

MAGE

Generated by Doxygen 1.8.12

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Class Index	3
2.1	Class List	3
3	Namespace Documentation	5
3.1	Image Namespace Reference	5
3.1.1	Detailed Description	6
3.1.2	Enumeration Type Documentation	6
3.1.2.1	ReadWriteMutexLockType	6
3.1.3	Function Documentation	7
3.1.3.1	AllocAligned() [1/2]	7
3.1.3.2	AllocAligned() [2/2]	7
3.1.3.3	AtomicAdd() [1/2]	8
3.1.3.4	AtomicAdd() [2/2]	8
3.1.3.5	AtomicCompareAndSwap()	8
3.1.3.6	AtomicCompareAndSwapPointer()	9
3.1.3.7	AttachConsole()	9
3.1.3.8	EnqueueTasks()	9
3.1.3.9	Error()	10
3.1.3.10	FindWordEnd()	10
3.1.3.11	FreeAligned()	10
3.1.3.12	Info()	10

3.1.3.13	NumberOfSystemCores()	11
3.1.3.14	PrintConsoleHeader()	11
3.1.3.15	ProcessError()	11
3.1.3.16	Severe()	11
3.1.3.17	task_entry()	12
3.1.3.18	TasksCleanup()	12
3.1.3.19	TasksInit()	12
3.1.3.20	TerminalWidth()	12
3.1.3.21	WaitForAllTasks()	12
3.1.3.22	Warning()	12
3.1.3.23	WindowProc()	13
3.1.4	Variable Documentation	13
3.1.4.1	g_engine	13
3.1.4.2	general_configuration	13
3.1.4.3	lvertex_input_element_desc	13
3.1.4.4	nb_unfinished_tasks	14
3.1.4.5	task_queue	14
3.1.4.6	task_queue_mutex	14
3.1.4.7	tasks_running_condition	14
3.1.4.8	threads	14
3.1.4.9	tlvertex_input_element_desc	14
3.1.4.10	vertex_input_element_desc	15
3.1.4.11	worker_semaphore	15

4 Class Documentation	17
4.1 mage::AABB Struct Reference	17
4.1.1 Detailed Description	17
4.1.2 Constructor & Destructor Documentation	17
4.1.2.1 AABB() [1/2]	17
4.1.2.2 AABB() [2/2]	17
4.1.3 Member Function Documentation	18
4.1.3.1 EnclosedBy()	18
4.1.3.2 Encloses() [1/2]	18
4.1.3.3 Encloses() [2/2]	18
4.1.4 Member Data Documentation	19
4.1.4.1 p_max	19
4.1.4.2 p_min	19
4.2 mage::ConditionVariable Class Reference	19
4.2.1 Detailed Description	20
4.2.2 Member Enumeration Documentation	20
4.2.2.1 anonymous enum	20
4.2.3 Constructor & Destructor Documentation	20
4.2.3.1 ConditionVariable()	20
4.2.3.2 ~ConditionVariable()	20
4.2.4 Member Function Documentation	20
4.2.4.1 Lock()	20
4.2.4.2 Signal()	20
4.2.4.3 Unlock()	21
4.2.4.4 Wait()	21
4.2.5 Member Data Documentation	21
4.2.5.1 m_condition_mutex	21
4.2.5.2 m_events	21
4.2.5.3 m_nb_waiters	21
4.2.5.4 m_nb_waiters_mutex	21

4.3	mage::Edge Struct Reference	21
4.3.1	Detailed Description	22
4.3.2	Constructor & Destructor Documentation	22
4.3.2.1	Edge()	22
4.3.3	Member Data Documentation	22
4.3.3.1	v0	22
4.3.3.2	v1	22
4.4	mage::Engine Class Reference	22
4.4.1	Detailed Description	23
4.4.2	Constructor & Destructor Documentation	23
4.4.2.1	Engine()	23
4.4.2.2	~Engine()	23
4.4.3	Member Function Documentation	23
4.4.3.1	GetInput()	23
4.4.3.2	GetStateManager()	24
4.4.3.3	GetWindow()	24
4.4.3.4	Run()	24
4.4.3.5	SetDeactiveFlag()	24
4.4.4	Member Data Documentation	24
4.4.4.1	m_deactive	24
4.4.4.2	m_hwindow	25
4.4.4.3	m_input	25
4.4.4.4	m_loaded	25
4.4.4.5	m_setup	25
4.4.4.6	m_state_manager	25
4.5	mage::EngineSetup Struct Reference	25
4.5.1	Detailed Description	26
4.5.2	Constructor & Destructor Documentation	26
4.5.2.1	EngineSetup() [1/2]	26
4.5.2.2	EngineSetup() [2/2]	27

4.5.3	Member Data Documentation	27
4.5.3.1	m_hinstance	27
4.5.3.2	m_name	27
4.5.3.3	StateSetup	27
4.6	mage::Face Struct Reference	27
4.6.1	Detailed Description	28
4.6.2	Constructor & Destructor Documentation	28
4.6.2.1	Face()	28
4.6.3	Member Data Documentation	28
4.6.3.1	v0	28
4.6.3.2	v1	28
4.6.3.3	v2	29
4.7	mage::GeneralConfiguration Struct Reference	29
4.7.1	Detailed Description	29
4.7.2	Constructor & Destructor Documentation	29
4.7.2.1	GeneralConfiguration()	29
4.7.3	Member Function Documentation	29
4.7.3.1	IsQuiet()	29
4.7.3.2	IsVerbose()	30
4.7.4	Member Data Documentation	30
4.7.4.1	m_quiet	30
4.7.4.2	m_verbose	30
4.8	mage::IndexedEdge Struct Reference	30
4.8.1	Detailed Description	30
4.8.2	Member Data Documentation	30
4.8.2.1	iv0	30
4.8.2.2	iv1	31
4.9	mage::IndexedFace Struct Reference	31
4.9.1	Detailed Description	31
4.9.2	Member Data Documentation	31

4.9.2.1	iv0	31
4.9.2.2	iv1	31
4.9.2.3	iv2	31
4.10	mage::Input Class Reference	31
4.10.1	Detailed Description	32
4.10.2	Constructor & Destructor Documentation	32
4.10.2.1	Input()	32
4.10.2.2	~Input()	32
4.10.3	Member Function Documentation	33
4.10.3.1	GetDeltaWheel()	33
4.10.3.2	GetDeltaX()	33
4.10.3.3	GetDeltaY()	33
4.10.3.4	GetKeyPress()	33
4.10.3.5	GetMouseButtonPress()	34
4.10.3.6	GetPosX()	34
4.10.3.7	GetPosY()	34
4.10.3.8	Update()	34
4.10.4	Member Data Documentation	35
4.10.4.1	m_di	35
4.10.4.2	m_hwindow	35
4.10.4.3	m_key_press_stamp	35
4.10.4.4	m_key_state	35
4.10.4.5	m_keyboard	35
4.10.4.6	m_mouse	35
4.10.4.7	m_mouse_button_press_stamp	36
4.10.4.8	m_mouse_position	36
4.10.4.9	m_mouse_state	36
4.10.4.10	m_press_stamp	36
4.11	mage::LinkedList< T > Class Template Reference	36
4.11.1	Detailed Description	37

4.11.2	Constructor & Destructor Documentation	37
4.11.2.1	LinkedList()	37
4.11.2.2	~LinkedList()	37
4.11.3	Member Function Documentation	37
4.11.3.1	Add()	37
4.11.3.2	Empty()	38
4.11.3.3	GetAt()	38
4.11.3.4	GetCompleteLinkedListElement()	38
4.11.3.5	GetFirst()	39
4.11.3.6	GetIterator()	39
4.11.3.7	GetLast()	39
4.11.3.8	GetNext()	39
4.11.3.9	GetPrevious()	40
4.11.3.10	GetRandom()	40
4.11.3.11	GetSize()	40
4.11.3.12	InsertAfter()	41
4.11.3.13	InsertBefore()	41
4.11.3.14	Remove()	41
4.11.4	Member Data Documentation	42
4.11.4.1	m_first	42
4.11.4.2	m_last	42
4.11.4.3	m_size	42
4.12	mage::LinkedList< T >::LinkedListElement Struct Reference	42
4.12.1	Detailed Description	42
4.12.2	Constructor & Destructor Documentation	42
4.12.2.1	LinkedListElement()	42
4.12.2.2	~LinkedListElement()	43
4.12.3	Member Data Documentation	43
4.12.3.1	data	43
4.12.3.2	next	43

