

Legend of the Great Unwashed

v0.1

Mon Nov 23 2015 13:39:00

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	11
5.1	MainNS Namespace Reference	11
5.1.1	Function Documentation	11
5.1.1.1	logError	11
5.2	mediawrap Namespace Reference	11
5.3	teamusa Namespace Reference	11
5.3.1	Typedef Documentation	13
5.3.1.1	ActorList	13
5.3.1.2	AudioID	13
5.3.1.3	BaseActorPtr	13
5.3.1.4	Region	13
5.3.1.5	TextureID	13
5.3.2	Enumeration Type Documentation	13
5.3.2.1	ActorEventType	13
5.3.2.2	CursorStyle	13
6	Class Documentation	15
6.1	teamusa::ActorEvent Class Reference	15
6.1.1	Detailed Description	15
6.1.2	Constructor & Destructor Documentation	15
6.1.2.1	ActorEvent	15

6.1.3	Member Data Documentation	15
6.1.3.1	type	15
6.1.3.2	value	15
6.2	teamusa::ActorVideo Class Reference	15
6.2.1	Detailed Description	16
6.2.2	Constructor & Destructor Documentation	16
6.2.2.1	ActorVideo	16
6.2.3	Member Data Documentation	16
6.2.3.1	layer	16
6.2.3.2	textureID	16
6.3	teamusa::AudioEngine Class Reference	16
6.3.1	Detailed Description	17
6.3.2	Member Function Documentation	17
6.3.2.1	deleteSound	17
6.3.2.2	deleteSoundGroup	17
6.3.2.3	loadSound	17
6.3.2.4	playSound	17
6.3.2.5	playStream	17
6.3.3	Member Data Documentation	18
6.3.3.1	audioPlayer	18
6.3.3.2	coreResources	18
6.3.3.3	levelResources	18
6.3.3.4	MAX_RESERVED_ID	18
6.4	mediawrap::AudioPlayer Class Reference	18
6.4.1	Detailed Description	19
6.4.2	Member Typedef Documentation	19
6.4.2.1	AudioID	19
6.4.3	Constructor & Destructor Documentation	19
6.4.3.1	AudioPlayer	19
6.4.3.2	~AudioPlayer	19
6.4.4	Member Function Documentation	19
6.4.4.1	clear_samples	19
6.4.4.2	delete_sample	19
6.4.4.3	load_sample	20
6.4.4.4	load_stream	20
6.4.4.5	play_sample	20
6.4.4.6	stream_audio	20
6.4.5	Member Data Documentation	20
6.4.5.1	audio_buffer	20
6.4.5.2	audio_channels	20

6.4.5.3	audio_format	20
6.4.5.4	audio_rate	20
6.4.5.5	audio_samples	20
6.4.5.6	audio_stream	21
6.5	teamusa::AudioStreamActor Class Reference	21
6.5.1	Detailed Description	21
6.5.2	Constructor & Destructor Documentation	21
6.5.2.1	AudioStreamActor	21
6.5.2.2	~AudioStreamActor	21
6.5.3	Member Function Documentation	21
6.5.3.1	getPath	22
6.5.3.2	step	22
6.5.4	Member Data Documentation	22
6.5.4.1	activated	22
6.5.4.2	path	22
6.6	teamusa::BaseActor Class Reference	22
6.6.1	Detailed Description	24
6.6.2	Constructor & Destructor Documentation	24
6.6.2.1	BaseActor	24
6.6.2.2	~BaseActor	24
6.6.3	Member Function Documentation	24
6.6.3.1	getLayer	24
6.6.3.2	getRegion	24
6.6.3.3	getTextureID	24
6.6.3.4	hasVideo	24
6.6.3.5	isInBounds	24
6.6.3.6	onClick	25
6.6.3.7	onHover	25
6.6.3.8	setRegion	25
6.6.3.9	step	25
6.6.4	Member Data Documentation	26
6.6.4.1	mAudioID	26
6.6.4.2	mRegion	26
6.6.4.3	mVideo	26
6.7	teamusa::DelayedAudioActor Class Reference	26
6.7.1	Detailed Description	27
6.7.2	Constructor & Destructor Documentation	27
6.7.2.1	DelayedAudioActor	27
6.7.2.2	~DelayedAudioActor	27
6.7.3	Member Function Documentation	27

6.7.3.1	step	27
6.7.4	Member Data Documentation	27
6.7.4.1	audiold	27
6.7.4.2	currentStep	27
6.7.4.3	delaySteps	27
6.8	teamusa::DelayedVideoActor Class Reference	27
6.8.1	Detailed Description	28
6.8.2	Constructor & Destructor Documentation	28
6.8.2.1	DelayedVideoActor	28
6.8.2.2	~DelayedVideoActor	28
6.8.3	Member Function Documentation	28
6.8.3.1	step	28
6.8.4	Member Data Documentation	29
6.8.4.1	currentStep	29
6.8.4.2	delaySteps	29
6.8.4.3	disappear	29
6.8.4.4	textureId	29
6.9	teamusa::Engine Class Reference	29
6.9.1	Detailed Description	30
6.9.2	Member Typedef Documentation	30
6.9.2.1	ActorEventHandler	30
6.9.3	Constructor & Destructor Documentation	30
6.9.3.1	Engine	30
6.9.3.2	~Engine	30
6.9.4	Member Function Documentation	30
6.9.4.1	freeAndLoadLevel	30
6.9.4.2	getMouseClickState	30
6.9.4.3	getMouseCoordinates	31
6.9.4.4	handleEvent	31
6.9.4.5	onChangeScene	31
6.9.4.6	onDisplayText	31
6.9.4.7	onExitGame	31
6.9.4.8	onLoadGame	31
6.9.4.9	onLoadLevel	31
6.9.4.10	onNewGame	32
6.9.4.11	onPlayAudio	32
6.9.4.12	onStreamAudio	32
6.9.4.13	render	32
6.9.4.14	run	32
6.9.5	Member Data Documentation	32

6.9.5.1	mActorEventHandlers	32
6.9.5.2	mAudioEngine	32
6.9.5.3	mCurrentLevelID	32
6.9.5.4	mIsRunning	32
6.9.5.5	mLevel	32
6.9.5.6	mMainMenu	33
6.9.5.7	mPlayer	33
6.9.5.8	mSerializer	33
6.9.5.9	mVideoEngine	33
6.10	teamusa::GameSaveSerializer Class Reference	33
6.10.1	Detailed Description	33
6.10.2	Constructor & Destructor Documentation	33
6.10.2.1	GameSaveSerializer	33
6.10.2.2	~GameSaveSerializer	33
6.10.3	Member Function Documentation	33
6.10.3.1	load	33
6.10.3.2	save	34
6.10.3.3	saveInThread	34
6.10.3.4	setSlot	34
6.10.4	Member Data Documentation	34
6.10.4.1	fileLock	34
6.10.4.2	slot	34
6.11	teamusa::InventoryItemActor Class Reference	34
6.11.1	Detailed Description	35
6.11.2	Constructor & Destructor Documentation	35
6.11.2.1	InventoryItemActor	35
6.11.2.2	~InventoryItemActor	35
6.11.3	Member Function Documentation	35
6.11.3.1	onClick	35
6.11.3.2	onHover	35
6.11.3.3	step	35
6.11.4	Member Data Documentation	36
6.11.4.1	itemID	36
6.11.4.2	pickedUp	36
6.12	teamusa::Level Class Reference	36
6.12.1	Detailed Description	37
6.12.2	Constructor & Destructor Documentation	37
6.12.2.1	Level	37
6.12.2.2	Level	37
6.12.3	Member Function Documentation	37

6.12.3.1	changeScene	37
6.12.3.2	clearAll	37
6.12.3.3	getActors	37
6.12.3.4	getBGImageID	38
6.12.3.5	getScene	38
6.12.3.6	loadLevel	38
6.12.3.7	parseAudioStreamActor	38
6.12.3.8	parseDelayedAudioActor	38
6.12.3.9	parseDelayedVideoActor	38
6.12.3.10	parseInventoryItemActor	38
6.12.3.11	parseLevelLink	38
6.12.3.12	parseMovingActor	38
6.12.3.13	parseResponsiveAudioActor	38
6.12.3.14	parseResponsiveVideoActor	38
6.12.3.15	parseSceneLink	38
6.12.3.16	parseTextboxSpawnActor	38
6.12.3.17	parseVideoActor	38
6.12.3.18	parseVideoEventActor	38
6.12.4	Member Data Documentation	38
6.12.4.1	activeScene	38
6.12.4.2	scenes	38
6.12.4.3	startScene	38
6.13	teamusa::LevelLink Class Reference	39
6.13.1	Detailed Description	39
6.13.2	Constructor & Destructor Documentation	39
6.13.2.1	LevelLink	39
6.13.2.2	~LevelLink	39
6.13.3	Member Function Documentation	39
6.13.3.1	getSceneID	40
6.13.3.2	getText	40
6.13.3.3	onClick	40
6.13.3.4	onHover	40
6.13.3.5	step	40
6.13.4	Member Data Documentation	41
6.13.4.1	itemRequiredText	41
6.13.4.2	levelID	41
6.13.4.3	requiredItemID	41
6.13.4.4	sceneID	41
6.14	teamusa::MovingActor Class Reference	41
6.14.1	Detailed Description	42

6.14.2	Constructor & Destructor Documentation	42
6.14.2.1	MovingActor	42
6.14.2.2	~MovingActor	42
6.14.3	Member Function Documentation	42
6.14.3.1	onClick	42
6.14.3.2	onHover	42
6.14.3.3	step	42
6.14.4	Member Data Documentation	43
6.14.4.1	currentStep	43
6.14.4.2	endRegion	43
6.14.4.3	hGrowth	43
6.14.4.4	isActive	43
6.14.4.5	transitionSteps	43
6.14.4.6	wGrowth	43
6.14.4.7	xSpeed	43
6.14.4.8	ySpeed	43
6.15	teamusa::Player Class Reference	43
6.15.1	Detailed Description	44
6.15.2	Member Typedef Documentation	44
6.15.2.1	Inventory	44
6.15.3	Constructor & Destructor Documentation	44
6.15.3.1	Player	44
6.15.3.2	~Player	44
6.15.4	Member Function Documentation	44
6.15.4.1	addItem	44
6.15.4.2	getCursorTextureID	45
6.15.4.3	getInventory	45
6.15.4.4	getPosition	45
6.15.4.5	hasItem	45
6.15.4.6	setCursor	45
6.15.4.7	setInventory	45
6.15.4.8	setPosition	45
6.15.4.9	setPosition	46
6.15.5	Member Data Documentation	46
6.15.5.1	CURSOR_DEFAULT_ID	46
6.15.5.2	CURSOR_DOWN_ID	46
6.15.5.3	CURSOR_LEFT_ID	46
6.15.5.4	CURSOR_RIGHT_ID	46
6.15.5.5	CURSOR_SELECT_ID	46
6.15.5.6	CURSOR_UP_ID	46

6.15.5.7	FLASHLIGHT_ID	46
6.15.5.8	mCursorStyle	46
6.15.5.9	mInventory	46
6.15.5.10	mLayer	46
6.15.5.11	MOUSE_CLICK_ID	46
6.15.5.12	mPosition	46
6.15.5.13	mRegion	46
6.15.5.14	mTextureID	46
6.16	teamusa::Point Class Reference	46
6.16.1	Detailed Description	47
6.16.2	Constructor & Destructor Documentation	47
6.16.2.1	Point	47
6.16.2.2	Point	47
6.16.3	Member Data Documentation	47
6.16.3.1	x	47
6.16.3.2	y	47
6.17	teamusa::ResponsiveAudioActor Class Reference	47
6.17.1	Detailed Description	48
6.17.2	Constructor & Destructor Documentation	48
6.17.2.1	ResponsiveAudioActor	48
6.17.2.2	~ResponsiveAudioActor	48
6.17.3	Member Function Documentation	48
6.17.3.1	onClick	48
6.17.3.2	onHover	48
6.17.3.3	step	49
6.17.4	Member Data Documentation	50
6.17.4.1	clickAudioId	50
6.17.4.2	hoverAudioId	50
6.18	teamusa::ResponsiveVideoActor Class Reference	50
6.18.1	Detailed Description	51
6.18.2	Constructor & Destructor Documentation	51
6.18.2.1	ResponsiveVideoActor	51
6.18.2.2	~ResponsiveVideoActor	51
6.18.3	Member Function Documentation	51
6.18.3.1	onClick	51
6.18.3.2	onHover	51
6.18.3.3	setTextureId	51
6.18.3.4	step	51
6.18.4	Member Data Documentation	52
6.18.4.1	clickTexture	52

6.18.4.2	defaultTextureId	52
6.18.4.3	hoverTexture	52
6.19	teamusa::Level::Scene Class Reference	52
6.19.1	Detailed Description	52
6.19.2	Member Data Documentation	52
6.19.2.1	actors	52
6.19.2.2	bgImageID	52
6.20	teamusa::SceneLink Class Reference	52
6.20.1	Detailed Description	53
6.20.2	Constructor & Destructor Documentation	53
6.20.2.1	SceneLink	53
6.20.2.2	~SceneLink	53
6.20.3	Member Function Documentation	53
6.20.3.1	getText	53
6.20.3.2	onClick	53
6.20.3.3	onHover	54
6.20.3.4	step	54
6.20.4	Member Data Documentation	54
6.20.4.1	cursorStyle	54
6.20.4.2	itemRequiredText	54
6.20.4.3	requiredItemID	54
6.20.4.4	sceneID	54
6.21	teamusa::TextboxSpawnActor Class Reference	55
6.21.1	Detailed Description	55
6.21.2	Constructor & Destructor Documentation	55
6.21.2.1	TextboxSpawnActor	55
6.21.2.2	~TextboxSpawnActor	55
6.21.3	Member Function Documentation	55
6.21.3.1	getText	55
6.21.3.2	onClick	55
6.21.3.3	step	56
6.21.4	Member Data Documentation	56
6.21.4.1	activated	56
6.21.4.2	text	56
6.22	teamusa::Timer Class Reference	56
6.22.1	Detailed Description	57
6.22.2	Constructor & Destructor Documentation	57
6.22.2.1	Timer	57
6.22.2.2	~Timer	57
6.22.3	Member Function Documentation	57

6.22.3.1	getTicks	57
6.22.3.2	pause	57
6.22.3.3	start	57
6.22.3.4	stop	57
6.22.3.5	unpause	57
6.22.4	Member Data Documentation	57
6.22.4.1	mPaused	57
6.22.4.2	mPauseTicks	57
6.22.4.3	mStarted	57
6.22.4.4	mStartTicks	58
6.23	teamusa::VideoActor Class Reference	58
6.23.1	Detailed Description	58
6.23.2	Constructor & Destructor Documentation	58
6.23.2.1	VideoActor	58
6.23.2.2	~VideoActor	58
6.23.3	Member Function Documentation	58
6.23.3.1	step	58
6.24	mediawrap::VideoContext Class Reference	59
6.24.1	Detailed Description	60
6.24.2	Member Typedef Documentation	60
6.24.2.1	Region	60
6.24.2.2	texture_iter	60
6.24.2.3	TextureID	60
6.24.3	Member Enumeration Documentation	61
6.24.3.1	BlendMode	61
6.24.3.2	DebugColor	61
6.24.3.3	Flip	61
6.24.4	Constructor & Destructor Documentation	61
6.24.4.1	VideoContext	61
6.24.4.2	~VideoContext	61
6.24.5	Member Function Documentation	62
6.24.5.1	create_texture	62
6.24.5.2	delete_texture	62
6.24.5.3	display	62
6.24.5.4	fill_texture	62
6.24.5.5	load_font	62
6.24.5.6	load_texture	63
6.24.5.7	render	63
6.24.5.8	render_clear	63
6.24.5.9	render_clear	63

6.24.5.10	render_onto	63
6.24.5.11	render_rotate	64
6.24.5.12	render_text	64
6.24.5.13	renderDebugBox	64
6.24.5.14	swapFullscreen	64
6.24.6	Member Data Documentation	64
6.24.6.1	font	64
6.24.6.2	renderer	64
6.24.6.3	textures	64
6.24.6.4	video_display	64
6.25	mediawrap::VideoDisplay Class Reference	65
6.25.1	Detailed Description	65
6.25.2	Constructor & Destructor Documentation	65
6.25.2.1	VideoDisplay	65
6.25.2.2	~VideoDisplay	65
6.25.3	Member Function Documentation	66
6.25.3.1	get_renderer	66
6.25.3.2	swapFullscreen	66
6.25.4	Member Data Documentation	66
6.25.4.1	window	66
6.26	teamusa::VideoEngine Class Reference	66
6.26.1	Detailed Description	67
6.26.2	Constructor & Destructor Documentation	67
6.26.2.1	VideoEngine	67
6.26.2.2	~VideoEngine	67
6.26.3	Member Function Documentation	68
6.26.3.1	clearLayers	68
6.26.3.2	deleteResourceGroup	68
6.26.3.3	deleteTexture	68
6.26.3.4	display	68
6.26.3.5	hideTextbox	68
6.26.3.6	isShowingTextbox	68
6.26.3.7	loadTexture	68
6.26.3.8	render	68
6.26.3.9	renderDebugBox	69
6.26.3.10	renderRotate	69
6.26.3.11	showTextbox	69
6.26.3.12	swapFullscreen	69
6.26.4	Member Data Documentation	69
6.26.4.1	coreResources	69

6.26.4.2	layers	69
6.26.4.3	levelResources	69
6.26.4.4	MAX_RESERVED_ID	69
6.26.4.5	NUM_LAYERS	69
6.26.4.6	SHADOW_LAYER	69
6.26.4.7	TEXT_LAYER	69
6.26.4.8	textboxActive	69
6.26.4.9	textboxPadding	69
6.26.4.10	textboxRegion	69
6.26.4.11	videoContext	70
6.27	teamusa::VideoEventActor Class Reference	70
6.27.1	Detailed Description	70
6.27.2	Constructor & Destructor Documentation	70
6.27.2.1	VideoEventActor	70
6.27.2.2	~VideoEventActor	70
6.27.3	Member Function Documentation	70
6.27.3.1	onClick	71
6.27.3.2	onHover	72
6.27.3.3	step	72
6.27.4	Member Data Documentation	72
6.27.4.1	actorEvent	72
7	File Documentation	73
7.1	ActorEvent.h File Reference	73
7.1.1	Detailed Description	73
7.2	Assert.h File Reference	73
7.2.1	Detailed Description	74
7.2.2	Macro Definition Documentation	74
7.2.2.1	Assert	74
7.3	AudioEngine.cpp File Reference	74
7.3.1	Detailed Description	74
7.4	AudioEngine.hpp File Reference	74
7.4.1	Detailed Description	74
7.5	AudioPlayer.cpp File Reference	75
7.5.1	Detailed Description	75
7.6	AudioPlayer.hpp File Reference	75
7.6.1	Detailed Description	75
7.7	AudioStreamActor.cpp File Reference	75
7.7.1	Detailed Description	75
7.8	AudioStreamActor.h File Reference	75

7.8.1 Detailed Description	76
7.9 BaseActor.cpp File Reference	76
7.9.1 Detailed Description	76
7.10 BaseActor.h File Reference	76
7.10.1 Detailed Description	77
7.11 CursorStyle.h File Reference	77
7.11.1 Detailed Description	77
7.12 DelayedAudioActor.cpp File Reference	77
7.12.1 Detailed Description	77
7.13 DelayedAudioActor.h File Reference	77
7.13.1 Detailed Description	78
7.14 DelayedVideoActor.cpp File Reference	78
7.14.1 Detailed Description	78
7.15 DelayedVideoActor.h File Reference	78
7.15.1 Detailed Description	78
7.16 Engine.cpp File Reference	78
7.16.1 Detailed Description	79
7.16.2 Macro Definition Documentation	79
7.16.2.1 BIND	79
7.16.3 Variable Documentation	79
7.16.3.1 FRAME_TIME	79
7.17 Engine.h File Reference	79
7.17.1 Detailed Description	80
7.18 GameSaveSerializer.cpp File Reference	80
7.18.1 Detailed Description	80
7.19 GameSaveSerializer.h File Reference	80
7.19.1 Detailed Description	80
7.20 Headers.h File Reference	80
7.20.1 Detailed Description	81
7.21 InventoryItemActor.cpp File Reference	81
7.21.1 Detailed Description	81
7.22 InventoryItemActor.h File Reference	81
7.22.1 Detailed Description	81
7.23 Level.cpp File Reference	82
7.23.1 Detailed Description	82
7.23.2 Function Documentation	82
7.23.2.1 loadError	82
7.23.2.2 operator>>	82
7.23.2.3 operator>>	82
7.24 Level.h File Reference	82

