::IncrementResourceReferenceCount ( )  [private]
```

Increments the resource reference count of this reference counted object.

Returns

The final resource reference count of this reference counted object.

5.24.4 Friends And Related Function Documentation

5.24.4.1 ResourceManager

```
template<typename T >
friend class ResourceManager  [friend]
```

5.24.5 Member Data Documentation

5.24.5.1 m_name

```
const string mage::Resource::m_name  [private]
```

The name of this resource.

5.24.5.2 m_path

```
const string mage::Resource::m_path  [private]
```

The path of this resource.

5.24.5.3 m_resource_reference_count

```
AtomicInt32 mage::Resource::m_resource_reference_count  [private]
```

The resource reference count of this resource.

5.25 mage::ResourceManager< T > Class Template Reference

`#include <resource.hpp>`

Public Member Functions

- ResourceManager (void(*CreateResourceFunction)(T **resource, const string &name, const string &path)=NULL)
- virtual ~ResourceManager ()
- T * AddResource (const string &name, const string &path="./")
- void RemoveResource (T *resource)
- void ClearResources ()
- T * GetResource (const string &name, const string &path="./") const

Private Attributes

- list< T *> m_resources
- void(* CreateResource)(T **resource, const string &name, const string &path)

5.25.1 Detailed Description

template<typename T>
class mage::ResourceManager< T >

A class of resource managers.

Template Parameters

T	The type of resources.

5.25.2 Constructor & Destructor Documentation

5.25.2.1 ResourceManager()

```
template<typename T>
mage::ResourceManager< T >::ResourceManager (
            void(*)(T **resource, const string &name, const string &path) CreateResource↩
Function = NULL )
```

Constructs a resource manager.

Parameters

in	*CreateResourceFunction*	The application specific resource creation function.

5.25.2.2 ~ResourceManager()

```
template<typename T>
virtual mage::ResourceManager< T >::~ResourceManager ( )  [virtual]
```

Destructs this resource manager.

5.25.3 Member Function Documentation

5.25.3.1 AddResource()

```
template<typename T>
T* mage::ResourceManager< T >::AddResource (
            const string & name,
            const string & path = "./" )
```

Adds a new resource to this resource manager.

Parameters

in	*name*	A reference to the name of the new resource.
in	*path*	A reference to the path of the new resource.

Returns

A pointer to the resource.

5.25.3.2 ClearResources()

```
template<typename T>
void mage::ResourceManager< T >::ClearResources ( )
```

Removes and destructs all the resources from this resource manager, and leaving the resource manager with no resources.

5.25.3.3 GetResource()

```
template<typename T>
T* mage::ResourceManager< T >::GetResource (
            const string & name,
            const string & path = "./" ) const
```

Returns a resource of this resource manager by its filename (given name and path).

Parameters

in	*name*	A reference to the name of the new resource.
in	*path*	A reference to the path of the new resource.

Returns

NULL if the resource is not present.
A pointer to the resource.

5.25.3.4 RemoveResource()

```
template<typename T>
void mage::ResourceManager< T >::RemoveResource (
            T * resource )
```

Removes the given resource from this resource manager.

Parameters

in,out	*resource*	A pointer to the resource.

5.25.4 Member Data Documentation

5.25.4.1 CreateResource

```
template<typename T>
void(* mage::ResourceManager< T >::CreateResource) (T **resource, const string &name, const
string &path)  [private]
```

The application specific resource creation function for the resources of this resource manager.

5.25.4.2 m_resources

```
template<typename T>
list< T * > mage::ResourceManager< T >::m_resources  [private]
```

The linked list containing the resources of this resource manager.

5.26 mage::Semaphore Class Reference

```
#include <lock.hpp>
```

Public Member Functions

- Semaphore ()
- ~Semaphore ()
- void Post (uint32_t count=1)
- void Wait ()
- bool TryWait ()

Private Attributes

- HANDLE m_handle

5.26.1 Detailed Description

A class of semaphores.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 Semaphore()

```
mage::Semaphore::Semaphore ( )
```

Constructs a semaphore.

5.26.2.2 ~Semaphore()

```
mage::Semaphore::~Semaphore ( )
```

Destructs this semaphore.

5.26.3 Member Function Documentation

5.26.3.1 Post()

```
void mage::Semaphore::Post (
            uint32_t count = 1 )
```

Increments the value of this semaphore variable by the given value.

The process executing wait is blocked until the value of the semaphore is greater or equal to 1.

Parameters

in	*count*	The increment value.

5.26.3.2 TryWait()