4.12.3.3	prev	43
4.13	mage::LinkedList< T >::LinkedListIterator Struct Reference	43
4.13.1	Detailed Description	44
4.13.2	Constructor & Destructor Documentation	44
4.13.2.1	LinkedListIterator()	44
4.13.2.2	~LinkedListIterator()	44
4.13.3	Member Function Documentation	44
4.13.3.1	HasNext()	44
4.13.3.2	Next()	45
4.13.4	Member Data Documentation	45
4.13.4.1	m_next	45
4.14	mage::LVertex Struct Reference	45
4.14.1	Detailed Description	45
4.14.2	Constructor & Destructor Documentation	45
4.14.2.1	LVertex() [1/2]	45
4.14.2.2	LVertex() [2/2]	45
4.14.3	Member Data Documentation	46
4.14.3.1	diffuse	46
4.14.3.2	p	46
4.14.3.3	tu	46
4.14.3.4	tv	46
4.15	mage::MemoryArena Class Reference	46
4.15.1	Detailed Description	47
4.15.2	Constructor & Destructor Documentation	47
4.15.2.1	MemoryArena()	47
4.15.2.2	~MemoryArena()	47
4.15.3	Member Function Documentation	47
4.15.3.1	Alloc() [1/2]	47
4.15.3.2	Alloc() [2/2]	48
4.15.3.3	FreeAll()	48

4.15.4	Member Data Documentation	48
4.15.4.1	m_available_blocks	48
4.15.4.2	m_block_size	48
4.15.4.3	m_current_block	49
4.15.4.4	m_current_block_pos	49
4.15.4.5	m_used_blocks	49
4.16	mage::Mutex Class Reference	49
4.16.1	Detailed Description	49
4.16.2	Constructor & Destructor Documentation	50
4.16.2.1	Mutex() [1/2]	50
4.16.2.2	Mutex() [2/2]	50
4.16.2.3	~Mutex()	50
4.16.3	Member Function Documentation	50
4.16.3.1	Create()	50
4.16.3.2	Destroy()	50
4.16.3.3	operator=()	51
4.16.4	Friends And Related Function Documentation	51
4.16.4.1	MutexLock	51
4.16.5	Member Data Documentation	51
4.16.5.1	m_critical_section	51
4.17	mage::MutexLock Struct Reference	51
4.17.1	Detailed Description	52
4.17.2	Constructor & Destructor Documentation	52
4.17.2.1	MutexLock() [1/2]	52
4.17.2.2	~MutexLock()	52
4.17.2.3	MutexLock() [2/2]	52
4.17.3	Member Function Documentation	52
4.17.3.1	operator=()	52
4.17.4	Member Data Documentation	53
4.17.4.1	m_mutex	53

4.18	mage::ProgressReporter Class Reference	53
4.18.1	Detailed Description	53
4.18.2	Constructor & Destructor Documentation	53
4.18.2.1	ProgressReporter()	53
4.18.2.2	~ProgressReporter()	54
4.18.3	Member Function Documentation	54
4.18.3.1	Done()	54
4.18.3.2	Update()	54
4.18.4	Member Data Documentation	54
4.18.4.1	m_buffer	54
4.18.4.2	m_current_pos	54
4.18.4.3	m_fout	55
4.18.4.4	m_mutex	55
4.18.4.5	m_nb_plusses_printed	55
4.18.4.6	m_nb_plusses_total	55
4.18.4.7	m_nb_work_done	55
4.18.4.8	m_nb_work_total	55
4.18.4.9	m_timer	55
4.19	mage::ReadWriteMutex Class Reference	55
4.19.1	Detailed Description	56
4.19.2	Constructor & Destructor Documentation	56
4.19.2.1	ReadWriteMutex() [1/2]	56
4.19.2.2	ReadWriteMutex() [2/2]	56
4.19.2.3	~ReadWriteMutex()	57
4.19.3	Member Function Documentation	57
4.19.3.1	AcquireRead()	57
4.19.3.2	AcquireWrite()	57
4.19.3.3	Create()	57
4.19.3.4	Destroy()	57
4.19.3.5	operator=()	57

4.19.3.6	ReleaseRead()	58
4.19.3.7	ReleaseWrite()	58
4.19.4	Friends And Related Function Documentation	58
4.19.4.1	ReadWriteMutexLock	58
4.19.5	Member Data Documentation	58
4.19.5.1	m_active_writer_readers	58
4.19.5.2	m_critical_section	58
4.19.5.3	m_nb_readers_waiting	59
4.19.5.4	m_nb_writers_waiting	59
4.19.5.5	m_ready_to_read_handle	59
4.19.5.6	m_ready_to_write_handle	59
4.20	mage::ReadWriteMutexLock Struct Reference	59
4.20.1	Detailed Description	60
4.20.2	Constructor & Destructor Documentation	60
4.20.2.1	ReadWriteMutexLock() [1/2]	60
4.20.2.2	~ReadWriteMutexLock()	60
4.20.2.3	ReadWriteMutexLock() [2/2]	60
4.20.3	Member Function Documentation	60
4.20.3.1	DowngradeToRead()	60
4.20.3.2	operator=()	61
4.20.3.3	UpgradeToWrite()	61
4.20.4	Member Data Documentation	61
4.20.4.1	m_mutex	61
4.20.4.2	m_type	61
4.21	mage::Reference< T > Class Template Reference	61
4.21.1	Detailed Description	62
4.21.2	Constructor & Destructor Documentation	62
4.21.2.1	Reference() [1/2]	62
4.21.2.2	Reference() [2/2]	62
4.21.2.3	~Reference()	62

4.21.3	Member Function Documentation	63
4.21.3.1	GetPtr()	63
4.21.3.2	operator bool()	63
4.21.3.3	operator->() [1/2]	63
4.21.3.4	operator->() [2/2]	63
4.21.3.5	operator=() [1/2]	63
4.21.3.6	operator=() [2/2]	64
4.21.4	Member Data Documentation	64
4.21.4.1	m_ptr	64
4.22	mage::ReferenceCounted Class Reference	64
4.22.1	Detailed Description	65
4.22.2	Constructor & Destructor Documentation	65
4.22.2.1	ReferenceCounted()	65
4.22.3	Member Function Documentation	65
4.22.3.1	DecrementReferenceCount()	65
4.22.3.2	IncrementReferenceCount()	65
4.22.4	Member Data Documentation	65
4.22.4.1	m_reference_count	65
4.23	mage::Resource Class Reference	66
4.23.1	Detailed Description	66
4.23.2	Constructor & Destructor Documentation	66
4.23.2.1	Resource()	66
4.23.2.2	~Resource()	67
4.23.3	Member Function Documentation	67
4.23.3.1	DecrementResourceReferenceCount()	67
4.23.3.2	GetFilename()	67
4.23.3.3	GetName()	67
4.23.3.4	GetPath()	67
4.23.3.5	IncrementResourceReferenceCount()	68
4.23.4	Friends And Related Function Documentation	68

4.23.4.1	ResourceManager	68
4.23.5	Member Data Documentation	68
4.23.5.1	m_name	68
4.23.5.2	m_path	68
4.23.5.3	m_resource_reference_count	68
4.24	mage::ResourceManager< T > Class Template Reference	68
4.24.1	Detailed Description	69
4.24.2	Constructor & Destructor Documentation	69
4.24.2.1	ResourceManager()	69
4.24.2.2	~ResourceManager()	69
4.24.3	Member Function Documentation	70
4.24.3.1	Add()	70
4.24.3.2	EmptyDestroy()	70
4.24.3.3	GetResource()	70
4.24.3.4	GetResources()	71
4.24.3.5	Remove()	71
4.24.4	Member Data Documentation	71
4.24.4.1	CreateResource	71
4.24.4.2	m_resources	71
4.25	mage::Semaphore Class Reference	71
4.25.1	Detailed Description	72
4.25.2	Constructor & Destructor Documentation	72
4.25.2.1	Semaphore()	72
4.25.2.2	~Semaphore()	72
4.25.3	Member Function Documentation	72
4.25.3.1	Post()	72
4.25.3.2	TryWait()	72
4.25.3.3	Wait()	73
4.25.4	Member Data Documentation	73
4.25.4.1	m_handle	73