7.24.1 Detailed Description	83
7.25 LevelLink.cpp File Reference	83
7.25.1 Detailed Description	83
7.26 LevelLink.h File Reference	83
7.26.1 Detailed Description	83
7.27 main.cpp File Reference	84
7.27.1 Detailed Description	84
7.27.2 Function Documentation	84
7.27.2.1 main	84
7.28 MovingActor.cpp File Reference	84
7.28.1 Detailed Description	84
7.29 MovingActor.h File Reference	84
7.29.1 Detailed Description	85
7.30 Player.cpp File Reference	85
7.30.1 Detailed Description	85
7.31 Player.h File Reference	85
7.31.1 Detailed Description	85
7.32 Point.h File Reference	85
7.32.1 Detailed Description	86
7.33 ResourceGroup.hpp File Reference	86
7.33.1 Detailed Description	86
7.33.2 Enumeration Type Documentation	86
7.33.2.1 ResourceGroup	86
7.34 ResponsiveAudioActor.cpp File Reference	86
7.34.1 Detailed Description	86
7.35 ResponsiveAudioActor.h File Reference	87
7.35.1 Detailed Description	87
7.36 ResponsiveVideoActor.cpp File Reference	87
7.36.1 Detailed Description	87
7.37 ResponsiveVideoActor.h File Reference	87
7.37.1 Detailed Description	87
7.38 SceneLink.cpp File Reference	88
7.38.1 Detailed Description	88
7.39 SceneLink.h File Reference	88
7.39.1 Detailed Description	88
7.40 TextboxSpawnActor.cpp File Reference	88
7.40.1 Detailed Description	88
7.41 TextboxSpawnActor.h File Reference	88
7.41.1 Detailed Description	89
7.42 Timer.cpp File Reference	89

7.42.1 Detailed Description	89
7.43 Timer.h File Reference	89
7.43.1 Detailed Description	89
7.44 VideoActor.cpp File Reference	90
7.44.1 Detailed Description	90
7.45 VideoActor.h File Reference	90
7.45.1 Detailed Description	90
7.46 VideoContext.cpp File Reference	90
7.46.1 Detailed Description	90
7.47 VideoContext.hpp File Reference	90
7.47.1 Detailed Description	91
7.48 VideoDisplay.cpp File Reference	91
7.48.1 Detailed Description	91
7.49 VideoDisplay.hpp File Reference	91
7.49.1 Detailed Description	92
7.50 VideoEngine.cpp File Reference	92
7.50.1 Detailed Description	92
7.51 VideoEngine.hpp File Reference	92
7.51.1 Detailed Description	92
7.52 VideoEventActor.cpp File Reference	92
7.52.1 Detailed Description	93
7.53 VideoEventActor.h File Reference	93
7.53.1 Detailed Description	93
Index	95

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

MainNS	11
mediawrap	11
teamusa	11

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

teamusa::ActorEvent	15
teamusa::ActorVideo	15
teamusa::AudioEngine	16
mediawrap::AudioPlayer	18
teamusa::BaseActor	22
teamusa::AudioStreamActor	21
teamusa::DelayedAudioActor	26
teamusa::DelayedVideoActor	27
teamusa::InventoryItemActor	34
teamusa::LevelLink	39
teamusa::MovingActor	41
teamusa::ResponsiveAudioActor	47
teamusa::ResponsiveVideoActor	50
teamusa::SceneLink	52
teamusa::TextboxSpawnActor	55
teamusa::VideoActor	58
teamusa::VideoEventActor	70
teamusa::Engine	29
teamusa::GameSaveSerializer	33
teamusa::Level	36
teamusa::Player	43
teamusa::Point	46
teamusa::Level::Scene	52
teamusa::Timer	56
mediawrap::VideoContext	59
mediawrap::VideoDisplay	65
teamusa::VideoEngine	66

Chapter 3

Class Index

3.1 Class List

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

teamusa::ActorEvent	Event data generated by Actors, handled by Engine	15
teamusa::ActorVideo	Contains data for rendering actor	15
teamusa::AudioEngine	Provides project-specific audio functionality for Legend of the Great Unwashed	16
mediawrap::AudioPlayer	Provides basic audio playing capabilities with WAV files	18
teamusa::AudioStreamActor	If this actor is not activated, it will emit a StreamAudio event and set its status to activated when the step method is called	21
teamusa::BaseActor	Abstract class which all actors must derive from	22
teamusa::DelayedAudioActor	Will increment a counter every time the step method is called	26
teamusa::DelayedVideoActor	Will increment a counter every time the step method is called	27
teamusa::Engine	Processes all components of the game each frame	29
teamusa::GameSaveSerializer	Provides multithreaded save, single-thread load of save files	33
teamusa::InventoryItemActor	InventoryItemActor creates a collectible item in the game environment	34
teamusa::Level	A Level is a container of Scenes and Actors corresponding to those scenes	36
teamusa::LevelLink	Allows the player to transition between levels	39
teamusa::MovingActor	Will transition from one region to the next by calculating the distance to move each frame for a set number of frames	41
teamusa::Player	Handles all data relevant to the player engaging the game	43
teamusa::Point	An (x,y) coordinate within the rendering window	46
teamusa::ResponsiveAudioActor	Will increment the value of stepCount until it is equal to durationSteps for each call to the step method	47

teamusa::ResponsiveVideoActor	
Changes its texture ID based on hovering and clicks	50
teamusa::Level::Scene	
A scene is a collection of images (Actors) that is displayed on the screen	52
teamusa::SceneLink	
Allows the player to transition between scenes	52
teamusa::TextboxSpawnActor	
Will emit a DisplayText event when the onClick method is called	55
teamusa::Timer	
A timer that counts up from zero in milliseconds	56
teamusa::VideoActor	
Displays a texture in a region and performs no other behavior	58
mediawrap::VideoContext	
Provides basic 2D rendering capabilities	59
mediawrap::VideoDisplay	
Creates a window and initializes SDL2 and SDL2_IMG	65
teamusa::VideoEngine	
Provides video capabilities that are specific to Legend of the Great Unwashed	66
teamusa::VideoEventActor	
Will display a texture and perform no action until clicked	70

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

ActorEvent.h	Declares ActorEvent struct	73
Assert.h	Declares custom Assert macro	73
AudioEngine.cpp	Declares the AudioEngine class	74
AudioEngine.hpp	Declares the AudioEngine class	74
AudioPlayer.cpp	Implements the AudioPlayer class	75
AudioPlayer.hpp	Declares the AudioPlayer class	75
AudioStreamActor.cpp	Implements AudioStreamActor class	75
AudioStreamActor.h	Declares AudioStreamActor class	75
BaseActor.cpp	Implements BaseActor class	76
BaseActor.h	Declares BaseActor class	76
CursorStyle.h	Declares CursorStyle enumerations	77
DelayedAudioActor.cpp	Implements DelayedAudioActor class	77
DelayedAudioActor.h	Declares DelayedAudioActor class	77
DelayedVideoActor.cpp	Implements the DelayedVideoActor class	78
DelayedVideoActor.h	Declares DelayedVideoActor class	78
Engine.cpp	Implements Engine class	78
Engine.h	Declares Engine class	79
GameSaveSerializer.cpp	Implements save file serializer class	80
GameSaveSerializer.h	Declares save file serializer class	80

Headers.h	Easy way to include all headers needed	80
InventoryItemActor.cpp	Implements InventoryItemActor class	81
InventoryItemActor.h	Declares InventoryItemActor class	81
Level.cpp	Implements Level class	82
Level.h	Declares Level class	82
LevelLink.cpp	Implements LevelLink class	83
LevelLink.h	Declares LevelLink class	83
main.cpp	Entry point of program	84
MovingActor.cpp	Implements the MovingActor class	84
MovingActor.h	Declares MovingActor class	84
Player.cpp	Implements Player class	85
Player.h	Declares Player class	85
Point.h	Declares Point struct	85
ResourceGroup.hpp	Declares the ResourceGroup enum types	86
ResponsiveAudioActor.cpp	Implements ResponsiveAudioActor class	86
ResponsiveAudioActor.h	Declares ResponsiveAudioActor class	87
ResponsiveVideoActor.cpp	Implements the ResponsiveVideoActor class	87
ResponsiveVideoActor.h	Declares ResponsivevideoActor class	87
SceneLink.cpp	Implements SceneLink class	88
SceneLink.h	Declares SceneLink class	88
TextboxSpawnActor.cpp	Implements TextboxSpawnActor class	88
TextboxSpawnActor.h	Declares TextboxSpawnActor class	88
Timer.cpp	Implements Timer class	89
Timer.h	Declares Timer class	89
VideoActor.cpp	Implements VideoActor class	90
VideoActor.h	Declares the VideoActor class This module makes sure An actor that will only display a texture at a given region	90
VideoContext.cpp	Implements the VideoContext class	90
VideoContext.hpp	Declares the VideoContext class	90

VideoDisplay.cpp	
Implements the VideoDisplay class	91
VideoDisplay.hpp	
Declares the VideoDisplay class	91
VideoEngine.cpp	
Implements the VideoEngine class	92
VideoEngine.hpp	
Declares the VideoEngine class	92
VideoEventActor.cpp	
Implements the VideoEventActor class	92
VideoEventActor.h	
Declares VideoEventActor class	93

Chapter 5

Namespace Documentation

5.1 MainNS Namespace Reference

Functions

- static void [logError](#) (const std::string &desc)
Writes an error message to the log file.

5.1.1 Function Documentation

5.1.1.1 static void MainNS::logError (const std::string & desc) [static]

Writes an error message to the log file.

Parameters

<i>desc</i>	The string containing the error message.
-------------	--

5.2 mediawrap Namespace Reference

Classes

- class [AudioPlayer](#)
Provides basic audio playing capabilities with WAV files.
- class [VideoContext](#)
Provides basic 2D rendering capabilities.
- class [VideoDisplay](#)
Creates a window and initializes SDL2 and SDL2_IMG.

5.3 teamusa Namespace Reference

Classes

- class [ActorEvent](#)
Event data generated by Actors, handled by [Engine](#).
- class [ActorVideo](#)

- Contains data for rendering actor.*

 - class [AudioEngine](#)

Provides project-specific audio functionality for Legend of the Great Unwashed.
 - class [AudioStreamActor](#)

If this actor is not activated, it will emit a `StreamAudio` event and set its status to activated when the step method is called.
 - class [BaseActor](#)

Abstract class which all actors must derive from.
 - class [DelayedAudioActor](#)

Will increment a counter every time the step method is called.
 - class [DelayedVideoActor](#)

Will increment a counter every time the step method is called.
 - class [Engine](#)

Processes all components of the game each frame.
 - class [GameSaveSerializer](#)

Provides multithreaded save, single-thread load of save files.
 - class [InventoryItemActor](#)

InventoryItemActor creates a collectible item in the game environment.
 - class [Level](#)

A `Level` is a container of Scenes and Actors corresponding to those scenes.
 - class [LevelLink](#)

Allows the player to transition between levels.
 - class [MovingActor](#)

Will transition from one region to the next by calculating the distance to move each frame for a set number of frames.
 - class [Player](#)

Handles all data relevant to the player engaging the game.
 - class [Point](#)

An (x,y) coordinate within the rendering window.
 - class [ResponsiveAudioActor](#)

Will increment the value of `stepCount` until it is equal to `durationSteps` for each call to the step method.
 - class [ResponsiveVideoActor](#)

Changes its texture ID based on hovering and clicks.
 - class [SceneLink](#)

Allows the player to transition between scenes.
 - class [TextboxSpawnActor](#)

Will emit a `DisplayText` event when the `onClick` method is called.
 - class [Timer](#)

A timer that counts up from zero in milliseconds.
 - class [VideoActor](#)

Displays a texture in a region and performs no other behavior.
 - class [VideoEngine](#)

Provides video capabilities that are specific to Legend of the Great Unwashed.
 - class [VideoEventActor](#)

Will display a texture and perform no action until clicked.

Typedefs

- typedef [mediawrap::AudioPlayer::AudioID](#) [AudioID](#)
- typedef `std::shared_ptr<` [BaseActor](#) `>` [BaseActorPtr](#)
- typedef `std::vector<` [BaseActorPtr](#) `>` [ActorList](#)
- typedef [mediawrap::VideoContext::TextureID](#) [TextureID](#)
- typedef [mediawrap::VideoContext::Region](#) [Region](#)

Enumerations

- enum [ActorEventType](#) {
[Nil](#) = -1, [ChangeScene](#), [LoadLevel](#), [PlayAudio](#),
[NewGame](#), [LoadGame](#), [DisplayText](#), [ExitGame](#),
[StreamAudio](#) }
Events that actors can trigger.
- enum [CursorStyle](#) {
[CursorStyle::CURSOR_DEFAULT](#), [CursorStyle::CURSOR_SELECT](#), [CursorStyle::CURSOR_LEFT](#),
[CursorStyle::CURSOR_RIGHT](#),
[CursorStyle::CURSOR_UP](#), [CursorStyle::CURSOR_DOWN](#) }
The possible styles for the mouse cursor.

5.3.1 Typedef Documentation

5.3.1.1 typedef std::vector<[BaseActorPtr](#)> [teamusa::ActorList](#)

5.3.1.2 typedef [mediawrap::AudioPlayer::AudioID](#) [teamusa::AudioID](#)

5.3.1.3 typedef std::shared_ptr<[BaseActor](#)> [teamusa::BaseActorPtr](#)

5.3.1.4 typedef [mediawrap::VideoContext::Region](#) [teamusa::Region](#)

5.3.1.5 typedef [mediawrap::VideoContext::TextureID](#) [teamusa::TextureID](#)

5.3.2 Enumeration Type Documentation

5.3.2.1 enum [teamusa::ActorEventType](#)

Events that actors can trigger.

Enumerator

Nil
ChangeScene
LoadLevel
PlayAudio
NewGame
LoadGame
DisplayText
ExitGame
StreamAudio

5.3.2.2 enum [teamusa::CursorStyle](#) [[strong](#)]

The possible styles for the mouse cursor.

Enumerator

CURSOR_DEFAULT Default cursor.
CURSOR_SELECT Offers the ability to select an object.
CURSOR_LEFT Points left.

CURSOR_RIGHT Points right.

CURSOR_UP Points up.

CURSOR_DOWN Points down.

Chapter 6

Class Documentation

6.1 teamusa::ActorEvent Class Reference

Event data generated by Actors, handled by [Engine](#).

```
#include <ActorEvent.h>
```

Public Member Functions

- [ActorEvent](#) (void)

Public Attributes

- [int32_t value](#)
- [ActorEventType type](#)

6.1.1 Detailed Description

Event data generated by Actors, handled by [Engine](#).

6.1.2 Constructor & Destructor Documentation

6.1.2.1 [teamusa::ActorEvent::ActorEvent \(void \)](#) `[inline]`

6.1.3 Member Data Documentation

6.1.3.1 [ActorEventType](#) [teamusa::ActorEvent::type](#)

6.1.3.2 [int32_t](#) [teamusa::ActorEvent::value](#)

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

- [ActorEvent.h](#)

6.2 teamusa::ActorVideo Class Reference

Contains data for rendering actor.

```
#include <BaseActor.h>
```

Public Member Functions

- [ActorVideo](#) (void)

Public Attributes

- int32_t [layer](#)
- int32_t [textureID](#)

6.2.1 Detailed Description

Contains data for rendering actor.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 `teamusa::ActorVideo::ActorVideo (void) [inline]`

6.2.3 Member Data Documentation

6.2.3.1 `int32_t teamusa::ActorVideo::layer`

6.2.3.2 `int32_t teamusa::ActorVideo::textureID`

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

- [BaseActor.h](#)

6.3 teamusa::AudioEngine Class Reference

Provides project-specific audio functionality for Legend of the Great Unwashed.

```
#include <AudioEngine.hpp>
```

Public Member Functions

- void [loadSound](#) (const std::string &path, [AudioID](#) id, [ResourceGroup](#) group)
Loads the given sound file and associates it with the given id.
- void [playSound](#) ([AudioID](#) id)
Plays the sound associated with the given id.
- void [playStream](#) (const std::string &path)
Plays the given stream in a loop continuously.
- void [deleteSound](#) ([AudioID](#) id)
Deletes the given sound from memory.
- void [deleteSoundGroup](#) ([ResourceGroup](#) resourceGroup)
Deletes the entire group of sounds.

Private Attributes

- `std::vector< AudioID > coreResources`
- `std::vector< AudioID > levelResources`
- `AudioPlayer audioPlayer`

Static Private Attributes

- `static const AudioID MAX_RESERVED_ID = 1000`

6.3.1 Detailed Description

Provides project-specific audio functionality for Legend of the Great Unwashed.

6.3.2 Member Function Documentation

6.3.2.1 `void teamusa::AudioEngine::deleteSound (AudioID id)`

Deletes the given sound from memory.

Parameters

<i>id</i>	The id of the audio to delete.
-----------	--------------------------------

6.3.2.2 `void teamusa::AudioEngine::deleteSoundGroup (ResourceGroup resourceGroup)`

Deletes the entire group of sounds.

Parameters

--	--

6.3.2.3 `void teamusa::AudioEngine::loadSound (const std::string & path, AudioID id, ResourceGroup group)`

Loads the given sound file and associates it with the given id.

Parameters

<i>path</i>	The relative path of the sound file to load.
<i>id</i>	The id to associate with the given sound file.

6.3.2.4 `void teamusa::AudioEngine::playSound (AudioID id)`

Plays the sound associated with the given id.

Parameters

<i>id</i>	The id of the sound to play.
-----------	------------------------------

6.3.2.5 `void teamusa::AudioEngine::playStream (const std::string & path)`

Plays the given stream in a loop continuously.

Parameters

<i>path</i>	The path of the audio to stream.
-------------	----------------------------------

6.3.3 Member Data Documentation

6.3.3.1 **AudioPlayer** teamusa::AudioEngine::audioPlayer [private]

6.3.3.2 **std::vector<AudioID>** teamusa::AudioEngine::coreResources [private]

6.3.3.3 **std::vector<AudioID>** teamusa::AudioEngine::levelResources [private]

6.3.3.4 **const AudioID** teamusa::AudioEngine::MAX_RESERVED_ID = 1000 [static], [private]

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

- [AudioEngine.hpp](#)
- [AudioEngine.cpp](#)

6.4 mediawrap::AudioPlayer Class Reference

Provides basic audio playing capabilities with WAV files.

```
#include <AudioPlayer.hpp>
```

Public Types

- typedef unsigned int [AudioID](#)
Used to uniquely identify each audio sample.

Public Member Functions

- [AudioPlayer](#) ()
Constructs a new audio player.
- [~AudioPlayer](#) ()
Deletes the audio player and all of its samples and streams.
- void [load_stream](#) (const std::string &file_path)
Loads the given audio file and prepares it for streaming.
- void [stream_audio](#) (int loops=-1)
Plays the loaded audio stream loop+1 times.
- void [load_sample](#) ([AudioID](#) id, const std::string &file_path)
Loads the given audio sample into memory.
- void [play_sample](#) ([AudioID](#) id)
Plays the given audio sample in the first available channel.
- void [delete_sample](#) ([AudioID](#) id)
Deletes the sample created by a call to [load_sample\(\)](#).
- void [clear_samples](#) ()
Deletes all samples created by a call to [load_sample\(\)](#).

Private Attributes

- `std::unordered_map< AudioID, Mix_Chunk * > * audio_samples`
- `Mix_Music * audio_stream`

Static Private Attributes

- `static const int audio_rate = 44100`
- `static const int audio_channels = 1`
- `static const int audio_buffer = 4096`
- `static const Uint16 audio_format = AUDIO_S16`

6.4.1 Detailed Description

Provides basic audio playing capabilities with WAV files.

Acts as an abstraction layer for SDL2.

6.4.2 Member Typedef Documentation

6.4.2.1 `typedef unsigned int mediawrap::AudioPlayer::AudioID`

Used to uniquely identify each audio sample.

6.4.3 Constructor & Destructor Documentation

6.4.3.1 `mediawrap::AudioPlayer::AudioPlayer ()`

Constructs a new audio player.

Enables SDL audio functionality.

6.4.3.2 `mediawrap::AudioPlayer::~~AudioPlayer ()`

Deletes the audio player and all of its samples and streams.

Disables SDL audio functionality.

6.4.4 Member Function Documentation

6.4.4.1 `void mediawrap::AudioPlayer::clear_samples ()`

Deletes all samples created by a call to [load_sample\(\)](#).

6.4.4.2 `void mediawrap::AudioPlayer::delete_sample (AudioID id)`

Deletes the sample created by a call to [load_sample\(\)](#).

Parameters

<i>id</i>	The id of the sample to delete.
-----------	---------------------------------

6.4.4.3 void mediawrap::AudioPlayer::load_sample (AudioID id, const std::string & file_path)

Loads the given audio sample into memory.

Loading a sample into an existing id will delete the sample associated with it before the new sample is loaded.

Parameters

<i>id</i>	The unique id to store the sample under.
<i>file_path</i>	The path of the audio file to load into memory.

6.4.4.4 void mediawrap::AudioPlayer::load_stream (const std::string & file_path)

Loads the given audio file and prepares it for streaming.

Only one audio stream can be loaded at a time. The previously loaded stream will be deleted if this method is called multiple times.

Parameters

<i>file_path</i>	The path of the file to load.
------------------	-------------------------------

6.4.4.5 void mediawrap::AudioPlayer::play_sample (AudioID id)

Plays the given audio sample in the first available channel.

Parameters

<i>id</i>	The id of the audio sample to play.
-----------	-------------------------------------

6.4.4.6 void mediawrap::AudioPlayer::stream_audio (int loops = -1)

Plays the loaded audio stream loop+1 times.

If set to -1, the audio will loop indefinitely. Only one audio stream can be played at a time.

