



Delta3D Version 2.4.0

dtAnim::

Reference Manual

Contents

1	Main Page	1
2	Todo List	3
3	Directory Documentation	5
3.1	inc/dtAnim/ Directory Reference	5
3.2	src/dtAnim/ Directory Reference	6
3.3	inc/ Directory Reference	7
3.4	src/ Directory Reference	8
4	Namespace Documentation	9
4.1	dtAnim Namespace Reference	9
4.1.1	Detailed Description	11
4.1.2	Typedef Documentation	12
4.1.2.1	AnimationCallback	12
4.1.2.2	AttachmentPair	12
4.1.2.3	StringVector	12
4.1.3	Function Documentation	12
4.1.3.1	FindWithFunctor	12
4.1.3.2	GetCelestialCoordinates	12
4.1.3.3	GetCelestialDirection	12
4.1.3.4	GetClosestPointOnSegment	12
4.1.3.5	IsPointBetweenVectors	12
4.1.3.6	MapCelestialToScreen	13
4.2	dtCore Namespace Reference	14
4.3	dtDAL Namespace Reference	15
4.4	dtGame Namespace Reference	16
4.5	dtUtil Namespace Reference	17
5	Class Documentation	19
5.1	AnimActorRegistry Class Reference	19
5.1.1	Constructor & Destructor Documentation	19
5.1.1.1	AnimActorRegistry	19
5.1.2	Member Function Documentation	19
5.1.2.1	RegisterActorTypes	19
5.1.3	Member Data Documentation	19
5.1.3.1	ANIMATION_ACTOR_TYPE	19
5.1.3.2	CAL3D_ACTOR_TYPE	19

5.2	Animatable Class Reference	20
5.2.1	Detailed Description	21
5.2.2	Constructor & Destructor Documentation	21
5.2.2.1	Animatable	21
5.2.2.2	Animatable	21
5.2.2.3	~Animatable	21
5.2.3	Member Function Documentation	21
5.2.3.1	Clone	21
5.2.3.2	ForceFadeOut	21
5.2.3.3	GetBaseWeight	22
5.2.3.4	GetCurrentWeight	22
5.2.3.5	GetElapsedTime	22
5.2.3.6	GetEndTime	22
5.2.3.7	GetFadeIn	22
5.2.3.8	GetFadeOut	22
5.2.3.9	GetName	22
5.2.3.10	GetSpeed	22
5.2.3.11	GetStartDelay	22
5.2.3.12	GetStartTime	22
5.2.3.13	IsActive	22
5.2.3.14	operator=	22
5.2.3.15	Prune	22
5.2.3.16	Recalculate	22
5.2.3.17	SetActive	22
5.2.3.18	SetBaseWeight	22
5.2.3.19	SetCurrentWeight	22
5.2.3.20	SetElapsedTime	22
5.2.3.21	SetEndCallback	22
5.2.3.22	SetEndTime	23
5.2.3.23	SetFadeIn	23
5.2.3.24	SetFadeOut	23
5.2.3.25	SetName	23
5.2.3.26	SetPrune	23
5.2.3.27	SetSpeed	23
5.2.3.28	SetStartDelay	23
5.2.3.29	SetStartTime	23
5.2.3.30	ShouldPrune	23
5.2.3.31	Update	23
5.3	AnimatableStruct Struct Reference	24
5.3.1	Constructor & Destructor Documentation	24
5.3.1.1	AnimatableStruct	24