4.26	mage::Sphere Struct Reference	73
4.26.1	Detailed Description	73
4.26.2	Constructor & Destructor Documentation	73
4.26.2.1	Sphere() [1/2]	73
4.26.2.2	Sphere() [2/2]	73
4.26.3	Member Function Documentation	74
4.26.3.1	Collides()	74
4.26.3.2	Encloses()	74
4.26.4	Member Data Documentation	74
4.26.4.1	p	74
4.26.4.2	r	75
4.27	mage::State Class Reference	75
4.27.1	Detailed Description	75
4.27.2	Constructor & Destructor Documentation	75
4.27.2.1	State()	75
4.27.2.2	~State()	76
4.27.3	Member Function Documentation	76
4.27.3.1	Close()	76
4.27.3.2	GetId()	76
4.27.3.3	Load()	76
4.27.3.4	Render()	76
4.27.3.5	RequestViewer()	76
4.27.3.6	Update()	77
4.27.4	Member Data Documentation	77
4.27.4.1	m_id	77
4.28	mage::StateManager Class Reference	77
4.28.1	Detailed Description	78
4.28.2	Constructor & Destructor Documentation	78
4.28.2.1	StateManager()	78
4.28.2.2	~StateManager()	78

4.28.3	Member Function Documentation	78
4.28.3.1	AddState()	78
4.28.3.2	ChangeState() [1/2]	78
4.28.3.3	ChangeState() [2/2]	79
4.28.3.4	GetCurrentState()	79
4.28.3.5	IsStateChanged()	79
4.28.3.6	RemoveState()	79
4.28.3.7	Update()	79
4.28.4	Member Data Documentation	80
4.28.4.1	m_current_state	80
4.28.4.2	m_state_changed	80
4.28.4.3	m_states	80
4.29	mage::Task Class Reference	80
4.29.1	Detailed Description	80
4.29.2	Constructor & Destructor Documentation	81
4.29.2.1	~Task()	81
4.29.3	Member Function Documentation	81
4.29.3.1	Run()	81
4.30	mage::Timer Class Reference	81
4.30.1	Detailed Description	81
4.30.2	Constructor & Destructor Documentation	82
4.30.2.1	Timer()	82
4.30.2.2	~Timer()	82
4.30.3	Member Function Documentation	82
4.30.3.1	Reset()	82
4.30.3.2	Restart()	82
4.30.3.3	Start()	82
4.30.3.4	Stop()	82
4.30.3.5	Time()	82
4.30.3.6	time()	83

4.30.4	Member Data Documentation	83
4.30.4.1	m_elapsed	83
4.30.4.2	m_performance_counter	83
4.30.4.3	m_performance_frequency	83
4.30.4.4	m_performance_period	83
4.30.4.5	m_running	83
4.30.4.6	m_time0	84
4.31	mage::TLVertex Struct Reference	84
4.31.1	Detailed Description	84
4.31.2	Constructor & Destructor Documentation	84
4.31.2.1	TLVertex() [1/2]	84
4.31.2.2	TLVertex() [2/2]	84
4.31.3	Member Data Documentation	85
4.31.3.1	diffuse	85
4.31.3.2	p	85
4.31.3.3	tu	85
4.31.3.4	tv	85
4.32	mage::Vertex Struct Reference	85
4.32.1	Detailed Description	86
4.32.2	Constructor & Destructor Documentation	86
4.32.2.1	Vertex() [1/2]	86
4.32.2.2	Vertex() [2/2]	86
4.32.3	Member Data Documentation	86
4.32.3.1	n	86
4.32.3.2	p	86
4.32.3.3	tu	87
4.32.3.4	tv	87
4.33	mage::ViewerSetup Struct Reference	87
4.33.1	Detailed Description	87
4.33.2	Constructor & Destructor Documentation	87
4.33.2.1	ViewerSetup()	87
4.33.3	Member Data Documentation	87
4.33.3.1	m_view_clear_flags	87

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

mage	5
--------------------------------	---

Chapter 2

Class Index

2.1 Class List

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

mage::AABB	17
mage::ConditionVariable	19
mage::Edge	21
mage::Engine	22
mage::EngineSetup	25
mage::Face	27
mage::GeneralConfiguration	29
mage::IndexedEdge	30
mage::IndexedFace	31
mage::Input	31
mage::LinkedList< T >	36
mage::LinkedList< T >::LinkedListElement	42
mage::LinkedList< T >::LinkedListIterator	43
mage::LVertex	45
mage::MemoryArena	46
mage::Mutex	49
mage::MutexLock	51
mage::ProgressReporter	53
mage::ReadWriteMutex	55
mage::ReadWriteMutexLock	59
mage::Reference< T >	61
mage::ReferenceCounted	64
mage::Resource	66
mage::ResourceManager< T >	68
mage::Semaphore	71
mage::Sphere	73
mage::State	75
mage::StateManager	77
mage::Task	80
mage::Timer	81
mage::TLVertex	84
mage::Vertex	85
mage::ViewerSetup	87

Chapter 3

Namespace Documentation

3.1 mage Namespace Reference

Classes

- struct [AABB](#)
- class [ConditionVariable](#)
- struct [Edge](#)
- class [Engine](#)
- struct [EngineSetup](#)
- struct [Face](#)
- struct [GeneralConfiguration](#)
- struct [IndexedEdge](#)
- struct [IndexedFace](#)
- class [Input](#)
- class [LinkedList](#)
- struct [LVertex](#)
- class [MemoryArena](#)
- class [Mutex](#)
- struct [MutexLock](#)
- class [ProgressReporter](#)
- class [ReadWriteMutex](#)
- struct [ReadWriteMutexLock](#)
- class [Reference](#)
- class [ReferenceCounted](#)
- class [Resource](#)
- class [ResourceManager](#)
- class [Semaphore](#)
- struct [Sphere](#)
- class [State](#)
- class [StateManager](#)
- class [Task](#)
- class [Timer](#)
- struct [TLVertex](#)
- struct [Vertex](#)
- struct [ViewerSetup](#)

Enumerations

- enum [ReadWriteMutexLockType](#) { [READ](#), [WRITE](#) }

Functions

- LRESULT CALLBACK [WindowProc](#) (HWND hwnd, UINT msg, WPARAM wparam, LPARAM lparam)
- static bool [AttachConsole](#) ()
- static void [PrintConsoleHeader](#) ()
- 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)
- int [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
- 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](#)

3.1.1 Detailed Description

The namespace for all the MAGE functionality.

3.1.2 Enumeration Type Documentation

3.1.2.1 ReadWriteMutexLockType

```
enum mage::ReadWriteMutexLockType
```

Type of read write mutex locks.

Enumerator

READ	
WRITE	

3.1.3 Function Documentation

3.1.3.1 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.

3.1.3.2 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.

3.1.3.3 AtomicAdd() [1/2]

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

Performs an atomic addition operation on the specified values.

Parameters

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

Returns

The function returns the result of the operation.

3.1.3.4 AtomicAdd() [2/2]

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

Performs an atomic addition operation on the specified values.

Parameters

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

Returns

The function returns the result of the operation.

3.1.3.5 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	<i>destination</i>	
in	<i>exchange</i>	The exchange value.
in	<i>comparand</i>	The value to compare to <i>destination</i> .

Returns

The function returns the initial value of *destination*.

3.1.3.6 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	<i>destination</i>	
in	<i>exchange</i>	The exchange pointer.
in	<i>comparand</i>	The pointer to compare to <i>destination</i> .

Returns

The function returns the initial pointer of *destination*.

3.1.3.7 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.

3.1.3.8 EnqueueTasks()

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

Enqueues the given tasks.

Parameters

in	<i>tasks</i>	The tasks.
----	--------------	------------

3.1.3.9 Error()

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

Notifies an error message.

Parameters

in	<i>format</i>	Pointer to the message format.
----	---------------	--------------------------------

3.1.3.10 FindWordEnd()

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

Finds the end of a word.

Parameters

in	<i>buffer</i>	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.

3.1.3.11 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	<i>ptr</i>	A pointer to the memory block that was allocated.
----	------------	---

3.1.3.12 Info()

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

Notifies an info message.

Parameters

in	<i>format</i>	Pointer to the message format.
----	---------------	--------------------------------

3.1.3.13 NumberOfSystemCores()

```
int mage::NumberOfSystemCores ( )
```

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

Returns

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

3.1.3.14 PrintConsoleHeader()

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

Prints the header of the engine to the console.