Parameters

<i>loops</i>	The number of times to play the audio. A value of -1 is infinite. Defaults to looping infinitely.
--------------	---

6.4.5 Member Data Documentation

6.4.5.1 const int mediawrap::AudioPlayer::audio_buffer = 4096 [static], [private]

6.4.5.2 const int mediawrap::AudioPlayer::audio_channels = 1 [static], [private]

6.4.5.3 const Uint16 mediawrap::AudioPlayer::audio_format = AUDIO_S16 [static], [private]

6.4.5.4 const int mediawrap::AudioPlayer::audio_rate = 44100 [static], [private]

6.4.5.5 std::unordered_map<AudioID, Mix_Chunk*>* mediawrap::AudioPlayer::audio_samples [private]

6.4.5.6 Mix_Music* mediawrap::AudioPlayer::audio_stream [private]

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

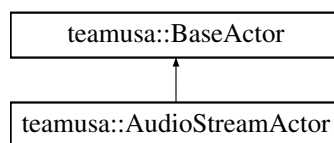
- [AudioPlayer.hpp](#)
- [AudioPlayer.cpp](#)

6.5 teamusa::AudioStreamActor Class Reference

If this actor is not activated, it will emit a StreamAudio event and set its status to activated when the step method is called.

```
#include <AudioStreamActor.h>
```

Inheritance diagram for teamusa::AudioStreamActor:



Public Member Functions

- [AudioStreamActor](#) (std::string [path](#))
- virtual [~AudioStreamActor](#) (void) override
- virtual const [ActorEvent](#) [step](#) ([Player](#) &player) override
This method updates the player on every frame.
- std::string [getPath](#) ()
This method gets the path to the requested audio file.

Private Attributes

- std::string [path](#)
- bool [activated](#)

Additional Inherited Members

6.5.1 Detailed Description

If this actor is not activated, it will emit a StreamAudio event and set its status to activated when the step method is called.

The engine can then retrieve the path to the audio file by a call to this actor's getPath method.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 [AudioStreamActor::AudioStreamActor](#) (std::string *path*) [explicit]

6.5.2.2 [AudioStreamActor::~~AudioStreamActor](#) (void) [override],[virtual]

6.5.3 Member Function Documentation

6.5.3.1 `std::string AudioStreamActor::getPath ()`

This method gets the path to the requested audio file.

Returns

Returns a file path formatted as a string.

6.5.3.2 `const ActorEvent AudioStreamActor::step (Player & player) [override],[virtual]`

This method updates the player on every frame.

Parameters

Player	The player
------------------------	------------

Returns

Returns an [ActorEvent](#) that triggers an action from one or more actors.

Implements [teamusa::BaseActor](#).

6.5.4 Member Data Documentation

6.5.4.1 `bool teamusa::AudioStreamActor::activated [private]`

6.5.4.2 `std::string teamusa::AudioStreamActor::path [private]`

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

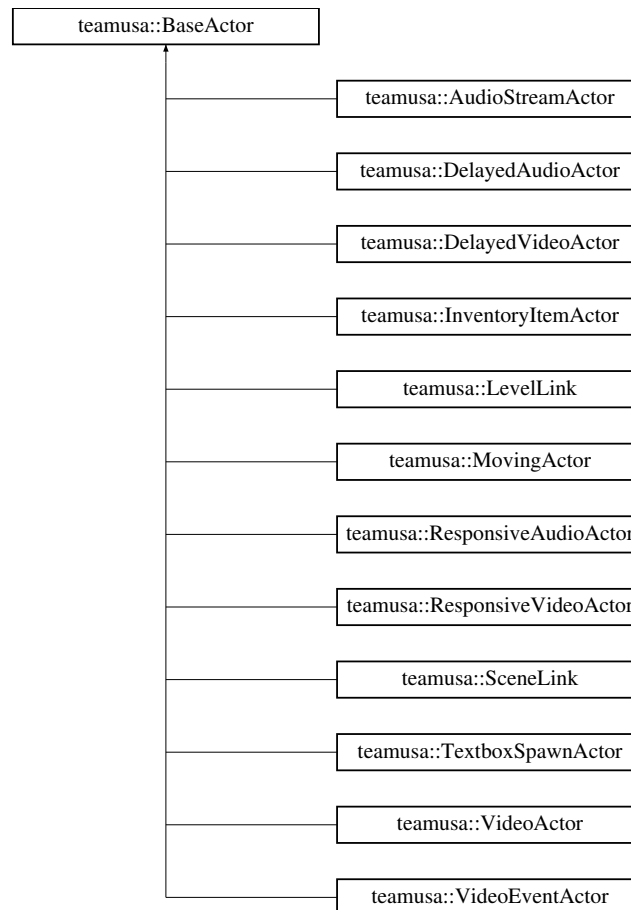
- [AudioStreamActor.h](#)
- [AudioStreamActor.cpp](#)

6.6 teamusa::BaseActor Class Reference

Abstract class which all actors must derive from.

```
#include <BaseActor.h>
```

Inheritance diagram for teamusa::BaseActor:



Public Member Functions

- **BaseActor** (const **Region** ®ion=**Region**())
- virtual **~BaseActor** (void)=0
- virtual const **ActorEvent** **onClick** (**Player** &player)
Called when the actor is clicked on.
- virtual const **ActorEvent** **onHover** (**Player** &player)
Called when the actor is hovered over with the mouse.
- virtual const **ActorEvent** **step** (**Player** &player)=0
Called each frame, each derived actor should handle this.
- virtual const bool **isInBounds** (const **Point** &point)
Calculates if point is in bounds of actor's region.
- virtual void **setRegion** (const **Region** ®ion)
*Sets the actor's region (can be used by **Level** when loading).*
- virtual const **Region** **getRegion** (void) const
Gets the actor's Region.
- virtual const int32_t **getLayer** (void) const
Gets the layer the actor should be rendered on.
- virtual const int32_t **getTextureID** (void) const
Gets the texture ID of the actor.
- const bool **hasVideo** (void) const
Returns true if the actor has a video component.

Protected Attributes

- [Region](#) mRegion
- [AudioID](#) mAudioID
- [ActorVideo](#) * mVideo

6.6.1 Detailed Description

Abstract class which all actors must derive from.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 `BaseActor::BaseActor (const Region & region = Region())` `[explicit]`

6.6.2.2 `BaseActor::~~BaseActor (void)` `[pure virtual]`

6.6.3 Member Function Documentation

6.6.3.1 `const int32_t BaseActor::getLayer (void) const` `[virtual]`

Gets the layer the actor should be rendered on.

Returns

An integer containing the layer.

6.6.3.2 `const Region BaseActor::getRegion (void) const` `[virtual]`

Gets the actor's Region.

Returns

The actor's Region struct.

6.6.3.3 `const int32_t BaseActor::getTextureID (void) const` `[virtual]`

Gets the texture ID of the actor.

Returns

The integer containing the texture ID.

6.6.3.4 `const bool BaseActor::hasVideo (void) const`

Returns true if the actor has a video component.

6.6.3.5 `const bool BaseActor::isInBounds (const Point & point)` `[virtual]`

Calculates if point is in bounds of actor's region.

Parameters

<i>point</i>	The point to test.
--------------	--------------------

Returns

True if point is within actor's region.

6.6.3.6 `const ActorEvent BaseActor::onClick (Player & player)` [virtual]

Called when the actor is clicked on.

Parameters

<i>player</i>	The player in the scene.
---------------	--------------------------

Returns

The [ActorEvent](#) to be handled by [Engine](#) when clicked on.

Reimplemented in [teamusa::MovingActor](#), [teamusa::InventoryItemActor](#), [teamusa::TextboxSpawnActor](#), [teamusa::ResponsiveAudioActor](#), [teamusa::VideoEventActor](#), [teamusa::SceneLink](#), [teamusa::LevelLink](#), and [teamusa::ResponsiveVideoActor](#).

6.6.3.7 `const ActorEvent BaseActor::onHover (Player & player)` [virtual]

Called when the actor is hovered over with the mouse.

Parameters

<i>player</i>	The player in the scene.
---------------	--------------------------

Returns

The [ActorEvent](#) to be handled by [Engine](#) when hovered over.

Reimplemented in [teamusa::MovingActor](#), [teamusa::ResponsiveAudioActor](#), [teamusa::VideoEventActor](#), [teamusa::SceneLink](#), [teamusa::LevelLink](#), [teamusa::ResponsiveVideoActor](#), and [teamusa::InventoryItemActor](#).

6.6.3.8 `void BaseActor::setRegion (const Region & region)` [virtual]

Sets the actor's region (can be used by [Level](#) when loading).

Parameters

<i>region</i>	The Region to set.
---------------	--------------------

6.6.3.9 `virtual const ActorEvent teamusa::BaseActor::step (Player & player)` [pure virtual]

Called each frame, each derived actor should handle this.

Parameters

<i>player</i>	The player in the scene.
---------------	--------------------------

Returns

Any [ActorEvent](#) that should be handled immediately by [Engine](#).

Implemented in [teamusa::MovingActor](#), [teamusa::ResponsiveAudioActor](#), [teamusa::VideoEventActor](#), [teamusa::SceneLink](#), [teamusa::InventoryItemActor](#), [teamusa::LevelLink](#), [teamusa::ResponsiveVideoActor](#), [teamusa::TextboxSpawnActor](#), [teamusa::DelayedVideoActor](#), [teamusa::DelayedAudioActor](#), [teamusa::AudioStreamActor](#), and [teamusa::VideoActor](#).

6.6.4 Member Data Documentation

6.6.4.1 **AudioID** [teamusa::BaseActor::mAudioID](#) [protected]

6.6.4.2 **Region** [teamusa::BaseActor::mRegion](#) [protected]

6.6.4.3 **ActorVideo*** [teamusa::BaseActor::mVideo](#) [protected]

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

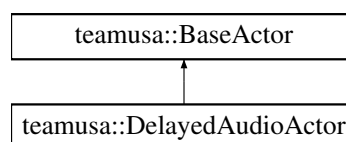
- [BaseActor.h](#)
- [BaseActor.cpp](#)

6.7 teamusa::DelayedAudioActor Class Reference

Will increment a counter every time the step method is called.

```
#include <DelayedAudioActor.h>
```

Inheritance diagram for [teamusa::DelayedAudioActor](#):



Public Member Functions

- [DelayedAudioActor](#) (int audioID, int [delaySteps](#))
- virtual [~DelayedAudioActor](#) (void) override
- virtual const [ActorEvent](#) [step](#) ([Player](#) &player) override

Advances the actor one frame.

Private Attributes

- int [audioid](#)
- int [delaySteps](#)
- int [currentStep](#)

Additional Inherited Members

6.7.1 Detailed Description

Will increment a counter every time the step method is called.

After a specified number of steps have occurred, this actor will change its TextureID to a valid value and will be displayed. When the number of steps is equal to the disappearing step, the TextureID will be set to an ignored value, causing the actor to disappear.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 `DelayedAudioActor::DelayedAudioActor (int audioID, int delaySteps = 0)` `[explicit]`

6.7.2.2 `DelayedAudioActor::~~DelayedAudioActor (void)` `[override]`, `[virtual]`

6.7.3 Member Function Documentation

6.7.3.1 `const ActorEvent DelayedAudioActor::step (Player & player)` `[override]`, `[virtual]`

Advances the actor one frame.

Parameters

Player	The Player .
------------------------	------------------------------

Returns

Returns an [ActorEvent](#) that triggers one or more actors to perform an action

Implements [teamusa::BaseActor](#).

6.7.4 Member Data Documentation

6.7.4.1 `int teamusa::DelayedAudioActor::audiold` `[private]`

6.7.4.2 `int teamusa::DelayedAudioActor::currentStep` `[private]`

6.7.4.3 `int teamusa::DelayedAudioActor::delaySteps` `[private]`

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

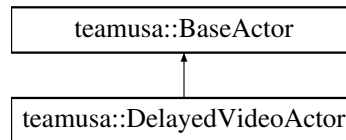
- [DelayedAudioActor.h](#)
- [DelayedAudioActor.cpp](#)

6.8 teamusa::DelayedVideoActor Class Reference

Will increment a counter every time the step method is called.

```
#include <DelayedVideoActor.h>
```

Inheritance diagram for teamusa::DelayedVideoActor:



Public Member Functions

- [DelayedVideoActor](#) ([Region](#) region, int textureID, int delaysteps, int disappearStep, int layer)
- virtual [~DelayedVideoActor](#) (void) override
- virtual const [ActorEvent](#) step ([Player](#) &player) override

Advances the actor one frame.

Private Attributes

- int [textureId](#)
- int [delaySteps](#)
- int [currentStep](#)
- int [disappear](#)

Additional Inherited Members

6.8.1 Detailed Description

Will increment a counter every time the step method is called.

After a specified number of steps have occurred, this actor will change its TextureID to a valid value and will be displayed. When the number of steps is equal to the disappearing step, the TextureID will be set to an ignored value, causing the actor to disappear

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `DelayedVideoActor::DelayedVideoActor (Region region, int textureID, int delaysteps, int disappearStep, int layer)` `[explicit]`

6.8.2.2 `DelayedVideoActor::~~DelayedVideoActor (void)` `[override],[virtual]`

6.8.3 Member Function Documentation

6.8.3.1 `const ActorEvent DelayedVideoActor::step (Player & player)` `[override],[virtual]`

Advances the actor one frame.

Parameters

Player	The Player .
------------------------	------------------------------

Returns

Returns an [ActorEvent](#) that triggers one or more actors to perform an action.

Implements [teamusa::BaseActor](#).

6.8.4 Member Data Documentation

6.8.4.1 int teamusa::DelayedVideoActor::currentStep [private]

6.8.4.2 int teamusa::DelayedVideoActor::delaySteps [private]

6.8.4.3 int teamusa::DelayedVideoActor::disappear [private]

6.8.4.4 int teamusa::DelayedVideoActor::textureId [private]

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

- [DelayedVideoActor.h](#)
- [DelayedVideoActor.cpp](#)

6.9 teamusa::Engine Class Reference

Processes all components of the game each frame.

```
#include <Engine.h>
```

Public Member Functions

- [Engine](#) (void)
- [~Engine](#) (void)
- void [run](#) (void)
Starts the game, runs until the player quits or there is an exception.

Private Types

- typedef std::function< void([BaseActorPtr](#) actor, const int32_t value) > [ActorEventHandler](#)

Private Member Functions

- const [Point](#) [getMouseCoordinates](#) (void) const
Retrieves the window mouse coordinates.
- const int32_t [getMouseClickState](#) (void) const
Retrives the current mouse button state.
- void [handleEvent](#) ([BaseActorPtr](#) actor, const [ActorEvent](#) &e)
Handles actor event on actor who triggered it.
- void [render](#) (const [ActorList](#) &actors)
Renders all actors in the scene.
- void [onChangeScene](#) ([BaseActorPtr](#) actor, const int32_t value)
Handles scene change events triggered by [SceneLink](#) actors.
- void [onLoadLevel](#) ([BaseActorPtr](#) actor, const int32_t value)
Handles level change events triggered by [Levellink](#) actors.
- void [onPlayAudio](#) ([BaseActorPtr](#) actor, const int32_t value)
Handles audio events triggered by actors.
- void [onNewGame](#) ([BaseActorPtr](#) actor, const int32_t value)
Handles new game events triggered by main menu actors.
- void [onLoadGame](#) ([BaseActorPtr](#) actor, const int32_t value)

Handles load game events triggered by main menu actors.

- void [onDisplayText](#) ([BaseActorPtr](#) actor, const int32_t value)

Handles text display events triggered by actors.

- void [onExitGame](#) ([BaseActorPtr](#) actor, const int32_t value)

Handles exit game events triggered by quit game button at main menu.

- void [onStreamAudio](#) ([BaseActorPtr](#) actor, const int32_t value)

Handles stream audio events triggered by actors, calls into [AudioEngine](#).

- void [freeAndLoadLevel](#) (const int32_t id)

Clears resource data for current level and loads the specified level.

Private Attributes

- std::shared_ptr< [AudioEngine](#) > [mAudioEngine](#)
- std::shared_ptr< [VideoEngine](#) > [mVideoEngine](#)
- [Level](#) [mLevel](#)
- int32_t [mCurrentLevelID](#)
- [Player](#) [mPlayer](#)
- bool [mIsRunning](#)
- bool [mMainMenu](#)
- [GameSaveSerializer](#) [mSerializer](#)
- std::vector< [ActorEventHandler](#) > [mActorEventHandlers](#)

6.9.1 Detailed Description

Processes all components of the game each frame.

6.9.2 Member Typedef Documentation

- 6.9.2.1 `typedef std::function< void(BaseActorPtr actor, const int32_t value) > teamusa::Engine::ActorEventHandler [private]`

6.9.3 Constructor & Destructor Documentation

- 6.9.3.1 `Engine::Engine (void) [explicit]`

- 6.9.3.2 `Engine::~~Engine (void)`

6.9.4 Member Function Documentation

- 6.9.4.1 `void Engine::freeAndLoadLevel (const int32_t id) [private]`

Clears resource data for current level and loads the specified level.

- 6.9.4.2 `const int32_t Engine::getMouseClickedState (void) const [private]`

Retrives the current mouse button state.

Returns

Integer describing mouse state.

6.9.4.3 `const Point Engine::getMouseCoordinates (void) const` `[private]`

Retrieves the window mouse coordinates.

Returns

A [Point](#) struct containing the x and y values of the mouse.

6.9.4.4 `void Engine::handleEvent (BaseActorPtr actor, const ActorEvent & e)` `[private]`

Handles actor event on actor who triggered it.

Looks up function pointer in table, calls the corresponding function.

6.9.4.5 `void Engine::onChangeScene (BaseActorPtr actor, const int32_t value)` `[private]`

Handles scene change events triggered by [SceneLink](#) actors.

Parameters

<i>actor</i>	The actor who triggered the event.
<i>value</i>	A value corresponding to the event, if needed.

6.9.4.6 `void Engine::onDisplayText (BaseActorPtr actor, const int32_t value)` `[private]`

Handles text display events triggered by actors.

Parameters

<i>actor</i>	The actor who triggered the event.
<i>value</i>	A value corresponding to the event, if needed.

6.9.4.7 `void Engine::onExitGame (BaseActorPtr actor, const int32_t value)` `[private]`

Handles exit game events triggered by quit game button at main menu.

Parameters

<i>actor</i>	The actor who triggered the event.
<i>value</i>	A value corresponding to the event, if needed.

6.9.4.8 `void Engine::onLoadGame (BaseActorPtr actor, const int32_t value)` `[private]`

Handles load game events triggered by main menu actors.

Parameters

<i>actor</i>	The actor who triggered the event.
<i>value</i>	A value corresponding to the event, if needed.

6.9.4.9 `void Engine::onLoadLevel (BaseActorPtr actor, const int32_t value)` `[private]`

Handles level change events triggered by [LevelLink](#) actors.

Parameters

<i>actor</i>	The actor who triggered the event.
<i>value</i>	A value corresponding to the event, if needed.

6.9.4.10 `void Engine::onNewGame (BaseActorPtr actor, const int32_t value)` [private]

Handles new game events triggered by main menu actors.

Parameters

<i>actor</i>	The actor who triggered the event.
<i>value</i>	A value corresponding to the event, if needed.

6.9.4.11 `void Engine::onPlayAudio (BaseActorPtr actor, const int32_t value)` [private]

Handles audio events triggered by actors.

Calls into the [AudioEngine](#).

Parameters

<i>actor</i>	The actor who triggered the event.
<i>value</i>	A value corresponding to the event, if needed.

6.9.4.12 `void Engine::onStreamAudio (BaseActorPtr actor, const int32_t value)` [private]

Handles stream audio events triggered by actors, calls into [AudioEngine](#).

Parameters

<i>actor</i>	The actor who triggered the event.
<i>value</i>	A value corresponding to the event, if needed.

6.9.4.13 `void Engine::render (const ActorList & actors)` [private]

Renders all actors in the scene.

6.9.4.14 `void Engine::run (void)`

Starts the game, runs until the player quits or there is an exception.

6.9.5 Member Data Documentation

6.9.5.1 `std::vector<ActorEventHandler> teamusa::Engine::mActorEventHandlers` [private]

6.9.5.2 `std::shared_ptr<AudioEngine> teamusa::Engine::mAudioEngine` [private]

6.9.5.3 `int32_t teamusa::Engine::mCurrentLevelID` [private]

6.9.5.4 `bool teamusa::Engine::mIsRunning` [private]

6.9.5.5 `Level teamusa::Engine::mLevel` [private]

6.9.5.6 `bool teamusa::Engine::mMainMenu` `[private]`

6.9.5.7 `Player teamusa::Engine::mPlayer` `[private]`

6.9.5.8 `GameSaveSerializer teamusa::Engine::mSerializer` `[private]`

6.9.5.9 `std::shared_ptr<VideoEngine> teamusa::Engine::mVideoEngine` `[private]`

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

- [Engine.h](#)
- [Engine.cpp](#)

6.10 teamusa::GameSaveSerializer Class Reference

Provides multithreaded save, single-thread load of save files.

```
#include <GameSaveSerializer.h>
```

Public Member Functions

- [GameSaveSerializer](#) (void)
- [~GameSaveSerializer](#) (void)
- void [setSlot](#) (const int32_t slot)
Sets the slot number to save/load in.
- bool [load](#) (int &levelID, int &sceneID, [Player::Inventory](#) &inventory)
Loads a save file.
- void [save](#) (const int &levelID, const int &sceneID, const [Player::Inventory](#) &inventory)
Saves a file.
- void [saveInThread](#) (const int levelID, const int sceneID, const [Player::Inventory](#) inventory)
Saves a file in a separate thread.