5.3.2	Member Data Documentation	24
5.3.2.1	mBaseWeight	24
5.3.2.2	mFadeIn	24
5.3.2.3	mFadeOut	24
5.3.2.4	mName	24
5.3.2.5	mSpeed	24
5.3.2.6	mStartDelay	24
5.4	AnimationChannel Class Reference	25
5.4.1	Detailed Description	26
5.4.2	Constructor & Destructor Documentation	26
5.4.2.1	AnimationChannel	26
5.4.2.2	AnimationChannel	26
5.4.2.3	~AnimationChannel	26
5.4.2.4	AnimationChannel	26
5.4.3	Member Function Documentation	26
5.4.3.1	Clone	26
5.4.3.2	ForceFadeOut	26
5.4.3.3	GetAnimation	26
5.4.3.4	GetAnimation	26
5.4.3.5	GetMaxDuration	26
5.4.3.6	GetModel	26
5.4.3.7	GetModel	27
5.4.3.8	IsAction	27
5.4.3.9	IsLooping	27
5.4.3.10	operator=	27
5.4.3.11	Prune	27
5.4.3.12	Recalculate	27
5.4.3.13	SetAction	27
5.4.3.14	SetAnimation	27
5.4.3.15	SetLooping	27
5.4.3.16	SetMaxDuration	27
5.4.3.17	SetModel	27
5.4.3.18	Update	27
5.5	AnimationChannelStruct Struct Reference	28
5.5.1	Constructor & Destructor Documentation	28
5.5.1.1	AnimationChannelStruct	28
5.5.2	Member Data Documentation	28
5.5.2.1	mAnimationName	28
5.5.2.2	mIsAction	28
5.5.2.3	mIsLooping	28
5.5.2.4	mMaxDuration	28

5.6	AnimationComponent Class Reference	29
5.6.1	Member Typedef Documentation	30
5.6.1.1	AnimComplter	30
5.6.1.2	AnimCompMap	30
5.6.1.3	AnimCompMapping	30
5.6.1.4	BaseClass	30
5.6.2	Constructor & Destructor Documentation	30
5.6.2.1	AnimationComponent	30
5.6.2.2	~AnimationComponent	30
5.6.3	Member Function Documentation	30
5.6.3.1	GetEyePointActor	30
5.6.3.2	GetEyePointActor	30
5.6.3.3	GetGroundClamper	30
5.6.3.4	GetGroundClamper	30
5.6.3.5	GetHelperForProxy	30
5.6.3.6	GetTerrainActor	30
5.6.3.7	GetTerrainActor	30
5.6.3.8	GroundClamp	30
5.6.3.9	IsRegisteredActor	30
5.6.3.10	ProcessMessage	30
5.6.3.11	RegisterActor	31
5.6.3.12	SetEyePointActor	31
5.6.3.13	SetGroundClamper	31
5.6.3.14	SetTerrainActor	31
5.6.3.15	TickLocal	31
5.6.3.16	UnregisterActor	31
5.6.4	Member Data Documentation	31
5.6.4.1	DEFAULT_NAME	31
5.7	AnimationController Class Reference	32
5.7.1	Detailed Description	32
5.7.2	Constructor & Destructor Documentation	32
5.7.2.1	AnimationController	32
5.7.2.2	AnimationController	32
5.7.2.3	~AnimationController	33
5.7.3	Member Function Documentation	33
5.7.3.1	Clone	33
5.7.3.2	GetParent	33
5.7.3.3	GetParent	33
5.7.3.4	operator=	33
5.7.3.5	Recalculate	33
5.7.3.6	SetComputeSpeed	33