```
bool mage::Semaphore::TryWait ( )
```

Checks whether waiting for this semaphore would be necessary.

Returns

`true` if waiting for this semaphore would be necessary. `false` otherwise.

5.26.3.3 Wait()

```
void mage::Semaphore::Wait ( )
```

Decrements the value of this semaphore variable by one.

If the initial value of the semaphore is negative, the waiting queue is not empty and thus one blocked process can be transferred to the ready queue.

5.26.4 Member Data Documentation

5.26.4.1 m_handle

```
HANDLE mage::Semaphore::m_handle  [private]
```

The handle of this semaphore.

5.27 mage::Sound Class Reference

```
#include <sound.hpp>
```

Public Member Functions

- Sound (const string &filename)
- virtual ~Sound ()

5.27.1 Detailed Description

A class of sounds.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 Sound()

```
mage::Sound::Sound (
            const string & filename )
```

Constructs a sound.

Parameters

in	*filename*	A reference to the filename.

5.27.2.2 ~Sound()

```
virtual mage::Sound::~Sound ( )  [virtual]
```

Destructs a sound.

5.28 mage::Sphere Struct Reference

```
#include <geometry.hpp>
```

Public Member Functions

- Sphere ()
- Sphere (XMFLOAT3 p, float r)
- bool Encloses (const list< XMFLOAT4 > &planes)
- bool Collides (const Sphere &sphere, const XMFLOAT3 velocity_sum, float *collision_distance)

Public Attributes

- XMFLOAT3 p
- float r

5.28.1 Detailed Description

A struct of spheres.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 Sphere() [1/2]

```
mage::Sphere::Sphere ( )
```

Constructs a sphere.

5.28.2.2 Sphere() [2/2]

```
mage::Sphere::Sphere (
            XMFLOAT3 p,
            float r )
```

Constructs a sphere.

Parameters

in	*p*	The position
in	*r*	The radius.

5.28.3 Member Function Documentation

5.28.3.1 Collides()

```
bool mage::Sphere::Collides (
            const Sphere & sphere,
            const XMFLOAT3 velocity_sum,
            float * collision_distance )
```

Checks whether this sphere collides with a given sphere.

Parameters

in	*sphere*	The sphere.
in	*velocity_sum*	The sum of the velocities of both spheres.
out	*collision_distance*	The collision distance (in case of collision).

Returns

`true` if this sphere collides with *sphere*. `false` otherwise.

5.28.3.2 Encloses()

```
bool mage::Sphere::Encloses (
            const list< XMFLOAT4 > & planes )
```

Checks whether this sphere completely encloses the given (closed) volume.

Parameters

in	*planes*	A reference to a linked list containing the planes of the volume (each plane's coefficients are represented as a `XMFLOAT4`).

Returns

`true` if this sphere completely encloses *planes*. `false` otherwise.

5.28.4 Member Data Documentation

5.28.4.1 p

```
XMFLOAT3 mage::Sphere::p
```

The position of this sphere.

5.28.4.2 r

```
float mage::Sphere::r
```

The radius of this sphere.

5.29 mage::State Class Reference

`#include <state.hpp>`

Public Member Functions

- State (uint64_t id=0)
- virtual ~State ()
- virtual void Load ()
- virtual void Close ()
- virtual void RequestViewer (ViewerSetup *viewer_setup)
- virtual void Update (double elapsed_time)
- virtual void Render ()
- uint64_t GetId () const

Private Attributes

- const uint64_t m_id

5.29.1 Detailed Description

A class of states

5.29.2 Constructor & Destructor Documentation

5.29.2.1 State()

```
mage::State::State (
            uint64_t id = 0 )
```

Constructs a state with given id.

Parameters

in	*id*	The id.

5.29.2.2 ~State()

```
virtual mage::State::~State ( )  [virtual]
```

Destructs this state.

5.29.3 Member Function Documentation

5.29.3.1 Close()

```
virtual void mage::State::Close ( )  [virtual]
```

Closes this state. Allows this state to preform any post-processing destruction.

5.29.3.2 GetId()

```
uint64_t mage::State::GetId ( ) const
```

Returns the id of this state.

Returns

The id of this state.

5.29.3.3 Load()

```
virtual void mage::State::Load ( )  [virtual]
```

Loads this state. Allows this state to preform any pre-processing construction.

5.29.3.4 Render()

```
virtual void mage::State::Render ( )  [virtual]
```

Render this state.

5.29.3.5 RequestViewer()