Private Attributes

- std::mutex [fileLock](#)
- int32_t [slot](#)

6.10.1 Detailed Description

Provides multithreaded save, single-thread load of save files.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `teamusa::GameSaveSerializer::GameSaveSerializer (void)`

6.10.2.2 `teamusa::GameSaveSerializer::~~GameSaveSerializer (void)`

6.10.3 Member Function Documentation

6.10.3.1 `bool teamusa::GameSaveSerializer::load (int & levelID, int & sceneID, Player::Inventory & inventory)`

Loads a save file.

Returns

True if save file was loaded successfully, false if it doesn't exist.

6.10.3.2 `void teamusa::GameSaveSerializer::save (const int & levelID, const int & sceneID, const Player::Inventory & inventory)`

Saves a file.

6.10.3.3 `void teamusa::GameSaveSerializer::saveInThread (const int levelID, const int sceneID, const Player::Inventory & inventory)`

Saves a file in a separate thread.

6.10.3.4 `void teamusa::GameSaveSerializer::setSlot (const int32_t slot)`

Sets the slot number to save/load in.

6.10.4 Member Data Documentation

6.10.4.1 `std::mutex teamusa::GameSaveSerializer::fileLock` [private]

6.10.4.2 `int32_t teamusa::GameSaveSerializer::slot` [private]

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

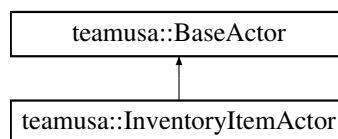
- [GameSaveSerializer.h](#)
- [GameSaveSerializer.cpp](#)

6.11 teamusa::InventoryItemActor Class Reference

InventoryItemActor creates a collectible item in the game environment.

```
#include <InventoryItemActor.h>
```

Inheritance diagram for teamusa::InventoryItemActor:

**Public Member Functions**

- `InventoryItemActor (Region region, const int itemID=-1, const int textureID=-1, const int layer=-1)`
- virtual `~InventoryItemActor (void)` override
- virtual const `ActorEvent onHover (Player &player)` override
Generates an ActorEvent if the player hovers over the actors' region.
- virtual const `ActorEvent onClick (Player &player)` override
Generates an ActorEvent if the player clicks in the actor's region.
- virtual const `ActorEvent step (Player &player)` override
Advances the actor one frame and sends the appropriate ActorEvent.

Private Attributes

- int [itemID](#)
- bool [pickedUp](#) = false

Additional Inherited Members

6.11.1 Detailed Description

InventoryItemActor creates a collectible item in the game environment.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 `InventoryItemActor::InventoryItemActor (Region region, const int itemID = -1, const int textureID = -1, const int layer = -1)` `[explicit]`

6.11.2.2 `InventoryItemActor::~InventoryItemActor (void)` `[override]`, `[virtual]`

6.11.3 Member Function Documentation

6.11.3.1 `const ActorEvent InventoryItemActor::onClick (Player & player)` `[override]`, `[virtual]`

Generates an [ActorEvent](#) if the player clicks in the actor's region.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.11.3.2 `const ActorEvent InventoryItemActor::onHover (Player & player)` `[override]`, `[virtual]`

Generates an [ActorEvent](#) if the player hovers over the actors' region.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.11.3.3 `const ActorEvent InventoryItemActor::step (Player & player)` `[override]`, `[virtual]`

Advances the actor one frame and sends the appropriate [ActorEvent](#).

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Implements [teamusa::BaseActor](#).

6.11.4 Member Data Documentation

6.11.4.1 `int teamusa::InventoryItemActor::itemID` [private]

6.11.4.2 `bool teamusa::InventoryItemActor::pickedUp = false` [private]

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

- [InventoryItemActor.h](#)
- [InventoryItemActor.cpp](#)

6.12 teamusa::Level Class Reference

A [Level](#) is a container of Scenes and Actors corresponding to those scenes.

```
#include <Level.h>
```

Classes

- class [Scene](#)

A scene is a collection of images (Actors) that is displayed on the screen.

Public Member Functions

- [Level](#) (void)
- [Level](#) (int levelID, [AudioEngine](#) &audioEngine, [VideoEngine](#) &videoEngine)
- const [ActorList](#) & [getActors](#) (void) const
Returns the list of actors in the current scene.
- const int [getBGImageID](#) (void) const
Returns the textureID of the background image in the current scene.
- const int [loadLevel](#) (const std::string &path, [AudioEngine](#) &audioEngine, [VideoEngine](#) &videoEngine)
Parses the specified level file, loads textures, audio samples, and stores the actors in a hash table.
- void [changeScene](#) (const int sceneID)
Changes the currently active scene.
- const int [getScene](#) ()
Returns the index of the currently active scene.
- void [clearAll](#) (void)
Removes all loaded scenes and actors from memory.

Private Member Functions

- [BaseActorPtr parseAudioStreamActor](#) (std::fstream &fs)
- [BaseActorPtr parseDelayedAudioActor](#) (std::fstream &fs)
- [BaseActorPtr parseDelayedVideoActor](#) (std::fstream &fs)
- [BaseActorPtr parseInventoryItemActor](#) (std::fstream &fs)
- [BaseActorPtr parseLevelLink](#) (std::fstream &fs)
- [BaseActorPtr parseMovingActor](#) (std::fstream &fs)
- [BaseActorPtr parseResponsiveAudioActor](#) (std::fstream &fs)
- [BaseActorPtr parseResponsiveVideoActor](#) (std::fstream &fs)
- [BaseActorPtr parseSceneLink](#) (std::fstream &fs)
- [BaseActorPtr parseTextboxSpawnActor](#) (std::fstream &fs)
- [BaseActorPtr parseVideoActor](#) (std::fstream &fs)
- [BaseActorPtr parseVideoEventActor](#) (std::fstream &fs)

Private Attributes

- std::unordered_map< int, [Scene](#) > [scenes](#)
- int [startScene](#)
- int [activeScene](#)

6.12.1 Detailed Description

A [Level](#) is a container of Scenes and Actors corresponding to those scenes.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 `Level::Level (void)`

6.12.2.2 `Level::Level (int levelID, AudioEngine & audioEngine, VideoEngine & videoEngine)`

6.12.3 Member Function Documentation

6.12.3.1 `void Level::changeScene (const int sceneID)`

Changes the currently active scene.

Subsequent calls to [getActors\(\)](#) will return the actors in that scene.

Parameters

<i>sceneID</i>	The ID of the new scene.
----------------	--------------------------

6.12.3.2 `void Level::clearAll (void)`

Removes all loaded scenes and actors from memory.

6.12.3.3 `const ActorList & Level::getActors (void) const`

Returns the list of actors in the current scene.

6.12.3.4 `const int Level::getBGImageID (void) const`

Returns the textureID of the background image in the current scene.

6.12.3.5 `const int Level::getScene ()`

Returns the index of the currently active scene.

6.12.3.6 `const int Level::loadLevel (const std::string & path, AudioEngine & audioEngine, VideoEngine & videoEngine)`

Parses the specified level file, loads textures, audio samples, and stores the actors in a hash table.

Parameters

<i>path</i>	The file path to the .lvl file.
<i>audioEngine</i>	A reference to the audio engine being used.
<i>videoEngine</i>	A reference to the video engine being used.

6.12.3.7 `BaseActorPtr Level::parseAudioStreamActor (std::fstream & fs) [private]`

6.12.3.8 `BaseActorPtr Level::parseDelayedAudioActor (std::fstream & fs) [private]`

6.12.3.9 `BaseActorPtr Level::parseDelayedVideoActor (std::fstream & fs) [private]`

6.12.3.10 `BaseActorPtr Level::parseInventoryItemActor (std::fstream & fs) [private]`

6.12.3.11 `BaseActorPtr Level::parseLevelLink (std::fstream & fs) [private]`

6.12.3.12 `BaseActorPtr Level::parseMovingActor (std::fstream & fs) [private]`

6.12.3.13 `BaseActorPtr Level::parseResponsiveAudioActor (std::fstream & fs) [private]`

6.12.3.14 `BaseActorPtr Level::parseResponsiveVideoActor (std::fstream & fs) [private]`

6.12.3.15 `BaseActorPtr Level::parseSceneLink (std::fstream & fs) [private]`

6.12.3.16 `BaseActorPtr Level::parseTextboxSpawnActor (std::fstream & fs) [private]`

6.12.3.17 `BaseActorPtr Level::parseVideoActor (std::fstream & fs) [private]`

6.12.3.18 `BaseActorPtr Level::parseVideoEventActor (std::fstream & fs) [private]`

6.12.4 Member Data Documentation

6.12.4.1 `int teamusa::Level::activeScene [private]`

6.12.4.2 `std::unordered_map<int, Scene> teamusa::Level::scenes [private]`

6.12.4.3 `int teamusa::Level::startScene [private]`

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

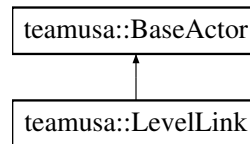
- [Level.h](#)
- [Level.cpp](#)

6.13 teamusa::LevelLink Class Reference

Allows the player to transition between levels.

```
#include <LevelLink.h>
```

Inheritance diagram for teamusa::LevelLink:



Public Member Functions

- [LevelLink](#) ([Region](#) region, const int Level_ID, const int [sceneID](#), const std::string itemRequired_Text, const int item_ID=-1)
- virtual [~LevelLink](#) (void) override
- virtual const [ActorEvent](#) [onClick](#) ([Player](#) &player) override
Returns an actor event when the actor's region is clicked on.
- virtual const [ActorEvent](#) [onHover](#) ([Player](#) &player) override
Returns an actor event when the actor's region is hovered over.
- virtual const [ActorEvent](#) [step](#) ([Player](#) &player) override
Advances the actor one frame.
- const int [getSceneID](#) (void) const
Gets the appropriate SceneID.
- virtual const std::string [getText](#) ()
Generates text when the player attempts to traverse a scene without a required item.

Private Attributes

- int [sceneID](#)
- int [levelID](#)
- std::string [itemRequiredText](#)
- int [requiredItemID](#) = -1

Additional Inherited Members

6.13.1 Detailed Description

Allows the player to transition between levels.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 `LevelLink::LevelLink (Region region, const int Level_ID, const int sceneID, const std::string itemRequired_Text, const int item_ID = -1) [explicit]`

6.13.2.2 `LevelLink::~~LevelLink (void) [override],[virtual]`

6.13.3 Member Function Documentation

6.13.3.1 `const int LevelLink::getSceneID (void) const`

Gets the appropriate SceneID.

Returns

Returns an integer representing the scene ID.

6.13.3.2 `const std::string LevelLink::getText () [virtual]`

Generates text when the player attempts to traverse a scene without a required item.

6.13.3.3 `const ActorEvent LevelLink::onClick (Player & player) [override],[virtual]`

Returns an actor event when the actor's region is clicked on.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.13.3.4 `const ActorEvent LevelLink::onHover (Player & player) [override],[virtual]`

Returns an actor event when the actor's region is hovered over.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.13.3.5 `const ActorEvent LevelLink::step (Player & player) [override],[virtual]`

Advances the actor one frame.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Implements [teamusa::BaseActor](#).

6.13.4 Member Data Documentation

6.13.4.1 `std::string teamusa::LevelLink::itemRequiredText` [private]

6.13.4.2 `int teamusa::LevelLink::levelID` [private]

6.13.4.3 `int teamusa::LevelLink::requiredItemID = -1` [private]

6.13.4.4 `int teamusa::LevelLink::sceneID` [private]

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

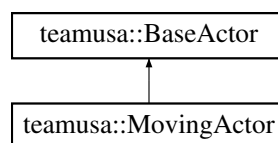
- [LevelLink.h](#)
- [LevelLink.cpp](#)

6.14 teamusa::MovingActor Class Reference

Will transition from one region to the next by calculating the distance to move each frame for a set number of frames.

```
#include <MovingActor.h>
```

Inheritance diagram for teamusa::MovingActor:



Public Member Functions

- [MovingActor](#) ([Region](#) startRegion, [Region](#) endregion, int textureId, int layer, int transitionsteps, bool move↔ OnSpawn)
- virtual [~MovingActor](#) (void) override
- virtual const [ActorEvent](#) [onClick](#) ([Player](#) &player) override
Generates an [ActorEvent](#) when the actor's region is clicked.
- virtual const [ActorEvent](#) [onHover](#) ([Player](#) &player) override
Generates an [ActorEvent](#) when the actor's region is hovered over.
- virtual const [ActorEvent](#) [step](#) ([Player](#) &player)
Advances the actor one frame.

Private Attributes

- [Region](#) endRegion
- int [transitionSteps](#) = 1
- int [currentStep](#) = 0
- int [xSpeed](#) = 0
- int [ySpeed](#) = 0
- int [hGrowth](#) = 0
- int [wGrowth](#) = 0
- bool [isActive](#) = false

Additional Inherited Members

6.14.1 Detailed Description

Will transition from one region to the next by calculating the distance to move each frame for a set number of frames. This allows for movement across the X and Y axis as well as scaling of the size of textures.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 `MovingActor::MovingActor (Region startRegion, Region endregion, int textureId, int layer, int transitionsteps, bool moveOnSpawn)` `[explicit]`

6.14.2.2 `MovingActor::~MovingActor (void)` `[override],[virtual]`

6.14.3 Member Function Documentation

6.14.3.1 `const ActorEvent MovingActor::onClick (Player & player)` `[override],[virtual]`

Generates an [ActorEvent](#) when the actor's region is clicked.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.14.3.2 `const ActorEvent MovingActor::onHover (Player & player)` `[override],[virtual]`

Generates an [ActorEvent](#) when the actor's region is hovered over.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.14.3.3 `const ActorEvent MovingActor::step (Player & player)` `[virtual]`

Advances the actor one frame.

Parameters

Player	The player
------------------------	------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Implements [teamusa::BaseActor](#).

6.14.4 Member Data Documentation

- 6.14.4.1 `int teamusa::MovingActor::currentStep = 0` `[private]`
- 6.14.4.2 `Region teamusa::MovingActor::endRegion` `[private]`
- 6.14.4.3 `int teamusa::MovingActor::hGrowth = 0` `[private]`
- 6.14.4.4 `bool teamusa::MovingActor::isActive = false` `[private]`
- 6.14.4.5 `int teamusa::MovingActor::transitionSteps = 1` `[private]`
- 6.14.4.6 `int teamusa::MovingActor::wGrowth = 0` `[private]`
- 6.14.4.7 `int teamusa::MovingActor::xSpeed = 0` `[private]`
- 6.14.4.8 `int teamusa::MovingActor::ySpeed = 0` `[private]`

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

- [MovingActor.h](#)
- [MovingActor.cpp](#)

6.15 teamusa::Player Class Reference

Handles all data relevant to the player engaging the game.

```
#include <Player.h>
```

Public Types

- typedef std::vector< int32_t > [Inventory](#)
Player inventory - an array of integer IDs.

Public Member Functions

- [Player](#) (void)
- [~Player](#) (void)
- const bool [hasItem](#) (const int32_t itemType) const
Tests if the player has an item in their inventory.
- void [addItem](#) (const int32_t itemType)
Inserts an item into the player's inventory.
- void [setCursor](#) (const [CursorStyle](#) style)
Sets the visual style of the player's mouse cursor.
- const int [getCursorTextureID](#) (void) const
Returns the current cursor texture ID associated with the cursor style.
- void [setPosition](#) (const int32_t x, const int32_t y)
Sets the position of the player's cursor.
- void [setPosition](#) (const [Point](#) &position)
Sets the position of the player's cursor.
- const [Point](#) [getPosition](#) (void) const
Gets the player's cursor position.

- const [Inventory](#) & [getInventory](#) () const
Returns the player's inventory.
- void [setInventory](#) (const [Inventory](#) &inventory)
Clears the player's current inventory and assigns the new one.

Static Public Attributes

- static const int [FLASHLIGHT_ID](#) = 1666
- static const int [CURSOR_DEFAULT_ID](#) = 1667
- static const int [CURSOR_SELECT_ID](#) = 1668
- static const int [CURSOR_UP_ID](#) = 1669
- static const int [CURSOR_DOWN_ID](#) = 1670
- static const int [CURSOR_LEFT_ID](#) = 1671
- static const int [CURSOR_RIGHT_ID](#) = 1672
- static const int [MOUSE_CLICK_ID](#) = 1700

Private Attributes

- [Region](#) mRegion
- int32_t mLayer
- int32_t mTextureID
- [Point](#) mPosition
- [Inventory](#) mInventory
- [CursorStyle](#) mCursorStyle

6.15.1 Detailed Description

Handles all data relevant to the player engaging the game.

6.15.2 Member Typedef Documentation

6.15.2.1 typedef std::vector<int32_t> teamusa::Player::Inventory

[Player](#) inventory - an array of integer IDs.

6.15.3 Constructor & Destructor Documentation

6.15.3.1 Player::Player (void) [explicit]

6.15.3.2 Player::~~Player (void)

6.15.4 Member Function Documentation

6.15.4.1 void Player::addItem (const int32_t itemType)

Inserts an item into the player's inventory.

Parameters

<i>itemType</i>	The item identifier to insert.
-----------------	--------------------------------

6.15.4.2 `const int Player::getCursorTextureID (void) const`

Returns the current cursor texture ID associated with the cursor style.

6.15.4.3 `const Player::Inventory & Player::getInventory () const`

Returns the player's inventory.

6.15.4.4 `const Point Player::getPosition (void) const`

Gets the player's cursor position.

Returns

A [Point](#) struct containing the cursor position.

6.15.4.5 `const bool Player::hasItem (const int32_t itemType) const`

Tests if the player has an item in their inventory.

Parameters

<i>itemType</i>	The item type identifier.
-----------------	---------------------------

Returns

True if the player has the item.

6.15.4.6 `void Player::setCursor (const CursorStyle style)`

Sets the visual style of the player's mouse cursor.

Parameters

<i>style</i>	The style type for the cursor.
--------------	--------------------------------

6.15.4.7 `void Player::setInventory (const Inventory & inventory)`

Clears the player's current inventory and assigns the new one.

Parameters

<i>inventory</i>	The inventory to assign to the player.
------------------	--

6.15.4.8 `void Player::setPosition (const int32_t x, const int32_t y)`

Sets the position of the player's cursor.

Parameters

<i>x</i>	The x-coordinate of the cursor.
<i>y</i>	The y-coordinate of the cursor.

6.15.4.9 void Player::setPosition (const Point & *position*)

Sets the position of the player's cursor.

Parameters

<i>position</i>	A Point struct containing the cursor position.
-----------------	--

6.15.5 Member Data Documentation

6.15.5.1 const int Player::CURSOR_DEFAULT_ID = 1667 [static]

6.15.5.2 const int Player::CURSOR_DOWN_ID = 1670 [static]

6.15.5.3 const int Player::CURSOR_LEFT_ID = 1671 [static]

6.15.5.4 const int Player::CURSOR_RIGHT_ID = 1672 [static]

6.15.5.5 const int Player::CURSOR_SELECT_ID = 1668 [static]

6.15.5.6 const int Player::CURSOR_UP_ID = 1669 [static]

6.15.5.7 const int Player::FLASHLIGHT_ID = 1666 [static]

6.15.5.8 CursorStyle teamusa::Player::mCursorStyle [private]

6.15.5.9 Inventory teamusa::Player::mInventory [private]

6.15.5.10 int32_t teamusa::Player::mLayer [private]

6.15.5.11 const int Player::MOUSE_CLICK_ID = 1700 [static]

6.15.5.12 Point teamusa::Player::mPosition [private]

6.15.5.13 Region teamusa::Player::mRegion [private]

6.15.5.14 int32_t teamusa::Player::mTextureID [private]

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

- [Player.h](#)
- [Player.cpp](#)

6.16 teamusa::Point Class Reference

An (x,y) coordinate within the rendering window.

```
#include <Point.h>
```


Public Member Functions

- [Point](#) (void)
- [Point](#) (int32_t x, const int32_t y)

Public Attributes

- int32_t x
- int32_t y

6.16.1 Detailed Description

An (x,y) coordinate within the rendering window.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 teamusa::Point::Point (void) `[inline]`

6.16.2.2 teamusa::Point::Point (int32_t x, const int32_t y) `[inline]`

6.16.3 Member Data Documentation

6.16.3.1 int32_t teamusa::Point::x

6.16.3.2 int32_t teamusa::Point::y

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

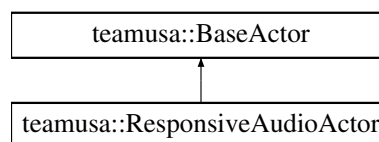
- [Point.h](#)

6.17 teamusa::ResponsiveAudioActor Class Reference

Will increment the value of stepCount until it is equal to durationSteps for each call to the step method.

```
#include <ResponsiveAudioActor.h>
```

Inheritance diagram for teamusa::ResponsiveAudioActor:



Public Member Functions

- [ResponsiveAudioActor](#) ([Region](#) region, int [hoverAudioId](#), int [clickAudioId](#))
- virtual [~ResponsiveAudioActor](#) (void) override
- virtual const [ActorEvent](#) [onClick](#) ([Player](#) &player) override
Generates an [ActorEvent](#) when the actor's region is clicked.
- virtual const [ActorEvent](#) [onHover](#) ([Player](#) &player) override
Generates an [ActorEvent](#) when the actor's region is hovered over.

- virtual const [ActorEvent step](#) ([Player](#) &player) override
Advances the actor one frame.