3.1.3.15 ProcessError()

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

Process the given error.

Parameters

in	<i>format</i>	The format of the error string.
in	<i>args</i>	The arguments of the format string.
in	<i>error_type</i>	The type of the error.
in	<i>error_disposition</i>	Disposition of the error.

3.1.3.16 Severe()

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

Notifies a severe message.

Parameters

in	<i>format</i>	Pointer to the message format.
----	---------------	--------------------------------

3.1.3.17 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	<i>lpParameter</i>	The thread data passed to the function using the <code>lpParameter</code> parameter of <code>CreateThread</code> .
----	--------------------	--

Returns

A value indicating success or failure.

3.1.3.18 TasksCleanup()

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

Clean the tasks.

3.1.3.19 TasksInit()

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

Initialize the tasks.

3.1.3.20 TerminalWidth()

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

Returns the fixed terminal width.

Returns

The fixed terminal width.

3.1.3.21 WaitForAllTasks()

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

Waits for all the tasks to finish.

3.1.3.22 Warning()

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

Notifies a warning message.

Parameters

in	<i>format</i>	Pointer to the message format.
----	---------------	--------------------------------

3.1.3.23 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	<i>hwnd</i>	A handle to the window.
in	<i>msg</i>	The message.
in	<i>wparam</i>	Additional message information. The contents of this parameter depend on the value of <i>msg</i> .
in	<i>lparam</i>	Additional message information. The contents of this parameter depend on the value of <i>msg</i> .

Returns

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

3.1.4 Variable Documentation

3.1.4.1 g_engine

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

The engine used by the user.

3.1.4.2 general_configuration

```
GeneralConfiguration mage::general_configuration
```

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

3.1.4.3 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](#).

3.1.4.4 nb_unfinished_tasks

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

The number of unfinished tasks.

3.1.4.5 task_queue

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

The task queue.

3.1.4.6 task_queue_mutex

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

The mutex for exclusive access to the task queue.

3.1.4.7 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.

3.1.4.8 threads

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

The thread handles.

3.1.4.9 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](#)

3.1.4.10 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](#).

3.1.4.11 worker_semaphore

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

The worker semaphore for being able to work.

Chapter 4

Class Documentation

4.1 `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 [LinkedList](#)< XMFLOAT4 > &planes) const

Public Attributes

- XMFLOAT3 [p_min](#)
- XMFLOAT3 [p_max](#)

4.1.1 Detailed Description

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

4.1.2 Constructor & Destructor Documentation

4.1.2.1 `AABB()` [1/2]

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

Constructs an [AABB](#).

4.1.2.2 `AABB()` [2/2]

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

Constructs an [AABB](#).

Parameters

in	<i>p_min</i>	The minimum extents.
in	<i>p_max</i>	The maximum extents.

4.1.3 Member Function Documentation

4.1.3.1 EnclosedBy()

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

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

Parameters

in	<i>planes</i>	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.

4.1.3.2 Encloses() [1/2]

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

Checks whether this [AABB](#) completely encloses the given [AABB](#).

Parameters

in	<i>aabb</i>	A reference to the AABB .
----	-------------	---

Returns

`true` if this [AABB](#) completely encloses *aabb*. `false` otherwise.

4.1.3.3 Encloses() [2/2]

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

Checks whether this [AABB](#) completely encloses the given face.

Parameters

<code>in</code>	<code>face</code>	A reference to the face.
-----------------	-------------------	--------------------------

Returns

`true` if this [AABB](#) completely encloses `face`. `false` otherwise.

4.1.4 Member Data Documentation

4.1.4.1 `p_max`

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

The maximum extents of this [AABB](#).

4.1.4.2 `p_min`

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

The minimum extents of this [AABB](#).

4.2 `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](#)]

4.2.1 Detailed Description

A class of condition variables.

4.2.2 Member Enumeration Documentation

4.2.2.1 anonymous enum

```
anonymous enum [private]
```

Type of events (indices).

Enumerator

SIGNAL	
BROADCAST	
NUM_EVENTS	

4.2.3 Constructor & Destructor Documentation

4.2.3.1 ConditionVariable()

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

Constructs a condition variable.

4.2.3.2 ~ConditionVariable()

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

Destructs this condition variable.

4.2.4 Member Function Documentation

4.2.4.1 Lock()

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

Locks this condition variable.

4.2.4.2 Signal()

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

Signal a condition change.

4.2.4.3 Unlock()

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

Unlocks this condition variable.

4.2.4.4 Wait()

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

Wait for a signal indicating a condition change.

4.2.5 Member Data Documentation

4.2.5.1 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.

4.2.5.2 m_events

```
HANDLE mage::ConditionVariable::m_events[NUM\_EVENTS] [private]
```

Signal and broadcast event handles of this condition variable.

4.2.5.3 m_nb_waiters

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

The number of waiters of this condition variable.

4.2.5.4 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.

4.3 mage::Edge Struct Reference

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

Public Member Functions

- [Edge](#) ([Vertex](#) *v0, [Vertex](#) *v1)

Public Attributes

- [Vertex](#) * v0
- [Vertex](#) * v1

4.3.1 Detailed Description

A struct of edges.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 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.

4.3.3 Member Data Documentation

4.3.3.1 v0

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

The first vertex of this edge.

4.3.3.2 v1

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

The second vertex of this edge.

4.4 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)
- [StateManager](#) * [GetStateManager](#) () const
- [Input](#) * [GetInput](#) () const

Private Attributes

- [EngineSetup](#) * [m_setup](#)
- bool [m_loaded](#)
- HWND [m_hwindow](#)
- bool [m_deactive](#)
- [StateManager](#) * [m_state_manager](#)
- [Input](#) * [m_input](#)

4.4.1 Detailed Description

A class of engines.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Engine()

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

Constructs an engine from the given engine setup.

Parameters

in	<i>setup</i>	A pointer to an engine setup.
----	--------------	-------------------------------

4.4.2.2 ~Engine()

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

Destructs this engine.

4.4.3 Member Function Documentation

4.4.3.1 GetInput()

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

Returns the input object of this engine.

Returns

A pointer to the input object of this engine

4.4.3.2 GetStateManager()

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

Returns the state manager of this engine.

Returns

A pointer to the state manager of this engine

4.4.3.3 GetWindow()

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

Returns a handle to the window of this engine.

4.4.3.4 Run()

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

Runs the engine setup.

4.4.3.5 SetDeactiveFlag()

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

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

Parameters

in	<i>deactive</i>	The new value for the deactive flag.
----	-----------------	--------------------------------------

4.4.4 Member Data Documentation**4.4.4.1 m_deactive**

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

Flag indicating whether the application is active or not.

4.4.4.2 m_hwindow

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

Main window handle of this engine.

4.4.4.3 m_input

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

A pointer to the input object of this engine.

4.4.4.4 m_loaded

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

Flag indicating whether this engine is loaded.

4.4.4.5 m_setup

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

Pointer to a copy of the engine setup structure.

4.4.4.6 m_state_manager

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

A pointer to the state manager of this engine.

4.5 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](#))()

4.5.1 Detailed Description

A struct of engine setups.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 EngineSetup() [1/2]

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

Constructs an engine setup with the given application name.

Parameters

in	<i>name</i>	A reference to the name of the application.
----	-------------	---

4.5.2.2 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	<i>setup</i>	A pointer to the engine setup.
----	--------------	--------------------------------

4.5.3 Member Data Documentation**4.5.3.1 m_hinstance**

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

Application instance handle.

4.5.3.2 m_name

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

Name of the application.

4.5.3.3 StateSetup

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

The state setup function.

4.6 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

4.6.1 Detailed Description

A struct of faces.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 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.

4.6.3 Member Data Documentation

4.6.3.1 v0

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

The first vertex of this face.

4.6.3.2 v1

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

The second vertex of this face.

4.6.3.3 v2

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

The third vertex of this face.

4.7 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](#)

4.7.1 Detailed Description

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

4.7.2 Constructor & Destructor Documentation

4.7.2.1 GeneralConfiguration()

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

Constructs a new general configuration.

4.7.3 Member Function Documentation

4.7.3.1 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.

4.7.3.2 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.

4.7.4 Member Data Documentation

4.7.4.1 m_quiet

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

Flag indicating the logging of the engine processing is quiet.