5.7.3.7	SetComputeWeight	33
5.7.3.8	SetParent	33
5.7.3.9	Update	33
5.8	AnimationGameActor Class Reference	34
5.8.1	Detailed Description	34
5.8.2	Constructor & Destructor Documentation	34
5.8.2.1	AnimationGameActor	34
5.8.2.2	~AnimationGameActor	34
5.8.3	Member Function Documentation	34
5.8.3.1	GetHelper	34
5.8.3.2	GetHelper	34
5.8.3.3	SetModel	34
5.8.4	Member Data Documentation	34
5.8.4.1	mHelper	34
5.9	AnimationGameActorProxy Class Reference	35
5.9.1	Detailed Description	35
5.9.2	Constructor & Destructor Documentation	35
5.9.2.1	AnimationGameActorProxy	35
5.9.2.2	~AnimationGameActorProxy	35
5.9.3	Member Function Documentation	35
5.9.3.1	BuildPropertyMap	35
5.9.3.2	CreateActor	35
5.9.3.3	GetBillBoardIcon	35
5.9.3.4	GetRenderMode	36
5.10	AnimationHelper Class Reference	37
5.10.1	Detailed Description	38
5.10.2	Constructor & Destructor Documentation	38
5.10.2.1	AnimationHelper	38
5.10.2.2	~AnimationHelper	38
5.10.3	Member Function Documentation	38
5.10.3.1	ClearAll	38
5.10.3.2	ClearAnimation	38
5.10.3.3	GetActorProperties	38
5.10.3.4	GetAnimator	38
5.10.3.5	GetAnimator	38
5.10.3.6	GetAttachmentController	38
5.10.3.7	GetGroundClamp	38
5.10.3.8	GetModelWrapper	38
5.10.3.9	GetModelWrapper	38
5.10.3.10	GetNode	39
5.10.3.11	GetNode	39

5.10.3.12	GetSequenceMixer	39
5.10.3.13	GetSequenceMixer	39
5.10.3.14	LoadModel	39
5.10.3.15	PlayAnimation	39
5.10.3.16	SetAttachmentController	39
5.10.3.17	SetGroundClamp	39
5.10.3.18	Update	39
5.10.4	Member Data Documentation	39
5.10.4.1	PROPERTY_SKELETAL_MESH	39
5.11	AnimationSequence Class Reference	40
5.11.1	Detailed Description	41
5.11.2	Member Typedef Documentation	41
5.11.2.1	AnimationContainer	41
5.11.2.2	ContainerType	41
5.11.3	Constructor & Destructor Documentation	41
5.11.3.1	AnimationSequence	41
5.11.3.2	AnimationSequence	41
5.11.3.3	~AnimationSequence	41
5.11.3.4	AnimationSequence	41
5.11.4	Member Function Documentation	41
5.11.4.1	AddAnimation	41
5.11.4.2	ClearAnimation	41
5.11.4.3	Clone	41
5.11.4.4	ForceFadeOut	42
5.11.4.5	GetAnimation	42
5.11.4.6	GetAnimation	42
5.11.4.7	GetChildAnimations	42
5.11.4.8	GetChildAnimations	42
5.11.4.9	GetController	42
5.11.4.10	GetController	42
5.11.4.11	IsAnimationPlaying	42
5.11.4.12	operator=	42
5.11.4.13	Prune	42
5.11.4.14	Recalculate	42
5.11.4.15	SetController	42
5.11.4.16	Update	42
5.12	AnimationSequenceStruct Struct Reference	43
5.12.1	Constructor & Destructor Documentation	43
5.12.1.1	AnimationSequenceStruct	43
5.12.2	Member Data Documentation	43
5.12.2.1	mChildNames	43

5.13	AnimationStruct Struct Reference	44
5.13.1	Detailed Description	44
5.13.2	Member Data Documentation	44
5.13.2.1	mFileName	44
5.13.2.2	mName	44
5.14	AnimationWrapper Class Reference	45
5.14.1	Detailed Description	45
5.14.2	Constructor & Destructor Documentation	45
5.14.2.1	AnimationWrapper	45
5.14.2.2	~AnimationWrapper	45
5.14.3	Member Function Documentation	45
5.14.3.1	GetDuration	45
5.14.3.2	GetID	45
5.14.3.3	GetName	45
5.14.3.4	GetSpeed	45
5.14.3.5	SetDuration	46
5.14.3.6	SetName	46
5.14.3.7	SetSpeed	46
5.15	AnimDriver Class Reference	47
5.15.1	Constructor & Destructor Documentation	47
5.15.1.1	AnimDriver	47
5.15.1.2	~AnimDriver	47
5.15.2	Member Function Documentation	47
5.15.2.1	SetWrapper	47
5.15.2.2	Update	47
5.16	AnimNodeBuilder Class Reference	48
5.16.1	Detailed Description	48
5.16.2	Member Typedef Documentation	49
5.16.2.1	CreateFunc	49
5.16.3	Constructor & Destructor Documentation	49
5.16.3.1	AnimNodeBuilder	49
5.16.3.2	AnimNodeBuilder	49
5.16.3.3	~AnimNodeBuilder	49
5.16.3.4	AnimNodeBuilder	49
5.16.4	Member Function Documentation	49
5.16.4.1	CreateHardware	49
5.16.4.2	CreateNode	49
5.16.4.3	CreateNULL	49
5.16.4.4	CreateSoftware	49
5.16.4.5	CreateSoftwareInternal	49
5.16.4.6	CreateSoftwareNoVBO	49