Private Attributes

- int [hoverAudioId](#)
- int [clickAudioId](#)

Additional Inherited Members

6.17.1 Detailed Description

Will increment the value of stepCount until it is equal to durationSteps for each call to the step method.

A call to onClick or onHover will set the value of stepCount to zero and emit an AudioID and value if stepCount is equal to durationSteps. The hoverAudioID or clickAudioID can be set to an invalid AudioID value to prevent sound from being played.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 **ResponsiveAudioActor::ResponsiveAudioActor** ([Region](#) *region*, int *hoverAudioId* = -1, int *clickAudioId* = -1)
[explicit]

6.17.2.2 **ResponsiveAudioActor::~~ResponsiveAudioActor** (void) [override],[virtual]

6.17.3 Member Function Documentation

6.17.3.1 **const ActorEvent ResponsiveAudioActor::onClick** ([Player](#) & *player*) [override],[virtual]

Generates an [ActorEvent](#) when the actor's region is clicked.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.17.3.2 **const ActorEvent ResponsiveAudioActor::onHover** ([Player](#) & *player*) [override],[virtual]

Generates an [ActorEvent](#) when the actor's region is hovered over.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.17.3.3 `const ActorEvent ResponsiveAudioActor::step (Player & player)` `[override],[virtual]`

Advances the actor one frame.

Parameters

Player	The player
------------------------	------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Implements [teamusa::BaseActor](#).

6.17.4 Member Data Documentation

6.17.4.1 `int teamusa::ResponsiveAudioActor::clickAudioid` [private]

6.17.4.2 `int teamusa::ResponsiveAudioActor::hoverAudioid` [private]

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

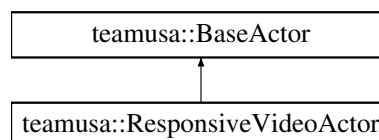
- [ResponsiveAudioActor.h](#)
- [ResponsiveAudioActor.cpp](#)

6.18 teamusa::ResponsiveVideoActor Class Reference

Changes its texture ID based on hovering and clicks.

```
#include <ResponsiveVideoActor.h>
```

Inheritance diagram for teamusa::ResponsiveVideoActor:



Public Member Functions

- [ResponsiveVideoActor](#) ([Region](#) region, int hoverTextureId, int clickTextureID, int defaultTextureID, int layer)
- virtual `~ResponsiveVideoActor` (void) override
- virtual const [ActorEvent](#) `onClick` ([Player](#) &player) override
Generates an [ActorEvent](#) when the actor's region is clicked.
- virtual const [ActorEvent](#) `onHover` ([Player](#) &player) override
Generates an [ActorEvent](#) when the actor's region is hovered over.
- virtual const [ActorEvent](#) `step` ([Player](#) &player) override
Advances the actor one frame.
- void `setTextureId` (int TextureId)
Sets the requested texture ID.

Private Attributes

- int `hoverTexture`
- int `clickTexture`
- int `defaultTextureId`

Additional Inherited Members

6.18.1 Detailed Description

Changes its texture ID based on hovering and clicks.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 ResponsiveVideoActor::ResponsiveVideoActor (*Region* *region*, int *hoverTextureId*, int *clickTextureId*, int *defaultTextureId*, int *layer*) `[explicit]`

6.18.2.2 ResponsiveVideoActor::~ResponsiveVideoActor (void) `[override], [virtual]`

6.18.3 Member Function Documentation

6.18.3.1 const ActorEvent ResponsiveVideoActor::onClick (*Player & player*) `[override], [virtual]`

Generates an [ActorEvent](#) when the actor's region is clicked.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.18.3.2 const ActorEvent ResponsiveVideoActor::onHover (*Player & player*) `[override], [virtual]`

Generates an [ActorEvent](#) when the actor's region is hovered over.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action

Reimplemented from [teamusa::BaseActor](#).

6.18.3.3 void ResponsiveVideoActor::setTextureId (int *TextureId*)

Sets the requested texture ID.

Parameters

<i>TextureID</i>	The integer ID of the requested texture.
------------------	--

6.18.3.4 const ActorEvent ResponsiveVideoActor::step (*Player & player*) `[override], [virtual]`

Advances the actor one frame.

Parameters

Player	The player
------------------------	------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Implements [teamusa::BaseActor](#).

6.18.4 Member Data Documentation

6.18.4.1 `int teamusa::ResponsiveVideoActor::clickTexture` [private]

6.18.4.2 `int teamusa::ResponsiveVideoActor::defaultTextureId` [private]

6.18.4.3 `int teamusa::ResponsiveVideoActor::hoverTexture` [private]

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

- [ResponsiveVideoActor.h](#)
- [ResponsiveVideoActor.cpp](#)

6.19 teamusa::Level::Scene Class Reference

A scene is a collection of images (Actors) that is displayed on the screen.

Public Attributes

- [ActorList](#) `actors`
- `int` [bgImageID](#)

6.19.1 Detailed Description

A scene is a collection of images (Actors) that is displayed on the screen.

6.19.2 Member Data Documentation

6.19.2.1 `ActorList` `teamusa::Level::Scene::actors`

6.19.2.2 `int` `teamusa::Level::Scene::bgImageID`

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

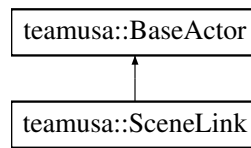
- [Level.h](#)

6.20 teamusa::SceneLink Class Reference

Allows the player to transition between scenes.

```
#include <SceneLink.h>
```

Inheritance diagram for teamusa::SceneLink:



Public Member Functions

- [SceneLink](#) ([Region](#) region, const int scene_ID, const std::string &itemRequired_Text, const int item_ID=-1)
- virtual [~SceneLink](#) (void) override
- virtual const [ActorEvent](#) [onClick](#) ([Player](#) &player) override
Generates an [ActorEvent](#) when the actor's region is clicked.
- virtual const [ActorEvent](#) [onHover](#) ([Player](#) &player) override
Generates an [ActorEvent](#) when the actor's region is hovered over.
- virtual const [ActorEvent](#) [step](#) ([Player](#) &player) override
Advances the actor one frame.
- virtual const std::string [getText](#) ()
Displays the appropriate text when a player attempts to traverse a scene without the required item.

Private Attributes

- int [sceneID](#)
- std::string [itemRequiredText](#)
- int [requiredItemID](#)
- [CursorStyle](#) [cursorStyle](#)

Additional Inherited Members

6.20.1 Detailed Description

Allows the player to transition between scenes.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 `SceneLink::SceneLink (Region region, const int scene_ID, const std::string & itemRequired_Text, const int item_ID = -1) [explicit]`

6.20.2.2 `SceneLink::~SceneLink (void) [override],[virtual]`

6.20.3 Member Function Documentation

6.20.3.1 `const std::string SceneLink::getText () [virtual]`

Displays the appropriate text when a player attempts to traverse a scene without the required item.

6.20.3.2 `const ActorEvent SceneLink::onClick (Player & player) [override],[virtual]`

Generates an [ActorEvent](#) when the actor's region is clicked.

Parameters

<i>Player</i>	The player.
-------------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.20.3.3 `const ActorEvent SceneLink::onHover (Player & player) [override],[virtual]`

Generates an [ActorEvent](#) when the actor's region is hovered over.

Parameters

<i>Player</i>	The player.
-------------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.20.3.4 `const ActorEvent SceneLink::step (Player & player) [override],[virtual]`

Advances the actor one frame.

Parameters

<i>Player</i>	The player
-------------------------------	------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Implements [teamusa::BaseActor](#).

6.20.4 Member Data Documentation

6.20.4.1 `CursorStyle teamusa::SceneLink::cursorStyle [private]`

6.20.4.2 `std::string teamusa::SceneLink::itemRequiredText [private]`

6.20.4.3 `int teamusa::SceneLink::requiredItemID [private]`

6.20.4.4 `int teamusa::SceneLink::sceneID [private]`

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

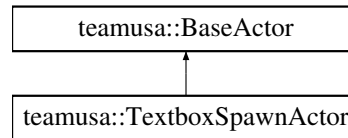
- [SceneLink.h](#)
- [SceneLink.cpp](#)

6.21 teamusa::TextboxSpawnActor Class Reference

Will emit a DisplayText event when the onClick method is called.

```
#include <TextboxSpawnActor.h>
```

Inheritance diagram for teamusa::TextboxSpawnActor:



Public Member Functions

- [TextboxSpawnActor](#) ([Region](#) region, std::string [text](#))
- virtual [~TextboxSpawnActor](#) (void)
- virtual const [ActorEvent](#) [onClick](#) ([Player](#) &player)
Generates an [ActorEvent](#) when the actor's region is clicked.
- virtual const [ActorEvent](#) [step](#) ([Player](#) &player)
Generates an [ActorEvent](#) when the actor's region is hovered over.
- std::string [getText](#) (void)
Retrieves the text for the textbox from the level file.

Private Attributes

- std::string [text](#)
- bool [activated](#)

Additional Inherited Members

6.21.1 Detailed Description

Will emit a DisplayText event when the onClick method is called.

The actor can then have its text accessed by the engine for display through a call to the getText method.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 [TextboxSpawnActor::TextboxSpawnActor](#) ([Region](#) *region*, std::string *text*) [explicit]

6.21.2.2 [TextboxSpawnActor::~~TextboxSpawnActor](#) (void) [virtual]

6.21.3 Member Function Documentation

6.21.3.1 std::string [TextboxSpawnActor::getText](#) (void)

Retrieves the text for the textbox from the level file.

6.21.3.2 const [ActorEvent](#) [TextboxSpawnActor::onClick](#) ([Player](#) & *player*) [virtual]

Generates an [ActorEvent](#) when the actor's region is clicked.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.21.3.3 `const ActorEvent TextboxSpawnActor::step (Player & player)` [virtual]

Generates an [ActorEvent](#) when the actor's region is hovered over.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Implements [teamusa::BaseActor](#).

6.21.4 Member Data Documentation

6.21.4.1 `bool teamusa::TextboxSpawnActor::activated` [private]

6.21.4.2 `std::string teamusa::TextboxSpawnActor::text` [private]

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

- [TextboxSpawnActor.h](#)
- [TextboxSpawnActor.cpp](#)

6.22 teamusa::Timer Class Reference

A timer that counts up from zero in milliseconds.

```
#include <Timer.h>
```

Public Member Functions

- [Timer](#) (void)
- [~Timer](#) (void)
- `const uint32_t start` (void)
Starts the timer.
- `void stop` (void)
Stops the timer.
- `void pause` (void)
Pauses the timer.
- `void unpause` (void)
Unpauses the timer.
- `const uint32_t getTicks` (void) const
Gets the time in milliseconds since the timer was started.

Private Attributes

- uint32_t [mStartTicks](#)
- uint32_t [mPauseTicks](#)
- bool [mPaused](#)
- bool [mStarted](#)

6.22.1 Detailed Description

A timer that counts up from zero in milliseconds.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 `Timer::Timer (void) [explicit]`

6.22.2.2 `Timer::~~Timer (void)`

6.22.3 Member Function Documentation

6.22.3.1 `const uint32_t Timer::getTicks (void) const`

Gets the time in milliseconds since the timer was started.

Returns

The elapsed time.

6.22.3.2 `void Timer::pause (void)`

Pauses the timer.

6.22.3.3 `const uint32_t Timer::start (void)`

Starts the timer.

6.22.3.4 `void Timer::stop (void)`

Stops the timer.

6.22.3.5 `void Timer::unpause (void)`

Unpauses the timer.

6.22.4 Member Data Documentation

6.22.4.1 `bool teamusa::Timer::mPaused [private]`

6.22.4.2 `uint32_t teamusa::Timer::mPauseTicks [private]`

6.22.4.3 `bool teamusa::Timer::mStarted [private]`

6.22.4.4 uint32_t teamusa::Timer::mStartTicks [private]

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

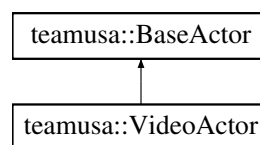
- [Timer.h](#)
- [Timer.cpp](#)

6.23 teamusa::VideoActor Class Reference

Displays a texture in a region and performs no other behavior.

```
#include <VideoActor.h>
```

Inheritance diagram for teamusa::VideoActor:



Public Member Functions

- [VideoActor](#) ([Region](#) region, int textureId, int layer)
- virtual [~VideoActor](#) (void) override
- virtual const [ActorEvent](#) step ([Player](#) &player) override

Advances the actor one frame.

Additional Inherited Members

6.23.1 Detailed Description

Displays a texture in a region and performs no other behavior.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 [VideoActor::VideoActor](#) ([Region](#) region, int textureId = -1, int layer = 1) [explicit]

6.23.2.2 [VideoActor::~~VideoActor](#) (void) [override],[virtual]

6.23.3 Member Function Documentation

6.23.3.1 const [ActorEvent](#) [VideoActor::step](#) ([Player](#) &player) [override],[virtual]

Advances the actor one frame.

Parameters

Player	The player
------------------------	------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Implements [teamusa::BaseActor](#).

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

- [VideoActor.h](#)
- [VideoActor.cpp](#)

6.24 mediawrap::VideoContext Class Reference

Provides basic 2D rendering capabilities.

```
#include <VideoContext.hpp>
```

Public Types

- enum [Flip](#) { [FLIP_NONE](#) = SDL_FLIP_NONE, [FLIP_HORIZONTAL](#) = SDL_FLIP_HORIZONTAL, [FLIP_VERTICAL](#) = SDL_FLIP_VERTICAL }
- Used to designate how an image should be flipped across an axis.*
- enum [BlendMode](#) { [BLENDMODE_NONE](#) = SDL_BLENDMODE_NONE, [BLENDMODE_BLEND](#) = SDL_BLENDMODE_BLEND, [BLENDMODE_ADD](#) = SDL_BLENDMODE_ADD, [BLENDMODE_MOD](#) = SDL_BLENDMODE_MOD }
- Used to specify how a texture should behave when objects are rendered onto it.*
- enum [DebugColor](#) { [RED](#) = 0, [GREEN](#), [BLUE](#) }
- typedef [SDL_Rect](#) [Region](#)
- Used to specify x, y, width, height of an texture source or destination region.*
- typedef unsigned int [TextureID](#)
- Used to identify each texture uniquely.*
- typedef std::unordered_map< [TextureID](#), [SDL_Texture](#) * >::iterator [texture_iter](#)
- Used to access elements in the texture map.*

Public Member Functions

- [VideoContext](#) (const std::string &title, unsigned int width, unsigned int height)
- Constructs a new rendering context that includes a window and the renderer associated with it.*
- [~VideoContext](#) (void)
- Deletes the renderer and window associated with this context.*
- void [display](#) (void)
- Displays the rendered textures on screen.*
- [Region load_texture](#) ([TextureID](#) id, const std::string &image_path, [BlendMode](#) blend=[BLENDMODE_BLEND](#))
- Loads a texture from the filename into the specified texture id.*
- [Region create_texture](#) ([TextureID](#) id, int width, int height, [BlendMode](#) blend=[BLENDMODE_BLEND](#))
- Creates a blank texture, which should be filled completely or cleared before rendering to prevent old fragments from appearing.*
- void [delete_texture](#) ([TextureID](#) id)
- The deletes the given texture from this context.*
- void [render](#) ([TextureID](#) id, [Region](#) *dest, [Region](#) *src)
- Draws the given texture onto the canvas.*
- void [renderDebugBox](#) (const [Region](#) ®ion, const [DebugColor](#) color, const [TextureID](#) layer)
- void [render_onto](#) ([TextureID](#) dest_id, [TextureID](#) src_id, const [Region](#) *dest_region, [Region](#) *src_region)

Draws the given source texture onto the destination texture.

- void `render_rotate` (`TextureID` dest_id, `TextureID` src_id, `Region` *dest_region, `Region` *src_region, double angle=0.0, `Flip` flip=`FLIP_NONE`)

Draws the given source texture onto the destination texture after applying a rotate and flip operation.

- void `render_clear` ()

Clears the canvas with the default clear color.

- void `render_clear` (`TextureID` id)

Clears the given texture with the default clear color.

- void `fill_texture` (`TextureID` id, int r, int g, int b, int a)

Fills the given texture with the given rgba value.

- void `load_font` (const std::string &font_path, int font_size)

Loads the given font from the path specified.

- void `render_text` (`TextureID` dest_id, `Region` *dest_region, const std::string &text, Uint8 r, Uint8 g, Uint8 b, Uint8 a)

Renders the given text onto the the destination texture.

- void `swapFullscreen` (void)

Checks the state of the window and swaps to fullscreen or windowed mode.

Private Attributes

- std::unordered_map< `TextureID`, `SDL_Texture` * > * `textures`
- `VideoDisplay` * `video_display`
- `SDL_Renderer` * `renderer`
- `TTF_Font` * `font`

6.24.1 Detailed Description

Provides basic 2D rendering capabilities.

Acts as an abstraction layer to the SDL2 video library.

6.24.2 Member Typedef Documentation

6.24.2.1 typedef `SDL_Rect` `mediawrap::VideoContext::Region`

Used to specify x, y, width, height of an texture source or destination region.

6.24.2.2 typedef std::unordered_map<`TextureID`, `SDL_Texture`*>::iterator `mediawrap::VideoContext::texture_iter`

Used to access elements in the texture map.

6.24.2.3 typedef unsigned int `mediawrap::VideoContext::TextureID`

Used to identify each texture uniquely.

Each texture loaded is to be assigned a key of this type.

6.24.3 Member Enumeration Documentation

6.24.3.1 enum mediawrap::VideoContext::BlendMode

Used to specify how a texture should behave when objects are rendered onto it.

Enumerator

BLENDMODE_NONE

BLENDMODE_BLEND

BLENDMODE_ADD

BLENDMODE_MOD

6.24.3.2 enum mediawrap::VideoContext::DebugColor

Enumerator

RED

GREEN

BLUE

6.24.3.3 enum mediawrap::VideoContext::Flip

Used to designate how an image should be flipped across an axis.

These two values can be ORed together to achive both effects.

Enumerator

FLIP_NONE

FLIP_HORIZONTAL

FLIP_VERTICAL

6.24.4 Constructor & Destructor Documentation

6.24.4.1 VideoContext::VideoContext (const std::string & *title*, unsigned int *width*, unsigned int *height*)

Constructs a new rendering context that includes a window and the renderer associated with it.

Provides utilities for loading textures and storing them in an internal mapping.

Parameters

<i>title</i>	The title to display at the top of the window.
<i>width</i>	The width of the window created.
<i>height</i>	The height of the window created.

6.24.4.2 VideoContext::~~VideoContext (void)

Deletes the renderer and window associated with this context.

Also deletes all textures currently loaded by this context.

6.24.5 Member Function Documentation

6.24.5.1 VideoContext::Region VideoContext::create_texture (TextureID *id*, int *width*, int *height*, BlendMode *blend* = BLENDMODE_BLEND)

Creates a blank texture, which should be filled completely or cleared before rendering to prevent old fragments from appearing.

Must be deleted using delete_texture.

Parameters

<i>id</i>	The id to assign to this texture. If this id is already in use, it deletes the existing texture first before loading this new one.
<i>width</i>	The width of the new texture
<i>height</i>	The height of the next texture
<i>blend</i>	The blending mode which decides how to react with other textures. Defaults to BLENDMODE_BLEND.

Returns

The source region of the new texture created.

6.24.5.2 void VideoContext::delete_texture (TextureID *id*)

The deletes the given texture from this context.

Parameters

<i>id</i>	The id of the texture to delete.
-----------	----------------------------------

6.24.5.3 void VideoContext::display (void)

Displays the rendered textures on screen.

6.24.5.4 void VideoContext::fill_texture (TextureID *id*, int *r*, int *g*, int *b*, int *a*)

Fills the given texture with the given rgba value.

Parameters

<i>id</i>	The id of the texture to fill with the specified color.
<i>r</i>	The red value 0-255
<i>g</i>	The green value 0-255
<i>b</i>	The blue value 0-255
<i>a</i>	The alpha value 0-255

6.24.5.5 void VideoContext::load_font (const std::string & *font_path*, int *font_size*)

Loads the given font from the path specified.

Only one font may be loaded at any given time. Repeated calls to this function will delete the previous font before creating a new one.

Parameters

<i>font_path</i>	The path to the ttf file to load as a font.
<i>font_size</i>	The size of the font to load.

6.24.5.6 VideoContext::Region VideoContext::load_texture (TextureID *id*, const std::string & *image_path*, BlendMode *blend* = BLENDMODE_BLEND)

Loads a texture from the filename into the specified texture id.

Must be deleted using delete_texture.

Parameters

<i>id</i>	The id to assign to this texture. If this id is already in use, it deletes the existing texture first before loading this new one.
<i>image_path</i>	The path of the file to load as a texture.
<i>blend</i>	The blending mode which decides how to react with other textures. Defaults to BLENDMODE_BLEND.

Returns

The auto-detected source rectangle for this image.

6.24.5.7 void VideoContext::render (TextureID *id*, Region * *dest*, Region * *src*)

Draws the given texture onto the canvas.

Parameters

<i>id</i>	The id of the texture to draw onto the canvas.
<i>dest</i>	The destination region to draw onto the canvas.
<i>src</i>	The source region to copy from when drawing.

6.24.5.8 void VideoContext::render_clear ()

Clears the canvas with the default clear color.

6.24.5.9 void VideoContext::render_clear (TextureID *id*)

Clears the given texture with the default clear color.