4.7.4.2 m_verbose

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

Flag indicating the logging of the engine processing is verbose.

4.8 mage::IndexedEdge Struct Reference

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

Public Attributes

- `uint16_t iv0`
- `uint16_t iv1`

4.8.1 Detailed Description

A struct of indexed edges.

4.8.2 Member Data Documentation

4.8.2.1 iv0

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

The index of the edge's first vertex.

4.8.2.2 iv1

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

The index of the edge's second vertex.

4.9 mage::IndexedFace Struct Reference

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

Public Attributes

- [uint16_t iv0](#)
- [uint16_t iv1](#)
- [uint16_t iv2](#)

4.9.1 Detailed Description

A struct of indexed faces.

4.9.2 Member Data Documentation

4.9.2.1 iv0

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

Index of the face's first vertex.

4.9.2.2 iv1

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

Index of the face's second vertex.

4.9.2.3 iv2

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

Index of the face's third vertex.

4.10 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](#)

4.10.1 Detailed Description

A class of input objects.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Input()

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

Constructs an input for the given window handle.

Parameters

in	<i>hwindow</i>	The handle of the parent window.
----	----------------	----------------------------------

4.10.2.2 ~Input()

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

Destructs this input object.

4.10.3 Member Function Documentation

4.10.3.1 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.

4.10.3.2 GetDeltaX()

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

Returns the change in the mouse's horizontal coordinate.

Returns

The change in the mouse's horizontal coordinate.

4.10.3.3 GetDeltaY()

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

Returns the change in the mouse's vertical coordinate.

Returns

The change in the mouse's vertical coordinate.

4.10.3.4 GetKeyPress()

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

Checks whether the given key is pressed.

Parameters

in	<i>key</i>	The key.
in	<i>ignore_press_stamp</i>	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.

4.10.3.5 GetMouseButtonPress()

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

Checks whether the given mouse button is pressed.

Parameters

in	<i>mouse_button</i>	The mouse button.
in	<i>ignore_press_stamp</i>	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.

4.10.3.6 GetPosX()

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

Returns the horizontal position of the mouse.

Returns

The horizontal position of the mouse.

4.10.3.7 GetPosY()

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

Returns the vertical position of the mouse.

Returns

The vertical position of the mouse.

4.10.3.8 Update()

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

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

4.10.4 Member Data Documentation

4.10.4.1 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.

4.10.4.2 m_hwindow

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

The handle of the parent window.

4.10.4.3 m_key_press_stamp

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

Stamps the keys pressed in the last frame.

4.10.4.4 m_key_state

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

[State](#) of the keys.

4.10.4.5 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.

4.10.4.6 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.

4.10.4.7 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.

4.10.4.8 m_mouse_position

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

The position of the mouse cursor on the screen.

4.10.4.9 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.

4.10.4.10 m_press_stamp

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

The current press stamp (incremented every frame).

4.11 mage::LinkedList< T > Class Template Reference

```
#include <linkedlist.hpp>
```

Classes

- struct [LinkedListElement](#)
- struct [LinkedListIterator](#)

Public Member Functions

- [LinkedList](#) ()
- virtual [~LinkedList](#) ()
- T * [Add](#) (T *data)
- T * [InsertBefore](#) (T *data, [LinkedListElement](#) *next_element)
- T * [InsertAfter](#) (T *data, [LinkedListElement](#) *prev_element)
- void [Remove](#) (T **data, bool data_destruction=true)
- void [Empty](#) (bool data_destruction=true)
- T * [GetFirst](#) () const
- T * [GetLast](#) () const
- T * [GetPrevious](#) (T *data) const
- T * [GetNext](#) (T *data) const
- T * [GetAt](#) (uint64_t index) const
- T * [GetRandom](#) () const
- [LinkedListIterator](#) [GetIterator](#) () const
- [LinkedListElement](#) * [GetCompleteLinkedListElement](#) (T *data) const
- uint64_t [GetSize](#) () const

Private Attributes

- [LinkedListElement](#) * `m_first`
- [LinkedListElement](#) * `m_last`
- `uint64_t` `m_size`

4.11.1 Detailed Description

```
template<typename T>
class mage::LinkedList< T >
```

A class of (doubly) linked lists.

Template Parameters

<code>T</code>	The type of data stored in the linked list.
----------------	---

4.11.2 Constructor & Destructor Documentation

4.11.2.1 LinkedList()

```
template<typename T>
mage::LinkedList< T >::LinkedList ( )
```

Constructs an empty linked list.

4.11.2.2 ~LinkedList()

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

Destructs this linked list.

Note

The data associated with the elements in this linked list will be destructed as well.

4.11.3 Member Function Documentation

4.11.3.1 Add()

```
template<typename T>
T* mage::LinkedList< T >::Add (
    T * data )
```

Adds the given data to the end of this linked list.

Parameters

in	<i>data</i>	A pointer to the data.
----	-------------	------------------------

Returns

A pointer to the data.

4.11.3.2 Empty()

```
template<typename T>
void mage::LinkedList< T >::Empty (
    bool data_destruction = true )
```

Destroys all the elements in this linked list.

Parameters

in	<i>data_destruction</i>	if <code>true</code> the data associated with the elements in this linkedlist will be destructed. if <code>false</code> the data associated with the elements in this linkedlist will not be destructed.
----	-------------------------	--

4.11.3.3 GetAt()

```
template<typename T>
T* mage::LinkedList< T >::GetAt (
    uint64_t index ) const
```

Returns a pointer to the data of the element in this linked list at the given index.

Parameters

in	<i>index</i>	The index of the element.
----	--------------	---------------------------

Returns

NULL if the index is out of bounds.

A pointer to the data of the element in this linked list at index *index*.

4.11.3.4 GetCompleteLinkedListElement()

```
template<typename T>
LinkedListElement* mage::LinkedList< T >::GetCompleteLinkedListElement (
    T * data ) const
```

Returns the (complete) element in this linked list associated with the given data.

Parameters

in	<i>data</i>	A pointer to the data.
----	-------------	------------------------

Returns

NULL if no element in this linkedlist is associated with the given data.
 The (complete) element in this linked list associated with *data*.

4.11.3.5 GetFirst()

```
template<typename T>
T* mage::LinkedList< T >::GetFirst ( ) const
```

Returns a pointer to the data of the first element in this linked list.

Returns

NULL if this linked list contains no elements.
 A pointer to the data of the first element in this linked list.

4.11.3.6 GetIterator()

```
template<typename T>
LinkedListIterator mage::LinkedList< T >::GetIterator ( ) const
```

Returns a forward iterator for this linked list.

Returns

An iterator for this linked list.

4.11.3.7 GetLast()

```
template<typename T>
T* mage::LinkedList< T >::GetLast ( ) const
```

Returns a pointer to the data of the last element in this linked list.

Returns

NULL if this linked list contains no elements.
 A pointer to the data of the last element in this linked list.

4.11.3.8 GetNext()

```
template<typename T>
T* mage::LinkedList< T >::GetNext (
    T * data ) const
```

Returns a pointer to the data of the next element in this linked list from the element corresponding to the given data.

Parameters

in	<i>data</i>	A pointer to the data.
----	-------------	------------------------

Returns

NULL if *data* is associated with the first element in this linked list.

A pointer to the data of the next element in this linked list from the element corresponding to *data*.

4.11.3.9 GetPrevious()

```
template<typename T>
T* mage::LinkedList< T >::GetPrevious (
    T * data ) const
```

Returns a pointer to the data of the previous element in this linked list from the element corresponding to the given data.

Parameters

in	<i>data</i>	A pointer to the data.
----	-------------	------------------------

Returns

NULL if *data* is associated with the last element in this linked list.

A pointer to the data of the previous element in this linked list from the element corresponding to *data*.

4.11.3.10 GetRandom()

```
template<typename T>
T* mage::LinkedList< T >::GetRandom ( ) const
```

Returns a pointer to the data of a random element in this linked list.

Returns

A pointer to the data of a random element in this linked list.

4.11.3.11 GetSize()

```
template<typename T>
uint64_t mage::LinkedList< T >::GetSize ( ) const
```

Returns the size of this linked list.

Returns

The size of this linked list.

4.11.3.12 InsertAfter()

```
template<typename T>
T* mage::LinkedList< T >::InsertAfter (
    T * data,
    LinkedListElement * prev_element )
```

Inserts the given data into this linked list just after the given element in this linked list.