5.16.4.7	GetCreate	49
5.16.4.8	LoadShaders	49
5.16.4.9	operator=	49
5.16.4.10	SetCreate	49
5.16.4.11	SupportsHardware	49
5.16.4.12	SupportsSoftware	50
5.17	AnimSeqForceFade Class Reference	51
5.17.1	Constructor & Destructor Documentation	51
5.17.1.1	AnimSeqForceFade	51
5.17.2	Member Function Documentation	51
5.17.2.1	operator()	51
5.18	AnimSequenceUpdater Struct Reference	52
5.18.1	Constructor & Destructor Documentation	52
5.18.1.1	AnimSequenceUpdater	52
5.18.2	Member Function Documentation	52
5.18.2.1	operator()	52
5.19	AttachmentController Class Reference	53
5.19.1	Detailed Description	53
5.19.2	Member Typedef Documentation	53
5.19.2.1	AttachmentContainer	53
5.19.3	Constructor & Destructor Documentation	53
5.19.3.1	AttachmentController	53
5.19.3.2	~AttachmentController	53
5.19.4	Member Function Documentation	53
5.19.4.1	AddAttachment	53
5.19.4.2	GetAttachments	53
5.19.4.3	RemoveAttachment	53
5.19.4.4	Update	54
5.20	AttachmentMover Class Reference	55
5.20.1	Detailed Description	55
5.20.2	Constructor & Destructor Documentation	55
5.20.2.1	AttachmentMover	55
5.20.2.2	AttachmentMover	55
5.20.3	Member Function Documentation	55
5.20.3.1	operator()	55
5.20.3.2	operator()	55
5.20.3.3	operator=	55
5.21	BaseReferenceBlend Struct Reference	56
5.21.1	Member Data Documentation	56
5.21.1.1	blendAlpha	56
5.21.1.2	endAnimID	56

5.21.1.3	endDirection	56
5.21.1.4	startAnimID	56
5.21.1.5	startDirection	56
5.22	Cal3DAnimator Class Reference	57
5.22.1	Constructor & Destructor Documentation	58
5.22.1.1	Cal3DAnimator	58
5.22.1.2	~Cal3DAnimator	58
5.22.2	Member Function Documentation	58
5.22.2.1	GetAnimationDriver	58
5.22.2.2	GetMorphTargetDriver	58
5.22.2.3	GetPhysiqueDriver	58
5.22.2.4	GetPostDriver	58
5.22.2.5	GetPreDriver	58
5.22.2.6	GetSkeletonDriver	58
5.22.2.7	GetSpringDriver	58
5.22.2.8	GetWrapper	58
5.22.2.9	GetWrapper	58
5.22.2.10	SetAnimationDriver	58
5.22.2.11	SetMorphTargetDriver	58
5.22.2.12	SetPhysiqueDriver	58
5.22.2.13	SetPostDriver	58
5.22.2.14	SetPreDriver	58
5.22.2.15	SetSkeletonDriver	58
5.22.2.16	SetSpringDriver	58
5.22.2.17	SetWrapper	58
5.22.2.18	Update	58
5.23	Cal3DBoundingBoxCalculator Class Reference	59
5.23.1	Constructor & Destructor Documentation	59
5.23.1.1	Cal3DBoundingBoxCalculator	59
5.23.2	Member Function Documentation	59
5.23.2.1	computeBound	59
5.24	Cal3DDatabase Class Reference	60
5.24.1	Member Typedef Documentation	61
5.24.1.1	ModelDataArray	61
5.24.2	Constructor & Destructor Documentation	61
5.24.2.1	Cal3DDatabase	61
5.24.2.2	~Cal3DDatabase	61
5.24.3	Member Function Documentation	61
5.24.3.1	Find	61
5.24.3.2	Find	61
5.24.3.3	Find	61