```
virtual void mage::State::RequestViewer (
            ViewerSetup * viewer_setup )  [virtual]
```

Requests the view setup details for the given frame.

Parameters

in	*viewer_setup*	A pointer to a viewer setup.

5.29.3.6 Update()

```
virtual void mage::State::Update (
            double elapsed_time )  [virtual]
```

Updates this state.

Parameters

in	*elapsed_time*	The elapsed time since the previous update.

5.29.4 Member Data Documentation

5.29.4.1 m_id

```
const uint64_t mage::State::m_id  [private]
```

Application defined identifier (must be unique for state switching) of this state.

5.30 mage::StateManager Class Reference

```
#include <state_manager.hpp>
```

Public Member Functions

- StateManager ()
- virtual ~StateManager ()
- void AddState (State *state, bool change=true)
- void RemoveState (State *state)
- void ChangeState (uint64_t id)
- State * GetCurrentState () const
- bool IsStateChanged () const
- bool Update (double elapsed_time)

Private Member Functions

- void ChangeState (State *new_state)

Private Attributes

- list< State *> m_states
- State * m_current_state
- bool m_state_changed

5.30.1 Detailed Description

A class of state managers.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 StateManager()

```
mage::StateManager::StateManager ( )
```

Constructs a state manager.

5.30.2.2 ~StateManager()

```
virtual mage::StateManager::~StateManager ( )  [virtual]
```

Destructs this state manager.

5.30.3 Member Function Documentation

5.30.3.1 AddState()

```
void mage::StateManager::AddState (
            State * state,
            bool change = true )
```

Adds the given state from the states of this state manager.

Parameters

in	*state*	A pointer to the state.
in	*change*	Flag indicating whether the current state of this engine need to be changed to *state*.

5.30.3.2 ChangeState() [1/2]

```
void mage::StateManager::ChangeState (
            uint64_t id )
```

Changes the state of this state manager to the state with the given id.

Parameters

in	*id*	The id.

5.30.3.3 ChangeState() [2/2]

```
void mage::StateManager::ChangeState (
            State * new_state )  [private]
```

Changes the state of this state manager to the given state.

Parameters

in	*new_state*	A pointer to the new state.

5.30.3.4 GetCurrentState()

```
State* mage::StateManager::GetCurrentState ( ) const
```

Returns the current state of this state manager.

Returns

A pointer to the current state of this state manager.

5.30.3.5 IsStateChanged()

```
bool mage::StateManager::IsStateChanged ( ) const
```

Checks whether the state of this state manager is changed.

Returns

`true` if the state is changed. `false` otherwise.

5.30.3.6 RemoveState()

```
void mage::StateManager::RemoveState (
            State * state )
```

Removes the given state from the states of this state manager.

Parameters

in	*state*	A pointer to the state.

5.30.3.7 Update()

```
bool mage::StateManager::Update (
            double elapsed_time )
```

Updates this state manager and its current state.

Parameters

in	*elapsed_time*	The elapsed time since the previous update.

Returns

`true` if the state is changed in the current frame. `false` otherwise.

5.30.4 Member Data Documentation

5.30.4.1 m_current_state

```
State* mage::StateManager::m_current_state  [private]
```

A pointer to the current state of this state manager.

5.30.4.2 m_state_changed

```
bool mage::StateManager::m_state_changed  [private]
```

Flag indicating if the state changed in the current frame.

5.30.4.3 m_states

```
list< State * > mage::StateManager::m_states  [private]
```

The states of this state manager.

5.31 mage::Task Class Reference

```
#include <task.hpp>
```

Public Member Functions

- virtual ~Task ()
- virtual void Run ()=0

5.31.1 Detailed Description

A class of tasks.

5.31.2 Constructor & Destructor Documentation

5.31.2.1 ~Task()

```
virtual mage::Task::~Task ( )  [virtual]
```

Destructs this task.

5.31.3 Member Function Documentation

5.31.3.1 Run()

```
virtual void mage::Task::Run ( )  [pure virtual]
```

5.32 mage::Timer Class Reference

```
#include <timer.hpp>
```

Public Member Functions

- Timer ()
- virtual ~Timer ()
- void Start ()
- void Stop ()
- void Reset ()
- void Restart ()
- double Time ()

Private Member Functions

- double time ()

Private Attributes

- double m_time0
- double m_elapsed
- bool m_running
- LARGE_INTEGER m_performance_counter
- LARGE_INTEGER m_performance_frequency
- double m_performance_period

5.32.1 Detailed Description

A class of (high precision) timers.

5.32.2 Constructor & Destructor Documentation

5.32.2.1 Timer()