Parameters

in	<i>data</i>	A pointer to the data.
in	<i>prev_element</i>	A pointer to the previous element in this linked list.

Returns

A pointer to the data.

4.11.3.13 InsertBefore()

```
template<typename T>
T* mage::LinkedList< T >::InsertBefore (
    T * data,
    LinkedListElement * next_element )
```

Inserts the given data into this linked list just before the given element in this linked list.

Parameters

in	<i>data</i>	A pointer to the data.
in	<i>next_element</i>	A pointer to the next element in this linked list.

Returns

A pointer to the data.

4.11.3.14 Remove()

```
template<typename T>
void mage::LinkedList< T >::Remove (
    T ** data,
    bool data_destruction = true )
```

Removes the given data from this linked list.

Parameters

in, out	<i>data</i>	A pointer to a pointer to the data which will point to <code>NULL</code> after removal.
in	<i>data_destruction</i>	if <code>true</code> the data will be destructed. if <code>false</code> the data will not be destructed.

4.11.4 Member Data Documentation

4.11.4.1 m_first

```
template<typename T>
LinkedListElement* mage::LinkedList< T >::m_first [private]
```

Pointer to first element in this linked list.

4.11.4.2 m_last

```
template<typename T>
LinkedListElement* mage::LinkedList< T >::m_last [private]
```

Pointer to last element in this linked list.

4.11.4.3 m_size

```
template<typename T>
uint64_t mage::LinkedList< T >::m_size [private]
```

Total number of elements in this linked list.

4.12 mage::LinkedList< T >::LinkedListElement Struct Reference

```
#include <linkedlist.hpp>
```

Public Member Functions

- [LinkedListElement](#) (T *data)
- virtual [~LinkedListElement](#) ()

Public Attributes

- T * [data](#)
- [LinkedListElement](#) * [next](#)
- [LinkedListElement](#) * [prev](#)

4.12.1 Detailed Description

```
template<typename T>
struct mage::LinkedList< T >::LinkedListElement
```

A struct of elements of a [mage::LinkedList<T>](#).

4.12.2 Constructor & Destructor Documentation

4.12.2.1 LinkedListElement()

```
template<typename T>
mage::LinkedList< T >::LinkedListElement::LinkedListElement (
    T * data )
```

Constructs a linked list element associated with the given data.

Parameters

in	data	The data to associate with.
----	------	-----------------------------

4.12.2.2 ~LinkedListElement()

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

Destructs this linked list element.

4.12.3 Member Data Documentation

4.12.3.1 data

```
template<typename T>
T* mage::LinkedList< T >::LinkedListElement::data
```

Pointer to the data held in this element.

4.12.3.2 next

```
template<typename T>
LinkedListElement* mage::LinkedList< T >::LinkedListElement::next
```

Pointer to the next element in the linked list.

4.12.3.3 prev

```
template<typename T>
LinkedListElement* mage::LinkedList< T >::LinkedListElement::prev
```

Pointer to the previous element in the linked list.

4.13 mage::LinkedList< T >::LinkedListIterator Struct Reference

```
#include <linkedlist.hpp>
```

Public Member Functions

- [LinkedListIterator](#) ([LinkedListElement](#) *first)
- virtual [~LinkedListIterator](#) ()
- bool [HasNext](#) () const
- T * [Next](#) ()

Private Attributes

- [LinkedListElement](#) * `m_next`

4.13.1 Detailed Description

```
template<typename T>
struct mage::LinkedList< T >::LinkedListIterator
```

A struct of forward iterators for a [mage::LinkedList<T>](#).

4.13.2 Constructor & Destructor Documentation

4.13.2.1 LinkedListIterator()

```
template<typename T>
mage::LinkedList< T >::LinkedListIterator::LinkedListIterator (
    LinkedListElement * first )
```

Constructs a linked list iterator starting from the given first element of a linked list.

Parameters

in	<i>first</i>	A pointer to the first element of a linked list.
----	--------------	--

4.13.2.2 ~LinkedListIterator()

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

Destructs this linked list iterator.

4.13.3 Member Function Documentation

4.13.3.1 HasNext()

```
template<typename T>
bool mage::LinkedList< T >::LinkedListIterator::HasNext ( ) const
```

Checks whether there is a next element in the linked list of this linked list iterator.

Returns

`true` if there is a next element in the linked list of this linked list iterator. `false` otherwise.

4.13.3.2 Next()

```
template<typename T>
T* mage::LinkedList< T >::LinkedListIterator::Next ( )
```

Returns a pointer to the data of the next element in the linked list of this linked list iterator.

Returns

A pointer to the data of the next element in the linked list of this linked list iterator.

4.13.4 Member Data Documentation

4.13.4.1 m_next

```
template<typename T>
LinkedListElement* mage::LinkedList< T >::LinkedListIterator::m_next [private]
```

Pointer to the next element in the linked list.

4.14 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*

4.14.1 Detailed Description

A struct of lit vertices.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 LVertex() [1/2]

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

Constructs a lit vertex.

4.14.2.2 LVertex() [2/2]

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

Constructs a lit vertex.

Parameters

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

4.14.3 Member Data Documentation**4.14.3.1 diffuse**

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

Diffuse colour of this lit vertex.

4.14.3.2 p

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

Position of this lit vertex (in world space).

4.14.3.3 tu

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

Texture u coordinate of this lit vertex.

4.14.3.4 tv

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

Texture v coordinate of this lit vertex.

4.15 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](#)

4.15.1 Detailed Description

A class of memory arena's.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 MemoryArena()

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

Constructs a memory arena with given block size.

Parameters

in	<i>block_size</i>	The block size in bytes.
----	-------------------	--------------------------

4.15.2.2 ~MemoryArena()

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

Destructs the given memory arena.

4.15.3 Member Function Documentation

4.15.3.1 Alloc() [1/2]

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

Allocates a block of memory of the given size.

Parameters

in	<i>size</i>	The requested size in bytes to allocate in memory.
----	-------------	--

Returns

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

4.15.3.2 Alloc() [2/2]

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

Allocates a block of memory.

Template Parameters

<i>T</i>	The type of objects to allocate in memory.
----------	--

Parameters

in	<i>count</i>	The number of objects of type <i>T</i> 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.

4.15.3.3 FreeAll()

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

Frees all blocks of this memory arena.

4.15.4 Member Data Documentation**4.15.4.1 m_available_blocks**

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

Pointers to the available blocks of this memory arena.

4.15.4.2 m_block_size

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

The fixed block size of this memory arena.

4.15.4.3 `m_current_block`

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

A pointer to the current block of this memory arena.

4.15.4.4 `m_current_block_pos`

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

The current block position of this memory arena.

4.15.4.5 `m_used_blocks`

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

Pointers to the used blocks of this memory arena.

4.16 `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`

4.16.1 Detailed Description

A class of mutexes.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 Mutex() [1/2]

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

Constructs a mutex.

4.16.2.2 Mutex() [2/2]

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

Constructs a mutex from the given mutex.

Parameters

in	<i>mutex</i>	A reference to a mutex.
----	--------------	-------------------------

4.16.2.3 ~Mutex()

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

Destructs this mutex.

4.16.3 Member Function Documentation

4.16.3.1 Create()

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

Creates a mutex.

4.16.3.2 Destroy()

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

Destroys a given mutex.

Parameters

in	<i>mutex</i>	The mutex to destroy.
----	--------------	-----------------------

4.16.3.3 `operator=()`

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

Copies the given mutex to this mutex.

Parameters

in	<i>mutex</i>	A reference to a mutex.
----	--------------	-------------------------

Returns

A reference to the copy of *mutex*.

4.16.4 Friends And Related Function Documentation

4.16.4.1 `MutexLock`

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

4.16.5 Member Data Documentation

4.16.5.1 `m_critical_section`

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

The critical section object of this mutex.

4.17 `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`

4.17.1 Detailed Description

A struct of mutex locks.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 MutexLock() [1/2]

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

Constructs a mutex lock for the given mutex.

Parameters

in	<i>mutex</i>	A reference to a mutex.
----	--------------	-------------------------

4.17.2.2 ~MutexLock()

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

Destructs this mutex lock.

4.17.2.3 MutexLock() [2/2]

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

Constructs a mutex lock from the given mutex lock.