5.24.3.4	Find	61
5.24.3.5	GetInstance	61
5.24.3.6	GetModelData	61
5.24.3.7	GetModelData	61
5.24.3.8	GetNodeBuilder	61
5.24.3.9	Load	61
5.24.3.10	PurgeLoaderCaches	61
5.24.3.11	TruncateDatabase	61
5.24.4	Member Data Documentation	61
5.24.4.1	mFileLoader	61
5.24.4.2	mInstance	61
5.24.4.3	mModelData	61
5.24.4.4	mNodeBuilder	61
5.25	Cal3DGameActor Class Reference	62
5.25.1	Detailed Description	63
5.25.2	Member Typedef Documentation	63
5.25.2.1	RenderModeBitContainer	63
5.25.3	Member Enumeration Documentation	63
5.25.3.1	RenderModeBits	63
5.25.4	Constructor & Destructor Documentation	63
5.25.4.1	Cal3DGameActor	63
5.25.4.2	~Cal3DGameActor	63
5.25.5	Member Function Documentation	63
5.25.5.1	AddedToScene	63
5.25.5.2	ApplyAnimationGroup	63
5.25.5.3	GetAnimator	63
5.25.5.4	GetAnimator	63
5.25.5.5	GetRenderMode	63
5.25.5.6	MakeAnimationGroup	63
5.25.5.7	OnEnteredWorld	63
5.25.5.8	OnTickLocal	63
5.25.5.9	SetModel	63
5.25.5.10	SetRenderMode	64
5.25.6	Member Data Documentation	64
5.25.6.1	mAnimator	64
5.25.6.2	mModelGeode	64
5.25.6.3	mModelLoader	64
5.25.6.4	mRenderModeBits	64
5.25.6.5	mSkeletalGeode	64
5.26	Cal3DGameActorProxy Class Reference	65
5.26.1	Detailed Description	65

5.26.2	Constructor & Destructor Documentation	65
5.26.2.1	Cal3DGameActorProxy	65
5.26.2.2	~Cal3DGameActorProxy	65
5.26.3	Member Function Documentation	65
5.26.3.1	BuildInvokables	65
5.26.3.2	BuildPropertyMap	65
5.26.3.3	CreateActor	66
5.26.3.4	GetBillBoardIcon	66
5.26.3.5	GetRenderMode	66
5.27	Cal3DLoader Class Reference	67
5.27.1	Detailed Description	67
5.27.2	Constructor & Destructor Documentation	67
5.27.2.1	Cal3DLoader	67
5.27.2.2	~Cal3DLoader	67
5.27.3	Member Function Documentation	67
5.27.3.1	Load	67
5.27.3.2	PurgeAllCaches	67
5.28	Cal3DModelData Class Reference	68
5.28.1	Member Typedef Documentation	69
5.28.1.1	AnimatableArray	69
5.28.1.2	AnimationWrapperArray	69
5.28.2	Constructor & Destructor Documentation	69
5.28.2.1	Cal3DModelData	69
5.28.2.2	~Cal3DModelData	69
5.28.2.3	Cal3DModelData	69
5.28.3	Member Function Documentation	69
5.28.3.1	Add	69
5.28.3.2	Add	69
5.28.3.3	GetAnimatables	69
5.28.3.4	GetAnimatables	69
5.28.3.5	GetAnimationWrappers	69
5.28.3.6	GetAnimationWrappers	69
5.28.3.7	GetCoreModel	69
5.28.3.8	GetCoreModel	69
5.28.3.9	GetFilename	69
5.28.3.10	GetIndexVBO	69
5.28.3.11	GetLODOptions	69
5.28.3.12	GetLODOptions	69
5.28.3.13	GetPoseMeshFilename	69
5.28.3.14	GetShaderGroupName	69
5.28.3.15	GetShaderMaxBones	69