Parameters

<i>id</i>	The id of the texture to clear.
-----------	---------------------------------

6.24.5.10 void VideoContext::render_onto (TextureID *dest_id*, TextureID *src_id*, const Region * *dest_region*, Region * *src_region*)

Draws the given source texture onto the destination texture.

Parameters

<i>dest_id</i>	The id of the texture that will act as a canvas and be drawn on.
<i>src_id</i>	The id of the texture to draw over the destination Texture.
<i>dest_region</i>	The region to draw the source texture into.
<i>src_region</i>	The region to copy the source texture from.

6.24.5.11 `void VideoContext::render_rotate (TextureID dest_id, TextureID src_id, Region * dest_region, Region * src_region, double angle = 0.0, Flip flip = FLIP_NONE)`

Draws the given source texture onto the destination texture after applying a rotate and flip operation.

Parameters

<i>dest_id</i>	The id of the texture that will act as a canvas and be drawn on.
<i>src_id</i>	The id of the texture to draw over the destination Texture.
<i>dest_region</i>	The region to draw the source texture into.
<i>src_region</i>	The region to copy the source texture from.
<i>angle</i>	The angle in degrees to rotate the source image. Defaults to zero.
<i>flip</i>	The direction to flip the source texture in. Defaults to none.

6.24.5.12 `void VideoContext::render_text (TextureID dest_id, Region * dest_region, const std::string & text, Uint8 r, Uint8 g, Uint8 b, Uint8 a)`

Renders the given text onto the the destination texture.

A successful call to `load_font` must be performed before this method should be called.

Parameters

<i>dest_id</i>	The destination texture to render onto.
<i>dest_region</i>	The region on the destination texture to render the font into.
<i>text</i>	The string to render.
<i>r</i>	The red value 0-255
<i>g</i>	The green value 0-255
<i>b</i>	The blue value 0-255
<i>a</i>	The alpha value 0-255

6.24.5.13 `void VideoContext::renderDebugBox (const Region & region, const DebugColor color, const TextureID layer)`

6.24.5.14 `void VideoContext::swapFullscreen (void)`

Checks the state of the window and swaps to fullscreen or windowed mode.

6.24.6 Member Data Documentation

6.24.6.1 `TTF_Font* mediawrap::VideoContext::font` [private]

6.24.6.2 `SDL_Renderer* mediawrap::VideoContext::renderer` [private]

6.24.6.3 `std::unordered_map<TextureID, SDL_Texture*>* mediawrap::VideoContext::textures` [private]

6.24.6.4 `VideoDisplay* mediawrap::VideoContext::video_display` [private]

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

- [VideoContext.hpp](#)
- [VideoContext.cpp](#)

6.25 mediawrap::VideoDisplay Class Reference

Creates a window and initializes SDL2 and SDL2_IMG.

```
#include <VideoDisplay.hpp>
```

Public Member Functions

- [VideoDisplay](#) (const std::string &title, unsigned int width, unsigned int height)
Attempts to init SDL2 and SDL2_IMG and create a window.
- [~VideoDisplay](#) (void)
Destroys the window and renderer.
- SDL_Renderer * [get_renderer](#) (void)
Creates a renderer attached to this window.
- void [swapFullscreen](#) (void)

Private Attributes

- SDL_Window * [window](#)

6.25.1 Detailed Description

Creates a window and initializes SDL2 and SDL2_IMG.

Must be destroyed after use.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 mediawrap::VideoDisplay::VideoDisplay (const std::string & title, unsigned int width, unsigned int height)

Attempts to init SDL2 and SDL2_IMG and create a window.

Throws runtime_error if unable to set up any of these.

Parameters

<i>title</i>	The title to display at the top of the window.
<i>width</i>	The width of the window created.
<i>height</i>	The height of the window created.

6.25.2.2 mediawrap::VideoDisplay::~~VideoDisplay (void)

Destroys the window and renderer.

Uninitializes SDL and SDL_Image.

6.25.3 Member Function Documentation

6.25.3.1 `SDL_Renderer * mediawrap::VideoDisplay::get_renderer (void)`

Creates a renderer attached to this window.

Must be deleted after use.

Returns

An SDL2 renderer for this window.

6.25.3.2 `void mediawrap::VideoDisplay::swapFullscreen (void)`

6.25.4 Member Data Documentation

6.25.4.1 `SDL_Window* mediawrap::VideoDisplay::window` [private]

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

- [VideoDisplay.hpp](#)
- [VideoDisplay.cpp](#)

6.26 teamusa::VideoEngine Class Reference

Provides video capabilities that are specific to Legend of the Great Unwashed.

```
#include <VideoEngine.hpp>
```

Public Member Functions

- [VideoEngine](#) (const std::string &title, unsigned int width, unsigned int height)
Creates a new window that provides basic 2D drawing capabilities.
- [~VideoEngine](#) ()
Destroys the video engine after freeing all associated textures.
- void [loadTexture](#) (const std::string &path, [TextureID](#) id, [ResourceGroup](#) group)
Loads the image file from the given path, transforms it into a surface, and pushes it onto the graphics card as a texture.
- void [render](#) (const [Region](#) ®ion, const unsigned int layer, const [TextureID](#) id)
Renders the texture onto the given layer in the given region.
- void [renderDebugBox](#) (const [Region](#) ®ion, const [VideoContext::DebugColor](#)=[VideoContext::DebugColor::BLUE](#))
- void [renderRotate](#) ([Region](#) ®ion, unsigned int layer, [TextureID](#) id, float angle=0.0)
Renders the texture onto the given layer in the given region with the given rotation angle.
- void [swapFullscreen](#) (void)
Calls [swapFullscreen\(\)](#) on VideoDisplay.
- bool [isShowingTextbox](#) ()
States whether a textbox is currently being displayed or not.
- void [showTextbox](#) (const std::string &text)
Displays the given text in a textbox.
- void [hideTextbox](#) ()
Clears the current textbox so it does not appear.
- void [deleteTexture](#) ([TextureID](#) id)

Removes the current texture from graphics memory.

- void [deleteResourceGroup](#) ([ResourceGroup](#) resourceGroup)

Deletes all textures associated with the given resource group.

- void [display](#) ()

Displays all rendered textures on screen.

Private Member Functions

- void [clearLayers](#) ()

Clears all layers with the default clear color.

Private Attributes

- bool [textboxActive](#)
- [TextureID](#) [layers](#) [[NUM_LAYERS](#)]
- std::vector< [TextureID](#) > [coreResources](#)
- std::vector< [TextureID](#) > [levelResources](#)
- [VideoContext](#) * [videoContext](#)
- [Region](#) [textboxPadding](#)
- [Region](#) [textboxRegion](#)

Static Private Attributes

- static const unsigned int [NUM_LAYERS](#) = 7
- static const unsigned int [SHADOW_LAYER](#) = 4
- static const [TextureID](#) [TEXT_LAYER](#) = 8
- static const [TextureID](#) [MAX_RESERVED_ID](#) = 1000

6.26.1 Detailed Description

Provides video capabilities that are specific to Legend of the Great Unwashed.

Utilizes VideoContext to perform rendering.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 VideoEngine::VideoEngine (const std::string & *title*, unsigned int *width*, unsigned int *height*)

Creates a new window that provides basic 2D drawing capabilities.

Parameters

<i>title</i>	The title to be displayed at the top of the window.
<i>width</i>	The width of the window in pixels.
<i>height</i>	The height of the window in pixels.

6.26.2.2 VideoEngine::~~VideoEngine ()

Destroys the video engine after freeing all associated textures.

6.26.3 Member Function Documentation

6.26.3.1 void VideoEngine::clearLayers () [private]

Clears all layers with the default clear color.

Does not modify the textbox layer.

6.26.3.2 void VideoEngine::deleteResourceGroup (ResourceGroup resourceGroup)

Deletes all textures associated with the given resource group.

Parameters

<i>resourceGroup</i>	The group of textures to delete from video memory.
----------------------	--

6.26.3.3 void VideoEngine::deleteTexture (TextureID id)

Removes the current texture from graphics memory.

Parameters

<i>id</i>	The id of the texture to delete.
-----------	----------------------------------

6.26.3.4 void VideoEngine::display (void)

Displays all rendered textures on screen.

6.26.3.5 void VideoEngine::hideTextbox ()

Clears the current textbox so it does not appear.

6.26.3.6 bool VideoEngine::isShowingTextbox ()

States whether a textbox is currently being displayed or not.

Returns

The status of the textbox.

6.26.3.7 void VideoEngine::loadTexture (const std::string & path, TextureID id, ResourceGroup group)

Loads the image file from the given path, transforms it into a surface, and pushes it onto the graphics card as a texture.

Parameters

<i>path</i>	The relative location of the image to load.
<i>id</i>	The id to assign to the loaded texture.
<i>resGroup</i>	The group to load the resource into.

6.26.3.8 void VideoEngine::render (const Region & region, const unsigned int layer, const TextureID id)

Renders the texture onto the given layer in the given region.

Parameters

<i>region</i>	The region to draw the texture into.
<i>layer</i>	The layer to render the image onto (0-6) are valid.
<i>id</i>	The id of the texture to draw.

6.26.3.9 `void VideoEngine::renderDebugBox (const Region & region, const VideoContext::DebugColor color = VideoContext::DebugColor::BLUE)`

6.26.3.10 `void VideoEngine::renderRotate (Region & region, unsigned int layer, TextureID id, float angle = 0.0)`

Renders the texture onto the given layer in the given region with the given rotation angle.

Parameters

<i>region</i>	The region to draw the texture into.
<i>layer</i>	The layer to render the image onto (0-6) are valid.
<i>id</i>	The id of the texture to draw.
<i>angle</i>	The angle in degrees to rotate the image. Defaults to 0.

6.26.3.11 `void VideoEngine::showTextbox (const std::string & text)`

Displays the given text in a textbox.

Parameters

<i>text</i>	The text to display on screen.
-------------	--------------------------------

6.26.3.12 `void VideoEngine::swapFullscreen (void)`

Calls [swapFullscreen\(\)](#) on VideoDisplay.

6.26.4 Member Data Documentation

6.26.4.1 `std::vector<TextureID> teamusa::VideoEngine::coreResources` [private]

6.26.4.2 `TextureID teamusa::VideoEngine::layers[NUM_LAYERS]` [private]

6.26.4.3 `std::vector<TextureID> teamusa::VideoEngine::levelResources` [private]

6.26.4.4 `const TextureID teamusa::VideoEngine::MAX_RESERVED_ID = 1000` [static], [private]

6.26.4.5 `const unsigned int teamusa::VideoEngine::NUM_LAYERS = 7` [static], [private]

6.26.4.6 `const unsigned int teamusa::VideoEngine::SHADOW_LAYER = 4` [static], [private]

6.26.4.7 `const TextureID teamusa::VideoEngine::TEXT_LAYER = 8` [static], [private]

6.26.4.8 `bool teamusa::VideoEngine::textboxActive` [private]

6.26.4.9 `Region teamusa::VideoEngine::textboxPadding` [private]

6.26.4.10 `Region teamusa::VideoEngine::textboxRegion` [private]

6.26.4.11 VideoContext* teamusa::VideoEngine::videoContext [private]

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

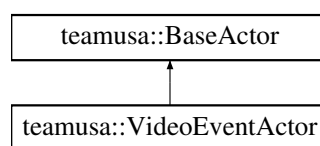
- [VideoEngine.hpp](#)
- [VideoEngine.cpp](#)

6.27 teamusa::VideoEventActor Class Reference

Will display a texture and perform no action until clicked.

```
#include <VideoEventActor.h>
```

Inheritance diagram for teamusa::VideoEventActor:



Public Member Functions

- [VideoEventActor](#) ([Region](#) region, int textureID, [ActorEventType](#) eventType, int eventValue, int layer)
- virtual [~VideoEventActor](#) (void) override
- virtual const [ActorEvent](#) [onClick](#) ([Player](#) &player) override
Generates an [ActorEvent](#) when the actor's region is clicked.
- virtual const [ActorEvent](#) [onHover](#) ([Player](#) &player) override
Generates an [ActorEvent](#) when the actor's region is hovered over.
- virtual const [ActorEvent](#) [step](#) ([Player](#) &player)
Advances the actor one frame.

Private Attributes

- [ActorEvent](#) actorEvent

Additional Inherited Members

6.27.1 Detailed Description

Will display a texture and perform no action until clicked.

The TextureID can be set to an invalid value during construction if no image needs to be displayed.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 `VideoEventActor::VideoEventActor (Region region, int textureID, ActorEventType eventType, int eventValue, int layer)` [explicit]

6.27.2.2 `VideoEventActor::~~VideoEventActor (void)` [override],[virtual]

6.27.3 Member Function Documentation

6.27.3.1 `const ActorEvent VideoEventActor::onClick (Player & player)` [override],[virtual]

Generates an [ActorEvent](#) when the actor's region is clicked.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.27.3.2 `const ActorEvent VideoEventActor::onHover (Player & player)` `[override], [virtual]`

Generates an [ActorEvent](#) when the actor's region is hovered over.

Parameters

Player	The player.
------------------------	-------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Reimplemented from [teamusa::BaseActor](#).

6.27.3.3 `const ActorEvent VideoEventActor::step (Player & player)` `[virtual]`

Advances the actor one frame.

Parameters

Player	The player
------------------------	------------

Returns

Returns an [ActorEvent](#) that triggers an actor to perform an action.

Implements [teamusa::BaseActor](#).

6.27.4 Member Data Documentation

6.27.4.1 `ActorEvent teamusa::VideoEventActor::actorEvent` `[private]`

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

- [VideoEventActor.h](#)
- [VideoEventActor.cpp](#)

Chapter 7

File Documentation

7.1 ActorEvent.h File Reference

Declares ActorEvent struct.

```
#include "Headers.h"
```

Classes

- class [teamusa::ActorEvent](#)
Event data generated by Actors, handled by [Engine](#).

Namespaces

- [teamusa](#)

Enumerations

- enum [teamusa::ActorEventType](#) {
[teamusa::Nil](#) = -1, [teamusa::ChangeScene](#), [teamusa::LoadLevel](#), [teamusa::PlayAudio](#),
[teamusa::NewGame](#), [teamusa::LoadGame](#), [teamusa::DisplayText](#), [teamusa::ExitGame](#),
[teamusa::StreamAudio](#) }
Events that actors can trigger.

7.1.1 Detailed Description

Declares ActorEvent struct.

7.2 Assert.h File Reference

Declares custom Assert macro.

Namespaces

- [teamusa](#)

Macros

- `#define Assert(exp) ;`

7.2.1 Detailed Description

Declares custom Assert macro.

7.2.2 Macro Definition Documentation

7.2.2.1 `#define Assert(exp);`

7.3 AudioEngine.cpp File Reference

Declares the AudioEngine class.

```
#include "AudioEngine.hpp"
```

7.3.1 Detailed Description

Declares the AudioEngine class.

7.4 AudioEngine.hpp File Reference

Declares the AudioEngine class.

```
#include <string>
#include <vector>
#include "AudioPlayer.hpp"
#include "Engine/ResourceGroup.hpp"
```

Classes

- class [teamusa::AudioEngine](#)
Provides project-specific audio functionality for Legend of the Great Unwashed.

Namespaces

- [teamusa](#)

Typedefs

- typedef [mediawrap::AudioPlayer::AudioID](#) [teamusa::AudioID](#)

7.4.1 Detailed Description

Declares the AudioEngine class.

7.5 AudioPlayer.cpp File Reference

Implements the AudioPlayer class.

```
#include "AudioPlayer.hpp"
```

7.5.1 Detailed Description

Implements the AudioPlayer class.

7.6 AudioPlayer.hpp File Reference

Declares the AudioPlayer class.

```
#include <stdexcept>
#include <string>
#include <unordered_map>
#include "SDL2/SDL.h"
#include "SDL2/SDL_mixer.h"
```

Classes

- class [mediawrap::AudioPlayer](#)
Provides basic audio playing capabilities with WAV files.

Namespaces

- [mediawrap](#)

7.6.1 Detailed Description

Declares the AudioPlayer class.

7.7 AudioStreamActor.cpp File Reference

Implements AudioStreamActor class.

```
#include "AudioStreamActor.h"
```

7.7.1 Detailed Description

Implements AudioStreamActor class.

7.8 AudioStreamActor.h File Reference

Declares AudioStreamActor class.

```
#include "BaseActor.h"
```

Classes

- class [teamusa::AudioStreamActor](#)

If this actor is not activated, it will emit a StreamAudio event and set its status to activated when the step method is called.

Namespaces

- [teamusa](#)

7.8.1 Detailed Description

Declares AudioStreamActor class.

7.9 BaseActor.cpp File Reference

Implements BaseActor class.

```
#include "BaseActor.h"  
#include "Engine/Assert.h"  
#include "Engine/Point.h"
```

7.9.1 Detailed Description

Implements BaseActor class.

7.10 BaseActor.h File Reference

Declares BaseActor class.

```
#include "ActorEvent.h"  
#include "Audio/AudioEngine.hpp"  
#include "Video/VideoEngine.hpp"
```

Classes

- class [teamusa::ActorVideo](#)

Contains data for rendering actor.

- class [teamusa::BaseActor](#)

Abstract class which all actors must derive from.

Namespaces

- [teamusa](#)

7.10.1 Detailed Description

Declares BaseActor class.

7.11 CursorStyle.h File Reference

Declares CursorStyle enumerations.

Namespaces

- [teamusa](#)

Enumerations

- enum [teamusa::CursorStyle](#) {
 [teamusa::CursorStyle::CURSOR_DEFAULT](#), [teamusa::CursorStyle::CURSOR_SELECT](#), [teamusa::CURSOR_SELECT](#),
 [teamusa::CursorStyle::CURSOR_LEFT](#), [teamusa::CursorStyle::CURSOR_RIGHT](#),
 [teamusa::CursorStyle::CURSOR_UP](#), [teamusa::CursorStyle::CURSOR_DOWN](#) }

The possible styles for the mouse cursor.

7.11.1 Detailed Description

Declares CursorStyle enumerations.

7.12 DelayedAudioActor.cpp File Reference

Implements DelayedAudioActor class.

```
#include "DelayedAudioActor.h"
```

7.12.1 Detailed Description

Implements DelayedAudioActor class.

7.13 DelayedAudioActor.h File Reference

Declares DelayedAudioActor class.

```
#include "BaseActor.h"
```

Classes

- class [teamusa::DelayedAudioActor](#)

Will increment a counter every time the step method is called.

Namespaces

- [teamusa](#)

7.13.1 Detailed Description

Declares DelayedAudioActor class.

7.14 DelayedVideoActor.cpp File Reference

Implements the DelayedVideoActor class.

```
#include "DelayedVideoActor.h"  
#include <iostream>
```

7.14.1 Detailed Description

Implements the DelayedVideoActor class.

7.15 DelayedVideoActor.h File Reference

Declares DelayedVideoActor class.

```
#include "BaseActor.h"
```

Classes

- class [teamusa::DelayedVideoActor](#)
Will increment a counter every time the step method is called.

Namespaces

- [teamusa](#)

7.15.1 Detailed Description

Declares DelayedVideoActor class.

7.16 Engine.cpp File Reference

Implements Engine class.


```
#include "Engine.h"
#include "Actor/AudioStreamActor.h"
#include "Actor/SceneLink.h"
#include "Actor/TextboxSpawnActor.h"
#include "Actor/VideoActor.h"
#include "Audio/AudioEngine.hpp"
#include "Engine/Assert.h"
#include "Engine/ResourceGroup.hpp"
#include "Engine/Timer.h"
#include "Video/VideoEngine.hpp"
```

Macros

- `#define BIND(function) (std::bind(function, this, std::placeholders::_1, std::placeholders::_2))`

Variables

- static const double `FRAME_TIME` = 16.67

7.16.1 Detailed Description

Implements Engine class.

7.16.2 Macro Definition Documentation

7.16.2.1 `#define BIND(function) (std::bind(function, this, std::placeholders::_1, std::placeholders::_2))`

7.16.3 Variable Documentation

7.16.3.1 `const double FRAME_TIME = 16.67` `[static]`

7.17 Engine.h File Reference

Declares Engine class.

```
#include "Headers.h"
#include "Engine/Level.h"
#include "GameSaveSerializer/GameSaveSerializer.h"
#include "Player/Player.h"
```

Classes

- class `teamusa::Engine`
Processes all components of the game each frame.

Namespaces

- `teamusa`

7.17.1 Detailed Description

Declares Engine class.

7.18 GameSaveSerializer.cpp File Reference

Implements save file serializer class.

```
#include "GameSaveSerializer.h"  
#include "Engine/Assert.h"
```

Namespaces

- [teamusa](#)

7.18.1 Detailed Description

Implements save file serializer class.

7.19 GameSaveSerializer.h File Reference

Declares save file serializer class.

```
#include <vector>  
#include <fstream>  
#include <mutex>  
#include <string>  
#include <thread>  
#include "Player/Player.h"
```

Classes

- class [teamusa::GameSaveSerializer](#)
Provides multithreaded save, single-thread load of save files.

Namespaces

- [teamusa](#)

7.19.1 Detailed Description

Declares save file serializer class.

7.20 Headers.h File Reference

Easy way to include all headers needed.

```
#include <exception>
#include <fstream>
#include <functional>
#include <iostream>
#include <map>
#include <memory>
#include <stack>
#include <string>
#include <vector>
#include <stdint.h>
```

7.20.1 Detailed Description

Easy way to include all headers needed.