Parameters

in	<i>mutex_lock</i>	A reference to a mutex lock.
----	-------------------	------------------------------

4.17.3 Member Function Documentation

4.17.3.1 operator=()

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

Copies the given mutex lock to this mutex lock.

Parameters

in	<i>mutex_lock</i>	A reference to a mutex lock.
----	-------------------	------------------------------

Returns

A reference to the copy of `mutex_lock`.

4.17.4 Member Data Documentation

4.17.4.1 `m_mutex`

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

The mutex of this mutex lock.

4.18 `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](#)

4.18.1 Detailed Description

A class of progress reporters.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 `ProgressReporter()`

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

Constructs a progress reporter.

Parameters

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

4.18.2.2 ~ProgressReporter()

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

Destructs this progress reporter.

4.18.3 Member Function Documentation

4.18.3.1 Done()

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

Finishes this progress reporter.

4.18.3.2 Update()

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

Updates this progress reporter.

Parameters

in	<i>nb_work</i>	The number of parts of the total work that are done.
----	----------------	--

4.18.4 Member Data Documentation

4.18.4.1 m_buffer

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

The output buffer of this progress reporter.

4.18.4.2 m_current_pos

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

The current (output) position of this progress reporter.

4.18.4.3 `m_fout`

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

The output file stream of this progress reporter.

4.18.4.4 `m_mutex`

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

The mutex needed for updating this progress reporter.

4.18.4.5 `m_nb_plusses_printed`

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

The total number of plusses that are already outputted.

4.18.4.6 `m_nb_plusses_total`

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

The total number of plusses to output.

4.18.4.7 `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.

4.18.4.8 `m_nb_work_total`

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

The number of parts of the total work.

4.18.4.9 `m_timer`

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

The timer of this progress reporter.

4.19 `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](#)

4.19.1 Detailed Description

A class of read write mutexes.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 [ReadWriteMutex\(\)](#) [1/2]

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

Constructs a read write mutex.

4.19.2.2 [ReadWriteMutex\(\)](#) [2/2]

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

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

Parameters

in	<i>mutex</i>	The read write mutex.
----	--------------	-----------------------

4.19.2.3 ~ReadWriteMutex()

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

Destructs this read write mutex.

4.19.3 Member Function Documentation

4.19.3.1 AcquireRead()

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

Acquires a read.

4.19.3.2 AcquireWrite()

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

Acquires a write.

4.19.3.3 Create()

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

Creates a mutex.

4.19.3.4 Destroy()

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

Destroys a given read write mutex.

Parameters

in	<i>mutex</i>	The read write mutex to destroy.
----	--------------	----------------------------------

4.19.3.5 operator=()

```
ReadWriteMutex& mage::ReadWriteMutex::operator= (
```

```
const ReadWriteMutex & mutex ) [private]
```

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

Parameters

in	<i>mutex</i>	A reference to a read write mutex.
----	--------------	------------------------------------

Returns

A reference to the copy of *mutex*.

4.19.3.6 ReleaseRead()

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

Release a read.

4.19.3.7 ReleaseWrite()

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

Release a write.

4.19.4 Friends And Related Function Documentation

4.19.4.1 ReadWriteMutexLock

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

4.19.5 Member Data Documentation

4.19.5.1 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.

4.19.5.2 m_critical_section

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

The critical section object of this read write mutex.

4.19.5.3 `m_nb_readers_waiting`

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

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

4.19.5.4 `m_nb_writers_waiting`

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

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

4.19.5.5 `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.

4.19.5.6 `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.

4.20 `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

4.20.1 Detailed Description

A struct of read write mutex locks.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 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	<i>mutex</i>	A reference to a read write mutex.
in	<i>lock_type</i>	The lock type.

4.20.2.2 ~ReadWriteMutexLock()

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

Destructs this read write mutex lock.

4.20.2.3 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	<i>mutex_lock</i>	A reference to a read write mutex lock.
----	-------------------	---

4.20.3 Member Function Documentation

4.20.3.1 DowngradeToRead()

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

Downgrades this read write lock to read.

4.20.3.2 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	<i>mutex_lock</i>	A reference to a read write mutex lock.
----	-------------------	---

Returns

A reference to the copy of *mutex_lock*.

4.20.3.3 UpgradeToWrite()

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

Upgrades this read write lock to write.

4.20.4 Member Data Documentation

4.20.4.1 m_mutex

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

The read write mutex of this read write mutex lock.

4.20.4.2 m_type

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

The lock type of this read write mutex lock.

4.21 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`

4.21.1 Detailed Description

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

A class of references.

Template Parameters

<i>T</i>	The type of reference.
----------	------------------------

4.21.2 Constructor & Destructor Documentation

4.21.2.1 Reference() [1/2]

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

Constructs a reference for the given pointer.

Parameters

in	<i>ptr</i>	The pointer.
----	------------	--------------

4.21.2.2 Reference() [2/2]

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

Constructs a reference from the given reference.

Parameters

in	<i>reference</i>	The reference.
----	------------------	----------------

4.21.2.3 ~Reference()

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

Destructs this reference.

4.21.3 Member Function Documentation

4.21.3.1 `GetPtr()`

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

Returns the pointer of this reference.

Returns

The pointer of this reference.

4.21.3.2 `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.

4.21.3.3 `operator->()` [1/2]

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

Dereferences this reference.

Returns

The pointer of this reference.

4.21.3.4 `operator->()` [2/2]

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

Dereferences this reference.

Returns

The pointer of this reference.

4.21.3.5 `operator=()` [1/2]

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

Copies the given pointer into a reference.

Parameters

in	<i>ptr</i>	The pointer.
----	------------	--------------

Returns

A reference for *ptr*.

4.21.3.6 operator=() [2/2]

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

Copies the given reference into a reference.

Parameters

in	<i>reference</i>	The reference.
----	------------------	----------------

Returns

A reference for *reference*.

4.21.4 Member Data Documentation**4.21.4.1 m_ptr**

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

The pointer of this reference.

4.22 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](#)

4.22.1 Detailed Description

A class of reference counted objects.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 ReferenceCounted()

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

Constructs a reference counted object.

4.22.3 Member Function Documentation

4.22.3.1 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.

4.22.3.2 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.

4.22.4 Member Data Documentation

4.22.4.1 m_reference_count

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

The reference count of this reference counted object.

4.23 mage::Resource Class Reference

```
#include <resource.hpp>
```

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](#)

4.23.1 Detailed Description

A class of resources.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 Resource()

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

Constructs a resource with a given name and path.

Parameters

in	<i>name</i>	A reference to the name.
in	<i>path</i>	A reference to the path.

4.23.2.2 ~Resource()

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

4.23.3 Member Function Documentation

4.23.3.1 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.

4.23.3.2 GetFilename()

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

Returns the filename of this resource.

Returns

The filename of this resource.

4.23.3.3 GetName()

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

Returns the name of this resource.

Returns

A reference to the name of this resource.

4.23.3.4 GetPath()

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

Returns the path of this resource.

Returns

A reference to the path of this resource.

4.23.3.5 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.

4.23.4 Friends And Related Function Documentation

4.23.4.1 ResourceManager

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

4.23.5 Member Data Documentation

4.23.5.1 m_name

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

The name of this resource.

4.23.5.2 m_path

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

The path of this resource.

4.23.5.3 m_resource_reference_count

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

The resource reference count of this resource.

4.24 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 * [Add](#) (const string &name, const string &path=".")
- void [Remove](#) (T **resource)
- void [EmptyDestroy](#) ()
- T * [GetResource](#) (const string &name, const string &path=".") const
- const [LinkedList](#)< T > * [GetResources](#) () const

Private Attributes

- [LinkedList](#)< T > * [m_resources](#)
- void(* [CreateResource](#))(T **resource, const string &name, const string &path)

4.24.1 Detailed Description

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

A class of resource managers.

Template Parameters

<i>T</i>	The type of resources.
----------	------------------------

4.24.2 Constructor & Destructor Documentation

4.24.2.1 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	<i>CreateResourceFunction</i>	The application specific resource creation function.
----	-------------------------------	--

4.24.2.2 ~ResourceManager()

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

Destructs this resource manager.

4.24.3 Member Function Documentation

4.24.3.1 Add()

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

Adds a new resource to this resource manager.

Parameters

in	<i>name</i>	A reference to the name of the new resource.
in	<i>path</i>	A reference to the path of the new resource.