```
mage::Timer::Timer ( )
```

Constructs a timer.

5.32.2.2 ~Timer()

```
virtual mage::Timer::~Timer ( )  [virtual]
```

Destructs this timer.

5.32.3 Member Function Documentation

5.32.3.1 Reset()

```
void mage::Timer::Reset ( )
```

Resets this timer.

5.32.3.2 Restart()

```
void mage::Timer::Restart ( )
```

Restarts this timer.

5.32.3.3 Start()

```
void mage::Timer::Start ( )
```

Starts this timer.

5.32.3.4 Stop()

```
void mage::Timer::Stop ( )
```

Stops this timer.

5.32.3.5 Time()

```
double mage::Timer::Time ( )
```

Returns the elapsed time of this timer.

Returns

The elapsed time of this timer.

5.32.3.6 time()

```
double mage::Timer::time ( )  [private]
```

Returns the time of this timer.

Returns

The time of this timer.

Note

This member method encapsulates the performance of the underlying counter/frequency processing.

5.32.4 Member Data Documentation

5.32.4.1 m_elapsed

```
double mage::Timer::m_elapsed  [private]
```

The elapsed time of this timer.

5.32.4.2 m_performance_counter

```
LARGE_INTEGER mage::Timer::m_performance_counter  [private]
```

The counter of this timer.

5.32.4.3 m_performance_frequency

```
LARGE_INTEGER mage::Timer::m_performance_frequency  [private]
```

The frequency of this timer.

5.32.4.4 m_performance_period

```
double mage::Timer::m_performance_period  [private]
```

The period of this timer.

5.32.4.5 m_running

```
bool mage::Timer::m_running  [private]
```

Flag indicating whether this timer is running.

5.32.4.6 m_time0

```
double mage::Timer::m_time0  [private]
```

The initial time stamp of this timer.

5.33 mage::TLVertex Struct Reference

```
#include <geometry.hpp>
```

Public Member Functions

- TLVertex ()
- TLVertex (XMFLOAT4 p, XMFLOAT4 diffuse, float tu, float tv)

Public Attributes

- XMFLOAT4 p
- XMFLOAT4 diffuse
- float tu
- float tv

5.33.1 Detailed Description

A struct of transformed and lit vertices.

5.33.2 Constructor & Destructor Documentation

5.33.2.1 TLVertex() [1/2]

```
mage::TLVertex::TLVertex ( )
```

Constructs a transformed and lit vertex.

5.33.2.2 TLVertex() [2/2]

```
mage::TLVertex::TLVertex (
            XMFLOAT4 p,
            XMFLOAT4 diffuse,
            float tu,
            float tv )
```

Constructs a transformed and lit vertex.

Parameters

in	*p*	Position of the transformed and lit vertex (in screen space).
in	*diffuse*	Diffuse colour of the transformed and lit vertex.
in	*tu*	Texture u coordinate of the transformed and lit vertex.
in	*tv*	Texture v coordinate of the transformed and lit vertex.

5.33.3 Member Data Documentation

5.33.3.1 diffuse

```
XMFLOAT4 mage::TLVertex::diffuse
```

Diffuse colour of this transformed and lit vertex.

5.33.3.2 p

```
XMFLOAT4 mage::TLVertex::p
```

Position of this transformed and lit vertex (in screen space).

5.33.3.3 tu

```
float mage::TLVertex::tu
```

Texture u coordinate of this transformed and lit vertex.

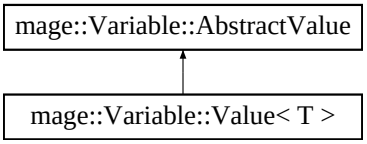
5.33.3.4 tv

```
float mage::TLVertex::tv
```

Texture v coordinate of this transformed and lit vertex.

5.34 mage::Variable::Value< T > Struct Template Reference

Inheritance diagram for mage::Variable::Value< T >:



Public Member Functions

- Value (const T *value)
- virtual ~Value ()
- virtual const void * GetValue () const

Private Attributes

- const T * m_value

5.34.1 Detailed Description

template<typename T>
struct mage::Variable::Value< T >

A struct of values.

Template Parameters

T	The type of the value.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 Value()

```
template<typename T >
mage::Variable::Value< T >::Value (
            const T * value )
```

Constructs a value.

Parameters

in	*value*	A pointer to the value.

5.34.2.2 ~Value()