7.21 InventoryItemActor.cpp File Reference

Implements InventoryItemActor class.

```
#include "InventoryItemActor.h"
#include "Player/Player.h"
```

7.21.1 Detailed Description

Implements InventoryItemActor class.

7.22 InventoryItemActor.h File Reference

Declares InventoryItemActor class.

```
#include "BaseActor.h"
```

Classes

- class [teamusa::InventoryItemActor](#)

InventoryItemActor creates a collectible item in the game environment.

Namespaces

- [teamusa](#)

7.22.1 Detailed Description

Declares InventoryItemActor class.

7.23 Level.cpp File Reference

Implements Level class.

```
#include "Assert.h"
#include "Level.h"
#include "Actor/ActorEvent.h"
#include "Actor/AudioStreamActor.h"
#include "Actor/DelayedAudioActor.h"
#include "Actor/DelayedVideoActor.h"
#include "Actor/InventoryItemActor.h"
#include "Actor/LevelLink.h"
#include "Actor/MovingActor.h"
#include "Actor/ResponsiveAudioActor.h"
#include "Actor/ResponsiveVideoActor.h"
#include "Actor/SceneLink.h"
#include "Actor/TextboxSpawnActor.h"
#include "Actor/VideoActor.h"
#include "Actor/VideoEventActor.h"
#include "Audio/AudioEngine.hpp"
#include "Video/VideoEngine.hpp"
```

Functions

- static std::istream & [operator>>](#) (std::istream &fs, [Region](#) &dst)
- static std::istream & [operator>>](#) (std::istream &fs, [ActorEventType](#) &dst)
- static void [loadError](#) (const std::string &msg)

7.23.1 Detailed Description

Implements Level class.

7.23.2 Function Documentation

7.23.2.1 static void [loadError](#) (const std::string & *msg*) [static]

7.23.2.2 static std::istream& [operator>>](#) (std::istream & *fs*, [Region](#) & *dst*) [inline],[static]

7.23.2.3 static std::istream& [operator>>](#) (std::istream & *fs*, [ActorEventType](#) & *dst*) [inline],[static]

7.24 Level.h File Reference

Declares Level class.

```
#include <unordered_map>
#include "Headers.h"
```

Classes

- class [teamusa::Level](#)
A [Level](#) is a container of Scenes and Actors corresponding to those scenes.
- class [teamusa::Level::Scene](#)

A scene is a collection of images (Actors) that is displayed on the screen.

Namespaces

- [teamusa](#)

Typedefs

- typedef std::shared_ptr< BaseActor > [teamusa::BaseActorPtr](#)
- typedef std::vector< BaseActorPtr > [teamusa::ActorList](#)

7.24.1 Detailed Description

Declares Level class.

7.25 LevelLink.cpp File Reference

Implements LevelLink class.

```
#include "LevelLink.h"
#include "Player/Player.h"
```

7.25.1 Detailed Description

Implements LevelLink class.

7.26 LevelLink.h File Reference

Declares LevelLink class.

```
#include "BaseActor.h"
```

Classes

- class [teamusa::LevelLink](#)
Allows the player to transition between levels.

Namespaces

- [teamusa](#)

7.26.1 Detailed Description

Declares LevelLink class.

7.27 main.cpp File Reference

Entry point of program.

```
#include "Headers.h"  
#include "Engine/Engine.h"
```

Namespaces

- [MainNS](#)

Functions

- static void [MainNS::logError](#) (const std::string &desc)
Writes an error message to the log file.
- int [main](#) (int argc, char **argv)

7.27.1 Detailed Description

Entry point of program.

7.27.2 Function Documentation

7.27.2.1 int main (int argc, char ** argv)

7.28 MovingActor.cpp File Reference

Implements the MovingActor class.

```
#include "MovingActor.h"  
#include "Player/Player.h"
```

7.28.1 Detailed Description

Implements the MovingActor class.

7.29 MovingActor.h File Reference

Declares MovingActor class.

```
#include "BaseActor.h"
```

Classes

- class [teamusa::MovingActor](#)
Will transition from one region to the next by calculating the distance to move each frame for a set number of frames.

Namespaces

- [teamusa](#)

7.29.1 Detailed Description

Declares MovingActor class.

7.30 Player.cpp File Reference

Implements Player class.

```
#include "Player.h"
#include "Engine/Assert.h"
```

7.30.1 Detailed Description

Implements Player class.

7.31 Player.h File Reference

Declares Player class.

```
#include "Headers.h"
#include "CursorStyle.h"
#include "Engine/Point.h"
#include "Video/VideoEngine.hpp"
```

Classes

- class [teamusa::Player](#)
Handles all data relevant to the player engaging the game.

Namespaces

- [teamusa](#)

7.31.1 Detailed Description

Declares Player class.

7.32 Point.h File Reference

Declares Point struct.

```
#include <stdint.h>
```

Classes

- class [teamusa::Point](#)

An (x,y) coordinate within the rendering window.

Namespaces

- [teamusa](#)

7.32.1 Detailed Description

Declares Point struct.

7.33 ResourceGroup.hpp File Reference

Declares the ResourceGroup enum types.

Enumerations

- enum [ResourceGroup](#) { [CORE_RESOURCE](#), [LEVEL_RESOURCE](#) }

7.33.1 Detailed Description

Declares the ResourceGroup enum types.

7.33.2 Enumeration Type Documentation

7.33.2.1 enum ResourceGroup

Enumerator

CORE_RESOURCE

LEVEL_RESOURCE

7.34 ResponsiveAudioActor.cpp File Reference

Implements ResponsiveAudioActor class.

```
#include "ResponsiveAudioActor.h"
```

7.34.1 Detailed Description

Implements ResponsiveAudioActor class.

7.35 ResponsiveAudioActor.h File Reference

Declares ResponsiveAudioActor class.

```
#include "BaseActor.h"
```

Classes

- class [teamusa::ResponsiveAudioActor](#)
Will increment the value of stepCount until it is equal to durationSteps for each call to the step method.

Namespaces

- [teamusa](#)

7.35.1 Detailed Description

Declares ResponsiveAudioActor class.

7.36 ResponsiveVideoActor.cpp File Reference

Implements the ResponsiveVideoActor class.

```
#include "ResponsiveVideoActor.h"
```

7.36.1 Detailed Description

Implements the ResponsiveVideoActor class.

7.37 ResponsiveVideoActor.h File Reference

Declares ResponsivevideoActor class.

```
#include "BaseActor.h"
```

Classes

- class [teamusa::ResponsiveVideoActor](#)
Changes its texture ID based on hovering and clicks.

Namespaces

- [teamusa](#)

7.37.1 Detailed Description

Declares ResponsivevideoActor class.

7.38 SceneLink.cpp File Reference

Implements SceneLink class.

```
#include "SceneLink.h"  
#include "Player/Player.h"
```

7.38.1 Detailed Description

Implements SceneLink class.

7.39 SceneLink.h File Reference

Declares SceneLink class.

```
#include "BaseActor.h"  
#include "Player/CursorStyle.h"
```

Classes

- class [teamusa::SceneLink](#)
Allows the player to transition between scenes.

Namespaces

- [teamusa](#)

7.39.1 Detailed Description

Declares SceneLink class.

7.40 TextboxSpawnActor.cpp File Reference

Implements TextboxSpawnActor class.

```
#include "TextboxSpawnActor.h"
```

7.40.1 Detailed Description

Implements TextboxSpawnActor class.

7.41 TextboxSpawnActor.h File Reference

Declares TextboxSpawnActor class.

```
#include "BaseActor.h"  
#include <string>
```

Classes

- class [teamusa::TextboxSpawnActor](#)

Will emit a `DisplayText` event when the `onClick` method is called.

Namespaces

- [teamusa](#)

7.41.1 Detailed Description

Declares TextboxSpawnActor class.

7.42 Timer.cpp File Reference

Implements Timer class.

```
#include "Engine/Timer.h"  
#include <SDL2/SDL.h>
```

7.42.1 Detailed Description

Implements Timer class.

7.43 Timer.h File Reference

Declares Timer class.

```
#include "Headers.h"
```

Classes

- class [teamusa::Timer](#)

A timer that counts up from zero in milliseconds.

Namespaces

- [teamusa](#)

7.43.1 Detailed Description

Declares Timer class.

7.44 VideoActor.cpp File Reference

Implements VideoActor class.

```
#include "VideoActor.h"
```

7.44.1 Detailed Description

Implements VideoActor class.

7.45 VideoActor.h File Reference

Declares the VideoActor class This module makes sure An actor that will only display a texture at a given region.

```
#include "BaseActor.h"
```

Classes

- class [teamusa::VideoActor](#)
Displays a texture in a region and performs no other behavior.

Namespaces

- [teamusa](#)

7.45.1 Detailed Description

Declares the VideoActor class This module makes sure An actor that will only display a texture at a given region.

This actor will have no interaction with the player.

7.46 VideoContext.cpp File Reference

Implements the VideoContext class.

```
#include "VideoContext.hpp"
```

7.46.1 Detailed Description

Implements the VideoContext class.

7.47 VideoContext.hpp File Reference

Declares the VideoContext class.

```
#include <unordered_map>
#include <string>
#include "SDL2/SDL.h"
#include "SDL2/SDL_image.h"
#include "SDL2/SDL_ttf.h"
#include "VideoDisplay.hpp"
```

Classes

- class [mediawrap::VideoContext](#)
Provides basic 2D rendering capabilities.

Namespaces

- [mediawrap](#)

7.47.1 Detailed Description

Declares the VideoContext class.

7.48 VideoDisplay.cpp File Reference

Implements the VideoDisplay class.

```
#include "VideoDisplay.hpp"
```

7.48.1 Detailed Description

Implements the VideoDisplay class.

7.49 VideoDisplay.hpp File Reference

Declares the VideoDisplay class.

```
#include <stdexcept>
#include "SDL2/SDL.h"
#include "SDL2/SDL_image.h"
#include "SDL2/SDL_ttf.h"
```

Classes

- class [mediawrap::VideoDisplay](#)
Creates a window and initializes SDL2 and SDL2_IMG.

Namespaces

- [mediawrap](#)

7.49.1 Detailed Description

Declares the VideoDisplay class.

7.50 VideoEngine.cpp File Reference

Implements the VideoEngine class.

```
#include "VideoEngine.hpp"
```

7.50.1 Detailed Description

Implements the VideoEngine class.

7.51 VideoEngine.hpp File Reference

Declares the VideoEngine class.

```
#include <stdexcept>
#include <string>
#include <vector>
#include "VideoContext.hpp"
#include "Engine/ResourceGroup.hpp"
```

Classes

- class [teamusa::VideoEngine](#)
Provides video capabilities that are specific to Legend of the Great Unwashed.

Namespaces

- [teamusa](#)

Typedefs

- typedef [mediawrap::VideoContext::TextureID](#) [teamusa::TextureID](#)
- typedef [mediawrap::VideoContext::Region](#) [teamusa::Region](#)

7.51.1 Detailed Description

Declares the VideoEngine class.

7.52 VideoEventActor.cpp File Reference

Implements the VideoEventActor class.

```
#include "VideoEventActor.h"
#include "Player/Player.h"
```

7.52.1 Detailed Description

Implements the VideoEventActor class.

7.53 VideoEventActor.h File Reference

Declares VideoEventActor class.

```
#include "BaseActor.h"
```

Classes

- class [teamusa::VideoEventActor](#)
Will display a texture and perform no action until clicked.

Namespaces

- [teamusa](#)

7.53.1 Detailed Description

Declares VideoEventActor class.

Index

- ~AudioPlayer
 - mediawrap::AudioPlayer, [19](#)
- ~AudioStreamActor
 - teamusa::AudioStreamActor, [21](#)
- ~BaseActor
 - teamusa::BaseActor, [24](#)
- ~DelayedAudioActor
 - teamusa::DelayedAudioActor, [27](#)
- ~DelayedVideoActor
 - teamusa::DelayedVideoActor, [28](#)
- ~Engine
 - teamusa::Engine, [30](#)
- ~GameSaveSerializer
 - teamusa::GameSaveSerializer, [33](#)
- ~InventoryItemActor
 - teamusa::InventoryItemActor, [35](#)
- ~LevelLink
 - teamusa::LevelLink, [39](#)
- ~MovingActor
 - teamusa::MovingActor, [42](#)
- ~Player
 - teamusa::Player, [44](#)
- ~ResponsiveAudioActor
 - teamusa::ResponsiveAudioActor, [48](#)
- ~ResponsiveVideoActor
 - teamusa::ResponsiveVideoActor, [51](#)
- ~SceneLink
 - teamusa::SceneLink, [53](#)
- ~TextboxSpawnActor
 - teamusa::TextboxSpawnActor, [55](#)
- ~Timer
 - teamusa::Timer, [57](#)
- ~VideoActor
 - teamusa::VideoActor, [58](#)
- ~VideoContext
 - mediawrap::VideoContext, [61](#)
- ~VideoDisplay
 - mediawrap::VideoDisplay, [65](#)
- ~VideoEngine
 - teamusa::VideoEngine, [67](#)
- ~VideoEventActor
 - teamusa::VideoEventActor, [70](#)
- activated
 - teamusa::AudioStreamActor, [22](#)
 - teamusa::TextboxSpawnActor, [56](#)
- activeScene
 - teamusa::Level, [38](#)
- ActorEvent
 - teamusa::ActorEvent, [15](#)
- actorEvent
 - teamusa::VideoEventActor, [72](#)
- ActorEvent.h, [73](#)
- ActorEventHandler
 - teamusa::Engine, [30](#)
- ActorEventType
 - teamusa, [13](#)
- ActorList
 - teamusa, [13](#)
- ActorVideo
 - teamusa::ActorVideo, [16](#)
- actors
 - teamusa::Level::Scene, [52](#)
- addItem
 - teamusa::Player, [44](#)
- Assert
 - Assert.h, [74](#)
- Assert.h, [73](#)
- Assert, [74](#)
- audio_buffer
 - mediawrap::AudioPlayer, [20](#)
- audio_channels
 - mediawrap::AudioPlayer, [20](#)
- audio_format
 - mediawrap::AudioPlayer, [20](#)
- audio_rate
 - mediawrap::AudioPlayer, [20](#)
- audio_samples
 - mediawrap::AudioPlayer, [20](#)
- audio_stream
 - mediawrap::AudioPlayer, [20](#)
- AudioEngine.cpp, [74](#)
- AudioEngine.hpp, [74](#)
- AudioID
 - mediawrap::AudioPlayer, [19](#)
 - teamusa, [13](#)
- audiold
 - teamusa::DelayedAudioActor, [27](#)
- AudioPlayer
 - mediawrap::AudioPlayer, [19](#)
- audioPlayer
 - teamusa::AudioEngine, [18](#)
- AudioPlayer.cpp, [75](#)
- AudioPlayer.hpp, [75](#)
- AudioStreamActor
 - teamusa::AudioStreamActor, [21](#)
- AudioStreamActor.cpp, [75](#)
- AudioStreamActor.h, [75](#)
- BIND

- Engine.cpp, 79
- BLENDMODE_ADD
 - mediawrap::VideoContext, 61
- BLENDMODE_BLEND
 - mediawrap::VideoContext, 61
- BLENDMODE_MOD
 - mediawrap::VideoContext, 61
- BLENDMODE_NONE
 - mediawrap::VideoContext, 61
- BLUE
 - mediawrap::VideoContext, 61
- BaseActor
 - teamusa::BaseActor, 24
- BaseActor.cpp, 76
- BaseActor.h, 76
- BaseActorPtr
 - teamusa, 13
- bglImageID
 - teamusa::Level::Scene, 52
- BlendMode
 - mediawrap::VideoContext, 61
- CORE_RESOURCE
 - ResourceGroup.hpp, 86
- CURSOR_DEFAULT
 - teamusa, 13
- CURSOR_DEFAULT_ID
 - teamusa::Player, 46
- CURSOR_DOWN
 - teamusa, 14
- CURSOR_DOWN_ID
 - teamusa::Player, 46
- CURSOR_LEFT
 - teamusa, 13
- CURSOR_LEFT_ID
 - teamusa::Player, 46
- CURSOR_RIGHT
 - teamusa, 13
- CURSOR_RIGHT_ID
 - teamusa::Player, 46
- CURSOR_SELECT
 - teamusa, 13
- CURSOR_SELECT_ID
 - teamusa::Player, 46
- CURSOR_UP
 - teamusa, 14
- CURSOR_UP_ID
 - teamusa::Player, 46
- ChangeScene
 - teamusa, 13
- changeScene
 - teamusa::Level, 37
- clear_samples
 - mediawrap::AudioPlayer, 19
- clearAll
 - teamusa::Level, 37
- clearLayers
 - teamusa::VideoEngine, 68
- clickAudioId
 - teamusa::ResponsiveAudioActor, 50
- clickTexture
 - teamusa::ResponsiveVideoActor, 52
- coreResources
 - teamusa::AudioEngine, 18
 - teamusa::VideoEngine, 69
- create_texture
 - mediawrap::VideoContext, 62
- currentStep
 - teamusa::DelayedAudioActor, 27
 - teamusa::DelayedVideoActor, 29
 - teamusa::MovingActor, 43
- CursorStyle
 - teamusa, 13
- cursorStyle
 - teamusa::SceneLink, 54
- CursorStyle.h, 77
- DebugColor
 - mediawrap::VideoContext, 61
- defaultTextureId
 - teamusa::ResponsiveVideoActor, 52
- delaySteps
 - teamusa::DelayedAudioActor, 27
 - teamusa::DelayedVideoActor, 29
- DelayedAudioActor
 - teamusa::DelayedAudioActor, 27
- DelayedAudioActor.cpp, 77
- DelayedAudioActor.h, 77
- DelayedVideoActor
 - teamusa::DelayedVideoActor, 28
- DelayedVideoActor.cpp, 78
- DelayedVideoActor.h, 78
- delete_sample
 - mediawrap::AudioPlayer, 19
- delete_texture
 - mediawrap::VideoContext, 62
- deleteResourceGroup
 - teamusa::VideoEngine, 68
- deleteSound
 - teamusa::AudioEngine, 17
- deleteSoundGroup
 - teamusa::AudioEngine, 17
- deleteTexture
 - teamusa::VideoEngine, 68
- disappear
 - teamusa::DelayedVideoActor, 29
- display
 - mediawrap::VideoContext, 62
 - teamusa::VideoEngine, 68
- DisplayText
 - teamusa, 13
- endRegion
 - teamusa::MovingActor, 43
- Engine
 - teamusa::Engine, 30
- Engine.cpp, 78
- BIND, 79

- FRAME_TIME, 79
- Engine.h, 79
- ExitGame
 - teamusa, 13
- FLASHLIGHT_ID
 - teamusa::Player, 46
- FLIP_HORIZONTAL
 - mediawrap::VideoContext, 61
- FLIP_NONE
 - mediawrap::VideoContext, 61
- FLIP_VERTICAL
 - mediawrap::VideoContext, 61
- FRAME_TIME
 - Engine.cpp, 79
- fileLock
 - teamusa::GameSaveSerializer, 34
- fill_texture
 - mediawrap::VideoContext, 62
- Flip
 - mediawrap::VideoContext, 61
- font
 - mediawrap::VideoContext, 64
- freeAndLoadLevel
 - teamusa::Engine, 30
- GREEN
 - mediawrap::VideoContext, 61
- GameSaveSerializer
 - teamusa::GameSaveSerializer, 33
- GameSaveSerializer.cpp, 80
- GameSaveSerializer.h, 80
- get_renderer
 - mediawrap::VideoDisplay, 66
- getActors
 - teamusa::Level, 37
- getBGImageID
 - teamusa::Level, 37
- getCursorTextureID
 - teamusa::Player, 45
- getInventory
 - teamusa::Player, 45
- getLayer
 - teamusa::BaseActor, 24
- getMouseClickedState
 - teamusa::Engine, 30
- getMouseCoordinates
 - teamusa::Engine, 30
- getPath
 - teamusa::AudioStreamActor, 21
- getPosition
 - teamusa::Player, 45
- getRegion
 - teamusa::BaseActor, 24
- getScene
 - teamusa::Level, 38
- getSceneID
 - teamusa::LevelLink, 39
- getText
 - teamusa::LevelLink, 40
 - teamusa::SceneLink, 53
 - teamusa::TextboxSpawnActor, 55
- getTextureID
 - teamusa::BaseActor, 24
- getTicks
 - teamusa::Timer, 57
- hGrowth
 - teamusa::MovingActor, 43
- handleEvent
 - teamusa::Engine, 31
- hasItem
 - teamusa::Player, 45
- hasVideo
 - teamusa::BaseActor, 24
- Headers.h, 80
- hideTextbox
 - teamusa::VideoEngine, 68
- hoverAudioId
 - teamusa::ResponsiveAudioActor, 50
- hoverTexture
 - teamusa::ResponsiveVideoActor, 52
- Inventory
 - teamusa::Player, 44
- InventoryItemActor
 - teamusa::InventoryItemActor, 35
- InventoryItemActor.cpp, 81
- InventoryItemActor.h, 81
- isActive
 - teamusa::MovingActor, 43
- isInBounds
 - teamusa::BaseActor, 24
- isShowingTextbox
 - teamusa::VideoEngine, 68
- itemId
 - teamusa::InventoryItemActor, 36
- itemRequiredText
 - teamusa::LevelLink, 41
 - teamusa::SceneLink, 54
- LEVEL_RESOURCE
 - ResourceGroup.hpp, 86
- layer
 - teamusa::ActorVideo, 16
- layers
 - teamusa::VideoEngine, 69
- Level
 - teamusa::Level, 37
- Level.cpp, 82
 - loadError, 82
 - operator>>, 82
- Level.h, 82
- levelID
 - teamusa::LevelLink, 41
- LevelLink
 - teamusa::LevelLink, 39
- LevelLink.cpp, 83