Returns

A pointer to the resource.

4.24.3.2 EmptyDestroy()

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

Destroys all the resources of this resource manager.

4.24.3.3 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	<i>name</i>	A reference to the name of the new resource.
in	<i>path</i>	A reference to the path of the new resource.

Returns

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

4.24.3.4 `GetResources()`

```
template<typename T >
const LinkedList< T >* mage::ResourceManager< T >::GetResources ( ) const
```

4.24.3.5 `Remove()`

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

Removes the given resource from this resource manager.

Parameters

<code>in, out</code>	<code><i>resource</i></code>	A pointer to a pointer of the resource.
----------------------	------------------------------	---

4.24.4 Member Data Documentation

4.24.4.1 `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.

4.24.4.2 `m_resources`

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

The linked list containing the resources of this resource manager.

4.25 `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](#)

4.25.1 Detailed Description

A class of semaphores.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 Semaphore()

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

Constructs a semaphore.

4.25.2.2 ~Semaphore()

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

Destructs this semaphore.

4.25.3 Member Function Documentation

4.25.3.1 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	<i>count</i>	The increment value.
----	--------------	----------------------

4.25.3.2 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.

4.25.3.3 `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.

4.25.4 Member Data Documentation

4.25.4.1 `m_handle`

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

The handle of this semaphore.

4.26 `mage::Sphere` Struct Reference

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

Public Member Functions

- [Sphere](#) ()
- [Sphere](#) (XMFLOAT3 `p`, float `r`)
- bool [Encloses](#) (const [LinkedList](#)< XMFLOAT4 > &`planes`)
- bool [Collides](#) (const [Sphere](#) &`sphere`, const XMFLOAT3 `velocity_sum`, float *`collision_distance`)

Public Attributes

- XMFLOAT3 `p`
- float `r`

4.26.1 Detailed Description

A struct of spheres.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 `Sphere()` [1/2]

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

Constructs a sphere.

4.26.2.2 `Sphere()` [2/2]

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

Constructs a sphere.

Parameters

in	p	The position
in	r	The radius.

4.26.3 Member Function Documentation**4.26.3.1 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	<i>sphere</i>	The sphere.
in	<i>velocity_sum</i>	The sum of the velocities of both spheres.
out	<i>collision_distance</i>	The collision distance (in case of collision).

Returns

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

4.26.3.2 Encloses()

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

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

Parameters

in	<i>planes</i>	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.

4.26.4 Member Data Documentation**4.26.4.1 p**

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

The position of this sphere.

4.26.4.2 r

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

The radius of this sphere.

4.27 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](#)

4.27.1 Detailed Description

A class of states

4.27.2 Constructor & Destructor Documentation

4.27.2.1 State()

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

Constructs a state with given id.

Parameters

in	<i>id</i>	The id.
----	-----------	---------

4.27.2.2 ~State()

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

Destructs this state.

4.27.3 Member Function Documentation**4.27.3.1 Close()**

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

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

4.27.3.2 GetId()

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

Returns the id of this state.

Returns

The id of this state.

4.27.3.3 Load()

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

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

4.27.3.4 Render()

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

Render this state.

4.27.3.5 RequestViewer()

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

Requests the view setup details for the given frame.

Parameters

in	<i>viewer_setup</i>	A pointer to a viewer setup.
----	---------------------	------------------------------

4.27.3.6 Update()

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

Updates this state.

Parameters

in	<i>elapsed_time</i>	The elapsed time since the previous update.
----	---------------------	---

4.27.4 Member Data Documentation

4.27.4.1 m_id

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

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

4.28 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

- [LinkedList](#)< [State](#) > * [m_states](#)
- [State](#) * [m_current_state](#)
- bool [m_state_changed](#)

4.28.1 Detailed Description

A class of state managers.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 StateManager()

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

Constructs a state manager.

4.28.2.2 ~StateManager()

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

Destructs this state manager.

4.28.3 Member Function Documentation

4.28.3.1 AddState()

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

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

Parameters

in	<i>state</i>	A pointer to the state.
in	<i>change</i>	Flag indicating whether the current state of this engine need to be changed to <i>state</i> .

4.28.3.2 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	<i>id</i>	The id.
----	-----------	---------

4.28.3.3 ChangeState() [2/2]

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

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

Parameters

in	<i>new_state</i>	A pointer to the new state.
----	------------------	-----------------------------

4.28.3.4 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.

4.28.3.5 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.

4.28.3.6 RemoveState()

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

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

Parameters

in	<i>state</i>	A pointer to the state.
----	--------------	-------------------------

4.28.3.7 Update()

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

Updates this state manager and its current state.

Parameters

in	<i>elapsed_time</i>	The elapsed time since the previous update.
----	---------------------	---

Returns

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

4.28.4 Member Data Documentation

4.28.4.1 `m_current_state`

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

A pointer to the current state of this state manager.

4.28.4.2 `m_state_changed`

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

Flag indicating if the state changed in the current frame.

4.28.4.3 `m_states`

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

The states of this state manager.

4.29 `mage::Task` Class Reference

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

Public Member Functions

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

4.29.1 Detailed Description

A class of tasks.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 `~Task()`

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

Destructs this task.

4.29.3 Member Function Documentation

4.29.3.1 `Run()`

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

4.30 `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`

4.30.1 Detailed Description

A class of (high precision) timers.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 Timer()

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

Constructs a timer.

4.30.2.2 ~Timer()

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

Destructs this timer.

4.30.3 Member Function Documentation

4.30.3.1 Reset()

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

Resets this timer.

4.30.3.2 Restart()

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

Restarts this timer.

4.30.3.3 Start()

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

Starts this timer.

4.30.3.4 Stop()

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

Stops this timer.

4.30.3.5 Time()

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

Returns the elapsed time of this timer.

Returns

The elapsed time of this timer.

4.30.3.6 `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.

4.30.4 Member Data Documentation

4.30.4.1 `m_elapsed`

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

The elapsed time of this timer.

4.30.4.2 `m_performance_counter`

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

The counter of this timer.

4.30.4.3 `m_performance_frequency`

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

The frequency of this timer.

4.30.4.4 `m_performance_period`

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

The period of this timer.

4.30.4.5 `m_running`

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

Flag indicating whether this timer is running.

4.30.4.6 m_time0

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

The initial time stamp of this timer.

4.31 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](#)

4.31.1 Detailed Description

A struct of transformed and lit vertices.

4.31.2 Constructor & Destructor Documentation

4.31.2.1 TLVertex() [1/2]

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

Constructs a transformed and lit vertex.

4.31.2.2 TLVertex() [2/2]

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

Constructs a transformed and lit vertex.

Parameters

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

4.31.3 Member Data Documentation

4.31.3.1 `diffuse`

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

Diffuse colour of this transformed and lit vertex.

4.31.3.2 `p`

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

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

4.31.3.3 `tu`

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

Texture u coordinate of this transformed and lit vertex.

4.31.3.4 `tv`

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

Texture v coordinate of this transformed and lit vertex.

4.32 `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](#)

4.32.1 Detailed Description

A struct of vertices.

4.32.2 Constructor & Destructor Documentation

4.32.2.1 `Vertex()` [1/2]

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

Constructs a vertex.

4.32.2.2 `Vertex()` [2/2]

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

Constructs a vertex.

Parameters

in	<i>p</i>	Position of the vertex (in world space).
in	<i>n</i>	Normal of the vertex.
in	<i>tu</i>	Texture u coordinate of the vertex.
in	<i>tv</i>	Texture v coordinate of the vertex.

4.32.3 Member Data Documentation

4.32.3.1 `n`

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

Normal of this vertex.

4.32.3.2 `p`

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

Position of this vertex (in world space).

4.32.3.3 tu

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

Texture u coordinate of this vertex.

4.32.3.4 tv

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

Texture v coordinate of this vertex.

4.33 mage::ViewerSetup Struct Reference

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

Public Member Functions

- [ViewerSetup\(\)](#)

Public Attributes

- `uint64_t` [m_view_clear_flags](#)

4.33.1 Detailed Description

A struct of viewer setups.

4.33.2 Constructor & Destructor Documentation

4.33.2.1 ViewerSetup()

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

Constructs a viewer setup.

4.33.3 Member Data Documentation

4.33.3.1 m_view_clear_flags

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

Flags used for clearing the view.