```
template<typename T >
virtual mage::Variable::Value< T >::~Value ( )  [virtual]
```

Destructs this value.

5.34.3 Member Function Documentation

5.34.3.1 GetValue()

```
template<typename T >
virtual const void* mage::Variable::Value< T >::GetValue ( ) const  [virtual]
```

Returns the value of this value.

Returns

A pointer to the value of this value.

Implements mage::Variable::AbstractValue.

5.34.4 Member Data Documentation

5.34.4.1 m_value

```
template<typename T >
const T* mage::Variable::Value< T >::m_value  [private]
```

A pointer to the value of this value.

5.35 mage::Variable Struct Reference

```
#include <variable.hpp>
```

Classes

- struct AbstractValue
- struct Value

Public Member Functions

- template<typename T >
  Variable (const string &name, VariableType type, const T *value)
- ~Variable ()
- bool operator== (const Variable &variable) const
- bool operator!= (const Variable &variable) const
- const string & GetName () const
- const VariableType & GetType () const
- const void * GetValue () const

Private Attributes

- const string m_name
- const VariableType m_type
- const AbstractValue * m_value

5.35.1 Detailed Description

A struct of (immutable) variables.

5.35.2 Constructor & Destructor Documentation

5.35.2.1 Variable()

```
template<typename T >
mage::Variable::Variable (
            const string & name,
            VariableType type,
            const T * value )
```

Constructs a variable.

Template Parameters

T	The (storage) type of the value.

Parameters

in	*name*	The name.
in	*type*	The (scripting) type of the value.
in	*value*	A pointer to the value.

5.35.2.2 ~Variable()

```
mage::Variable::~Variable ( )
```

Destructs this variable.

5.35.3 Member Function Documentation

5.35.3.1 GetName()

```
const string& mage::Variable::GetName ( ) const
```

Returns the name of this variable.

Returns

A reference to the name of this variable.

5.35.3.2 GetType()

```
const VariableType& mage::Variable::GetType ( ) const
```

Returns the type of this value.

Returns

The type of this value.

5.35.3.3 GetValue()

```
const void* mage::Variable::GetValue ( ) const
```

Returns the value of this variable.

Returns

A pointer to the value of this variable.

5.35.3.4 operator"!=()

```
bool mage::Variable::operator!= (
            const Variable & variable ) const
```

Checks whether the given variable is not equal to this variable.

Parameters

in	*variable*	A reference to the variable to compare with.

Returns

`true` if and only if this variable and *variable* have not the same name. `false` otherwise.

5.35.3.5 operator==()

```
bool mage::Variable::operator== (
            const Variable & variable ) const
```

Checks whether the given variable is equal to this variable.

Parameters

in	*variable*	A reference to the variable to compare with.

Returns

`true` if and only if this variable and *variable* have the same name. `false` otherwise.

5.35.4 Member Data Documentation

5.35.4.1 m_name

```
const string mage::Variable::m_name  [private]
```

The name of this variable.

5.35.4.2 m_type

```
const VariableType mage::Variable::m_type  [private]
```

The type of this value.

Note

It is not possible to use typeid(T).name() since this assumes a bijection between the scripting types and the storage types, which is not the case. Thus the type needs to be stored explicitly.

5.35.4.3 m_value

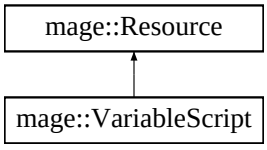
```
const AbstractValue* mage::Variable::m_value  [private]
```

A pointer to the value of this variable.

5.36 mage::VariableScript Class Reference

```
#include <variable_script.hpp>
```

Inheritance diagram for mage::VariableScript:



Public Member Functions

- VariableScript (const string &name, const string &path="./")
- virtual ~VariableScript ()
- void SaveScript (const string &filename="")
- void ImportVariable (const string &name, FILE *file)
- template<typename T >
  void AddVariable (const string &name, VariableType type, const T *value)
- void RemoveVariable (const string &name)
- template<typename T >
  const T * GetValueOfVariable (const string &name) const
- template<typename T >
  void SetValueOfVariable (const string &name, const T *value)

Private Attributes

- list< Variable *> m_variables

5.36.1 Detailed Description

A class of variable scripts.

5.36.2 Constructor & Destructor Documentation

5.36.2.1 VariableScript()