- LevelLink.h, 83
- levelResources
 - teamusa::AudioEngine, 18
 - teamusa::VideoEngine, 69
- load
 - teamusa::GameSaveSerializer, 33
- load_font
 - mediawrap::VideoContext, 62
- load_sample
 - mediawrap::AudioPlayer, 20
- load_stream
 - mediawrap::AudioPlayer, 20
- load_texture
 - mediawrap::VideoContext, 63
- loadError
 - Level.cpp, 82
- LoadGame
 - teamusa, 13
- LoadLevel
 - teamusa, 13
- loadLevel
 - teamusa::Level, 38
- loadSound
 - teamusa::AudioEngine, 17
- loadTexture
 - teamusa::VideoEngine, 68
- logError
 - MainNS, 11
- MAX_RESERVED_ID
 - teamusa::AudioEngine, 18
 - teamusa::VideoEngine, 69
- mActorEventHandlers
 - teamusa::Engine, 32
- mAudioEngine
 - teamusa::Engine, 32
- mAudioID
 - teamusa::BaseActor, 26
- mCurrentLevelID
 - teamusa::Engine, 32
- mCursorStyle
 - teamusa::Player, 46
- mInventory
 - teamusa::Player, 46
- mIsRunning
 - teamusa::Engine, 32
- mLayer
 - teamusa::Player, 46
- mLevel
 - teamusa::Engine, 32
- mMainMenu
 - teamusa::Engine, 32
- MOUSE_CLICK_ID
 - teamusa::Player, 46
- mPauseTicks
 - teamusa::Timer, 57
- mPaused
 - teamusa::Timer, 57
- mPlayer
 - teamusa::Engine, 33
- mPosition
 - teamusa::Player, 46
- mRegion
 - teamusa::BaseActor, 26
 - teamusa::Player, 46
- mSerializer
 - teamusa::Engine, 33
- mStartTicks
 - teamusa::Timer, 57
- mStarted
 - teamusa::Timer, 57
- mTextureID
 - teamusa::Player, 46
- mVideo
 - teamusa::BaseActor, 26
- mVideoEngine
 - teamusa::Engine, 33
- main
 - main.cpp, 84
- main.cpp, 84
 - main, 84
- MainNS, 11
 - logError, 11
- mediawrap, 11
 - mediawrap::AudioPlayer, 18
 - ~AudioPlayer, 19
 - audio_buffer, 20
 - audio_channels, 20
 - audio_format, 20
 - audio_rate, 20
 - audio_samples, 20
 - audio_stream, 20
 - AudioID, 19
 - AudioPlayer, 19
 - clear_samples, 19
 - delete_sample, 19
 - load_sample, 20
 - load_stream, 20
 - play_sample, 20
 - stream_audio, 20
 - mediawrap::VideoContext, 59
 - ~VideoContext, 61
 - BLENDDMODE_ADD, 61
 - BLENDDMODE_BLEND, 61
 - BLENDDMODE_MOD, 61
 - BLENDDMODE_NONE, 61
 - BLUE, 61
 - BlendMode, 61
 - create_texture, 62
 - DebugColor, 61
 - delete_texture, 62
 - display, 62
 - FLIP_HORIZONTAL, 61
 - FLIP_NONE, 61
 - FLIP_VERTICAL, 61
 - fill_texture, 62
 - Flip, 61

- font, [64](#)
- GREEN, [61](#)
- load_font, [62](#)
- load_texture, [63](#)
- RED, [61](#)
- Region, [60](#)
- render, [63](#)
- render_clear, [63](#)
- render_onto, [63](#)
- render_rotate, [64](#)
- render_text, [64](#)
- renderDebugBox, [64](#)
- renderer, [64](#)
- swapFullscreen, [64](#)
- texture_iter, [60](#)
- TextureID, [60](#)
- textures, [64](#)
- video_display, [64](#)
- VideoContext, [61](#)
- mediawrap::VideoDisplay, [65](#)
 - ~VideoDisplay, [65](#)
 - get_renderer, [66](#)
 - swapFullscreen, [66](#)
 - VideoDisplay, [65](#)
 - window, [66](#)
- MovingActor
 - teamusa::MovingActor, [42](#)
- MovingActor.cpp, [84](#)
- MovingActor.h, [84](#)
- NUM_LAYERS
 - teamusa::VideoEngine, [69](#)
- NewGame
 - teamusa, [13](#)
- Nil
 - teamusa, [13](#)
- onChangeScene
 - teamusa::Engine, [31](#)
- onClick
 - teamusa::BaseActor, [25](#)
 - teamusa::InventoryItemActor, [35](#)
 - teamusa::LevelLink, [40](#)
 - teamusa::MovingActor, [42](#)
 - teamusa::ResponsiveAudioActor, [48](#)
 - teamusa::ResponsiveVideoActor, [51](#)
 - teamusa::SceneLink, [53](#)
 - teamusa::TextboxSpawnActor, [55](#)
 - teamusa::VideoEventActor, [70](#)
- onDisplayText
 - teamusa::Engine, [31](#)
- onExitGame
 - teamusa::Engine, [31](#)
- onHover
 - teamusa::BaseActor, [25](#)
 - teamusa::InventoryItemActor, [35](#)
 - teamusa::LevelLink, [40](#)
 - teamusa::MovingActor, [42](#)
 - teamusa::ResponsiveAudioActor, [48](#)
 - teamusa::ResponsiveVideoActor, [51](#)
 - teamusa::SceneLink, [54](#)
 - teamusa::VideoEventActor, [72](#)
- onLoadGame
 - teamusa::Engine, [31](#)
- onLoadLevel
 - teamusa::Engine, [31](#)
- onNewGame
 - teamusa::Engine, [32](#)
- onPlayAudio
 - teamusa::Engine, [32](#)
- onStreamAudio
 - teamusa::Engine, [32](#)
- operator>>
 - Level.cpp, [82](#)
- parseAudioStreamActor
 - teamusa::Level, [38](#)
- parseDelayedAudioActor
 - teamusa::Level, [38](#)
- parseDelayedVideoActor
 - teamusa::Level, [38](#)
- parseInventoryItemActor
 - teamusa::Level, [38](#)
- parseLevelLink
 - teamusa::Level, [38](#)
- parseMovingActor
 - teamusa::Level, [38](#)
- parseResponsiveAudioActor
 - teamusa::Level, [38](#)
- parseResponsiveVideoActor
 - teamusa::Level, [38](#)
- parseSceneLink
 - teamusa::Level, [38](#)
- parseTextboxSpawnActor
 - teamusa::Level, [38](#)
- parseVideoActor
 - teamusa::Level, [38](#)
- parseVideoEventActor
 - teamusa::Level, [38](#)
- path
 - teamusa::AudioStreamActor, [22](#)
- pause
 - teamusa::Timer, [57](#)
- pickedUp
 - teamusa::InventoryItemActor, [36](#)
- play_sample
 - mediawrap::AudioPlayer, [20](#)
- PlayAudio
 - teamusa, [13](#)
- playSound
 - teamusa::AudioEngine, [17](#)
- playStream
 - teamusa::AudioEngine, [17](#)
- Player
 - teamusa::Player, [44](#)
- Player.cpp, [85](#)
- Player.h, [85](#)
- Point

- teamusa::Point, 47
- Point.h, 85
- RED
 - mediawrap::VideoContext, 61
- Region
 - mediawrap::VideoContext, 60
 - teamusa, 13
- render
 - mediawrap::VideoContext, 63
 - teamusa::Engine, 32
 - teamusa::VideoEngine, 68
- render_clear
 - mediawrap::VideoContext, 63
- render_onto
 - mediawrap::VideoContext, 63
- render_rotate
 - mediawrap::VideoContext, 64
- render_text
 - mediawrap::VideoContext, 64
- renderDebugBox
 - mediawrap::VideoContext, 64
 - teamusa::VideoEngine, 69
- renderRotate
 - teamusa::VideoEngine, 69
- renderer
 - mediawrap::VideoContext, 64
- requiredItemID
 - teamusa::LevelLink, 41
 - teamusa::SceneLink, 54
- ResourceGroup
 - ResourceGroup.hpp, 86
- ResourceGroup.hpp, 86
 - CORE_RESOURCE, 86
 - LEVEL_RESOURCE, 86
 - ResourceGroup, 86
- ResponsiveAudioActor
 - teamusa::ResponsiveAudioActor, 48
- ResponsiveAudioActor.cpp, 86
- ResponsiveAudioActor.h, 87
- ResponsiveVideoActor
 - teamusa::ResponsiveVideoActor, 51
- ResponsiveVideoActor.cpp, 87
- ResponsiveVideoActor.h, 87
- run
 - teamusa::Engine, 32
- SHADOW_LAYER
 - teamusa::VideoEngine, 69
- save
 - teamusa::GameSaveSerializer, 34
- saveInThread
 - teamusa::GameSaveSerializer, 34
- sceneID
 - teamusa::LevelLink, 41
 - teamusa::SceneLink, 54
- SceneLink
 - teamusa::SceneLink, 53
- SceneLink.cpp, 88
- SceneLink.h, 88
- scenes
 - teamusa::Level, 38
- setCursor
 - teamusa::Player, 45
- setInventory
 - teamusa::Player, 45
- setPosition
 - teamusa::Player, 45, 46
- setRegion
 - teamusa::BaseActor, 25
- setSlot
 - teamusa::GameSaveSerializer, 34
- setTextureId
 - teamusa::ResponsiveVideoActor, 51
- showTextbox
 - teamusa::VideoEngine, 69
- slot
 - teamusa::GameSaveSerializer, 34
- start
 - teamusa::Timer, 57
- startScene
 - teamusa::Level, 38
- step
 - teamusa::AudioStreamActor, 22
 - teamusa::BaseActor, 25
 - teamusa::DelayedAudioActor, 27
 - teamusa::DelayedVideoActor, 28
 - teamusa::InventoryItemActor, 35
 - teamusa::LevelLink, 40
 - teamusa::MovingActor, 42
 - teamusa::ResponsiveAudioActor, 48
 - teamusa::ResponsiveVideoActor, 51
 - teamusa::SceneLink, 54
 - teamusa::TextboxSpawnActor, 56
 - teamusa::VideoActor, 58
 - teamusa::VideoEventActor, 72
- stop
 - teamusa::Timer, 57
- stream_audio
 - mediawrap::AudioPlayer, 20
- StreamAudio
 - teamusa, 13
- swapFullscreen
 - mediawrap::VideoContext, 64
 - mediawrap::VideoDisplay, 66
 - teamusa::VideoEngine, 69
- TEXT_LAYER
 - teamusa::VideoEngine, 69
- teamusa, 11
 - ActorEventType, 13
 - ActorList, 13
 - AudioID, 13
 - BaseActorPtr, 13
 - CURSOR_DEFAULT, 13
 - CURSOR_DOWN, 14
 - CURSOR_LEFT, 13
 - CURSOR_RIGHT, 13

- CURSOR_SELECT, 13
- CURSOR_UP, 14
- ChangeScene, 13
- CursorStyle, 13
- DisplayText, 13
- ExitGame, 13
- LoadGame, 13
- LoadLevel, 13
- NewGame, 13
- Nil, 13
- PlayAudio, 13
- Region, 13
- StreamAudio, 13
- TextureID, 13
- teamusa::ActorEvent, 15
 - ActorEvent, 15
 - type, 15
 - value, 15
- teamusa::ActorVideo, 15
 - ActorVideo, 16
 - layer, 16
 - textureID, 16
- teamusa::AudioEngine, 16
 - audioPlayer, 18
 - coreResources, 18
 - deleteSound, 17
 - deleteSoundGroup, 17
 - levelResources, 18
 - loadSound, 17
 - MAX_RESERVED_ID, 18
 - playSound, 17
 - playStream, 17
- teamusa::AudioStreamActor, 21
 - ~AudioStreamActor, 21
 - activated, 22
 - AudioStreamActor, 21
 - getPath, 21
 - path, 22
 - step, 22
- teamusa::BaseActor, 22
 - ~BaseActor, 24
 - BaseActor, 24
 - getLayer, 24
 - getRegion, 24
 - getTextureID, 24
 - hasVideo, 24
 - isInBounds, 24
 - mAudioID, 26
 - mRegion, 26
 - mVideo, 26
 - onClick, 25
 - onHover, 25
 - setRegion, 25
 - step, 25
- teamusa::DelayedAudioActor, 26
 - ~DelayedAudioActor, 27
 - audioid, 27
 - currentStep, 27
 - delaySteps, 27
 - DelayedAudioActor, 27
 - step, 27
- teamusa::DelayedVideoActor, 27
 - ~DelayedVideoActor, 28
 - currentStep, 29
 - delaySteps, 29
 - DelayedVideoActor, 28
 - disappear, 29
 - step, 28
 - textureId, 29
- teamusa::Engine, 29
 - ~Engine, 30
 - ActorEventHandler, 30
 - Engine, 30
 - freeAndLoadLevel, 30
 - getMouseClickState, 30
 - getMouseCoordinates, 30
 - handleEvent, 31
 - mActorEventHandlers, 32
 - mAudioEngine, 32
 - mCurrentLevelID, 32
 - mIsRunning, 32
 - mLevel, 32
 - mMainMenu, 32
 - mPlayer, 33
 - mSerializer, 33
 - mVideoEngine, 33
 - onChangeScene, 31
 - onDisplayText, 31
 - onExitGame, 31
 - onLoadGame, 31
 - onLoadLevel, 31
 - onNewGame, 32
 - onPlayAudio, 32
 - onStreamAudio, 32
 - render, 32
 - run, 32
- teamusa::GameSaveSerializer, 33
 - ~GameSaveSerializer, 33
 - fileLock, 34
 - GameSaveSerializer, 33
 - load, 33
 - save, 34
 - saveInThread, 34
 - setSlot, 34
 - slot, 34
- teamusa::InventoryItemActor, 34
 - ~InventoryItemActor, 35
 - InventoryItemActor, 35
 - itemID, 36
 - onClick, 35
 - onHover, 35
 - pickedUp, 36
 - step, 35
- teamusa::Level, 36
 - activeScene, 38
 - changeScene, 37

- clearAll, 37
- getActors, 37
- getBGImageID, 37
- getScene, 38
- Level, 37
- loadLevel, 38
- parseAudioStreamActor, 38
- parseDelayedAudioActor, 38
- parseDelayedVideoActor, 38
- parseInventoryItemActor, 38
- parseLevelLink, 38
- parseMovingActor, 38
- parseResponsiveAudioActor, 38
- parseResponsiveVideoActor, 38
- parseSceneLink, 38
- parseTextboxSpawnActor, 38
- parseVideoActor, 38
- parseVideoEventActor, 38
- scenes, 38
- startScene, 38
- teamusa::Level::Scene, 52
 - actors, 52
 - bgImageID, 52
- teamusa::LevelLink, 39
 - ~LevelLink, 39
 - getSceneID, 39
 - getText, 40
 - itemRequiredText, 41
 - levelID, 41
 - LevelLink, 39
 - onClick, 40
 - onHover, 40
 - requiredItemID, 41
 - sceneID, 41
 - step, 40
- teamusa::MovingActor, 41
 - ~MovingActor, 42
 - currentStep, 43
 - endRegion, 43
 - hGrowth, 43
 - isActive, 43
 - MovingActor, 42
 - onClick, 42
 - onHover, 42
 - step, 42
 - transitionSteps, 43
 - wGrowth, 43
 - xSpeed, 43
 - ySpeed, 43
- teamusa::Player, 43
 - ~Player, 44
 - addItem, 44
 - CURSOR_DEFAULT_ID, 46
 - CURSOR_DOWN_ID, 46
 - CURSOR_LEFT_ID, 46
 - CURSOR_RIGHT_ID, 46
 - CURSOR_SELECT_ID, 46
 - CURSOR_UP_ID, 46
 - FLASHLIGHT_ID, 46
 - getCursorTextureID, 45
 - getInventory, 45
 - getPosition, 45
 - hasItem, 45
 - Inventory, 44
 - mCursorStyle, 46
 - mInventory, 46
 - mLayer, 46
 - MOUSE_CLICK_ID, 46
 - mPosition, 46
 - mRegion, 46
 - mTextureID, 46
 - Player, 44
 - setCursor, 45
 - setInventory, 45
 - setPosition, 45, 46
- teamusa::Point, 46
 - Point, 47
 - x, 47
 - y, 47
- teamusa::ResponsiveAudioActor, 47
 - ~ResponsiveAudioActor, 48
 - clickAudioID, 50
 - hoverAudioID, 50
 - onClick, 48
 - onHover, 48
 - ResponsiveAudioActor, 48
 - step, 48
- teamusa::ResponsiveVideoActor, 50
 - ~ResponsiveVideoActor, 51
 - clickTexture, 52
 - defaultTextureID, 52
 - hoverTexture, 52
 - onClick, 51
 - onHover, 51
 - ResponsiveVideoActor, 51
 - setTextureID, 51
 - step, 51
- teamusa::SceneLink, 52
 - ~SceneLink, 53
 - cursorStyle, 54
 - getText, 53
 - itemRequiredText, 54
 - onClick, 53
 - onHover, 54
 - requiredItemID, 54
 - sceneID, 54
 - SceneLink, 53
 - step, 54
- teamusa::TextboxSpawnActor, 55
 - ~TextboxSpawnActor, 55
 - activated, 56
 - getText, 55
 - onClick, 55
 - step, 56
 - text, 56
 - TextboxSpawnActor, 55

- teamusa::Timer, 56
 - ~Timer, 57
 - getTicks, 57
 - mPauseTicks, 57
 - mPaused, 57
 - mStartTicks, 57
 - mStarted, 57
 - pause, 57
 - start, 57
 - stop, 57
 - Timer, 57
 - unpause, 57
- teamusa::VideoActor, 58
 - ~VideoActor, 58
 - step, 58
 - VideoActor, 58
- teamusa::VideoEngine, 66
 - ~VideoEngine, 67
 - clearLayers, 68
 - coreResources, 69
 - deleteResourceGroup, 68
 - deleteTexture, 68
 - display, 68
 - hideTextbox, 68
 - isShowingTextbox, 68
 - layers, 69
 - levelResources, 69
 - loadTexture, 68
 - MAX_RESERVED_ID, 69
 - NUM_LAYERS, 69
 - render, 68
 - renderDebugBox, 69
 - renderRotate, 69
 - SHADOW_LAYER, 69
 - showTextbox, 69
 - swapFullscreen, 69
 - TEXT_LAYER, 69
 - textboxActive, 69
 - textboxPadding, 69
 - textboxRegion, 69
 - videoContext, 69
 - VideoEngine, 67
- teamusa::VideoEventActor, 70
 - ~VideoEventActor, 70
 - actorEvent, 72
 - onClick, 70
 - onHover, 72
 - step, 72
 - VideoEventActor, 70
- text
 - teamusa::TextboxSpawnActor, 56
- textboxActive
 - teamusa::VideoEngine, 69
- textboxPadding
 - teamusa::VideoEngine, 69
- textboxRegion
 - teamusa::VideoEngine, 69
- TextboxSpawnActor
 - teamusa::TextboxSpawnActor, 55
- TextboxSpawnActor.cpp, 88
- TextboxSpawnActor.h, 88
- texture_iter
 - mediawrap::VideoContext, 60
- TextureID
 - mediawrap::VideoContext, 60
 - teamusa, 13
- textureID
 - teamusa::ActorVideo, 16
- textureId
 - teamusa::DelayedVideoActor, 29
- textures
 - mediawrap::VideoContext, 64
- Timer
 - teamusa::Timer, 57
- Timer.cpp, 89
- Timer.h, 89
- transitionSteps
 - teamusa::MovingActor, 43
- type
 - teamusa::ActorEvent, 15
- unpause
 - teamusa::Timer, 57
- value
 - teamusa::ActorEvent, 15
- video_display
 - mediawrap::VideoContext, 64
- VideoActor
 - teamusa::VideoActor, 58
- VideoActor.cpp, 90
- VideoActor.h, 90
- VideoContext
 - mediawrap::VideoContext, 61
- videoContext
 - teamusa::VideoEngine, 69
- VideoContext.cpp, 90
- VideoContext.hpp, 90
- VideoDisplay
 - mediawrap::VideoDisplay, 65
- VideoDisplay.cpp, 91
- VideoDisplay.hpp, 91
- VideoEngine
 - teamusa::VideoEngine, 67
- VideoEngine.cpp, 92
- VideoEngine.hpp, 92
- VideoEventActor
 - teamusa::VideoEventActor, 70
- VideoEventActor.cpp, 92
- VideoEventActor.h, 93
- wGrowth
 - teamusa::MovingActor, 43
- window
 - mediawrap::VideoDisplay, 66
- x

teamusa::Point, [47](#)
xSpeed
teamusa::MovingActor, [43](#)

y
teamusa::Point, [47](#)
ySpeed
teamusa::MovingActor, [43](#)