```
mage::VariableScript::VariableScript (
            const string & name,
            const string & path = "./" )
```

Constructs a variable script.

Parameters

in	*name*	A reference to the name of the variable script.
in	*path*	A reference to the path of the variable script.

5.36.2.2 ~VariableScript()

```
virtual mage::VariableScript::~VariableScript ( )  [virtual]
```

Destruct this variable script.

5.36.3 Member Function Documentation

5.36.3.1 AddVariable()

```
template<typename T >
void mage::VariableScript::AddVariable (
            const string & name,
            VariableType type,
            const T * value )
```

Adds the given variable to this variable script.

Precondition

No variable with the name *name* exists in this variable script.

Template Parameters

T	The type of the value.

Parameters

in	*name*	The name of the variable.
in	*type*	The type of the variable.
in	*value*	A pointer to the value of the variable.

5.36.3.2 GetValueOfVariable()

```
template<typename T >
const T* mage::VariableScript::GetValueOfVariable (
            const string & name ) const
```

Returns the value of the given variable in this variable script.

Template Parameters

T	The type of the value.

Parameters

in	*name*	The name of the variable.

Returns

NULL if no variable with the name *name* exists in this variable script.
A pointer to the value of the variable.

5.36.3.3 ImportVariable()

```
void mage::VariableScript::ImportVariable (
```

```
            const string & name,
            FILE * file )
```

Import the given variable from the given file to this variable script .

Precondition

No variable with the name *name* exists in this variable script.

Parameters

in	*name*	The name of the variable.
in,out	*file*	A pointer to a file containing the value of the variable.

5.36.3.4 RemoveVariable()

```
void mage::VariableScript::RemoveVariable (
            const string & name )
```

Removes the given variable from this variable script.

Parameters

in	*name*	The name of the variable.

5.36.3.5 SaveScript()

```
void mage::VariableScript::SaveScript (
            const string & filename = "" )
```

Saves this variable script with the given filename.

Parameters

in	*filename*	A reference to the filename.

5.36.3.6 SetValueOfVariable()

```
template<typename T >
void mage::VariableScript::SetValueOfVariable (
            const string & name,
            const T * value )
```

Sets the value of the given variable in this variable script.

Template Parameters

T	The type of the value.

Parameters

in	*name*	The name of the variable.
in	*value*	A pointer to the value of the variable.

Note

Nothing happens if no variable with the name *name* exists in this variable script.

5.36.4 Member Data Documentation

5.36.4.1 m_variables

```
list< Variable * > mage::VariableScript::m_variables  [private]
```

Linked list containing the variables in this variable script.

5.37 mage::Vertex Struct Reference

```
#include <geometry.hpp>
```

Public Member Functions

- Vertex ()
- Vertex (XMFLOAT3 p, XMFLOAT3 n, float tu, float tv)

Public Attributes

- XMFLOAT3 p
- XMFLOAT3 n
- float tu
- float tv

5.37.1 Detailed Description

A struct of vertices.

5.37.2 Constructor & Destructor Documentation

5.37.2.1 Vertex() [1/2]

```
mage::Vertex::Vertex ( )
```

Constructs a vertex.

5.37.2.2 Vertex() [2/2]

```
mage::Vertex::Vertex (
            XMFLOAT3 p,
            XMFLOAT3 n,
            float tu,
            float tv )
```

Constructs a vertex.

Parameters

in	*p*	Position of the vertex (in world space).
in	*n*	Normal of the vertex.
in	*tu*	Texture u coordinate of the vertex.
in	*tv*	Texture v coordinate of the vertex.

5.37.3 Member Data Documentation

5.37.3.1 n

```
XMFLOAT3 mage::Vertex::n
```

Normal of this vertex.

5.37.3.2 p

```
XMFLOAT3 mage::Vertex::p
```

Position of this vertex (in world space).

5.37.3.3 tu

```
float mage::Vertex::tu
```

Texture u coordinate of this vertex.

5.37.3.4 tv

```
float mage::Vertex::tv
```

Texture v coordinate of this vertex.

5.38 mage::ViewerSetup Struct Reference

```
#include <state.hpp>
```

Public Member Functions

- ViewerSetup ()

Public Attributes

- uint64_t m_view_clear_flags

5.38.1 Detailed Description

A struct of viewer setups.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 ViewerSetup()

```
mage::ViewerSetup::ViewerSetup ( )
```

Constructs a viewer setup.

5.38.3 Member Data Documentation

5.38.3.1 m_view_clear_flags

```
uint64_t mage::ViewerSetup::m_view_clear_flags
```

Flags used for clearing the view.