5.28.3.16	GetShaderName	69
5.28.3.17	GetVertexVBO	69
5.28.3.18	operator=	69
5.28.3.19	Remove	69
5.28.3.20	Remove	69
5.28.3.21	SetIndexVBO	69
5.28.3.22	SetPoseMeshFilename	70
5.28.3.23	SetShaderGroupName	70
5.28.3.24	SetShaderMaxBones	70
5.28.3.25	SetShaderName	70
5.28.3.26	SetVertexVBO	70
5.29	Cal3DModelWrapper Class Reference	71
5.29.1	Detailed Description	73
5.29.2	Constructor & Destructor Documentation	74
5.29.2.1	Cal3DModelWrapper	74
5.29.2.2	~Cal3DModelWrapper	74
5.29.3	Member Function Documentation	74
5.29.3.1	ApplyCoreModelScaleFactor	74
5.29.3.2	AttachMesh	74
5.29.3.3	BeginRenderingQuery	74
5.29.3.4	BlendCycle	74
5.29.3.5	ClearAll	74
5.29.3.6	ClearCycle	74
5.29.3.7	DetachMesh	74
5.29.3.8	EndRenderingQuery	74
5.29.3.9	ExecuteAction	74
5.29.3.10	GetAmbientColor	74
5.29.3.11	GetAnimationTime	74
5.29.3.12	GetBoneAbsoluteRotation	75
5.29.3.13	GetBoneAbsoluteRotationForKeyFrame	75
5.29.3.14	GetBoneAbsoluteTranslation	75
5.29.3.15	GetBoneRelativeRotation	75
5.29.3.16	GetBoundingBox	75
5.29.3.17	GetCalModel	75
5.29.3.18	GetCalModel	75
5.29.3.19	GetCoreAnimationCount	75
5.29.3.20	GetCoreAnimationDuration	75
5.29.3.21	GetCoreAnimationIDByName	75
5.29.3.22	GetCoreAnimationKeyframeCount	75
5.29.3.23	GetCoreAnimationKeyframeCountForTrack	75
5.29.3.24	GetCoreAnimationName	76

5.29.3.25	GetCoreAnimationTrackCount	76
5.29.3.26	GetCoreBoneChildrenIDs	76
5.29.3.27	GetCoreBoneID	76
5.29.3.28	GetCoreBoneNames	76
5.29.3.29	GetCoreMaterial	76
5.29.3.30	GetCoreMaterialAmbient	76
5.29.3.31	GetCoreMaterialCount	76
5.29.3.32	GetCoreMaterialDiffuse	76
5.29.3.33	GetCoreMaterialName	76
5.29.3.34	GetCoreMaterialShininess	76
5.29.3.35	GetCoreMaterialSpecular	76
5.29.3.36	GetCoreMeshCount	76
5.29.3.37	GetCoreMeshName	76
5.29.3.38	GetCoreTrackKeyFrameQuat	76
5.29.3.39	GetDiffuseColor	77
5.29.3.40	GetFaceCount	77
5.29.3.41	GetFaces	77
5.29.3.42	GetMapCount	77
5.29.3.43	GetMapUserData	77
5.29.3.44	GetMeshCount	77
5.29.3.45	GetNormals	77
5.29.3.46	GetOrCreateCalHardwareModel	77
5.29.3.47	GetParentBoneID	77
5.29.3.48	GetRootBoneIDs	77
5.29.3.49	GetShininess	77
5.29.3.50	GetSpecularColor	77
5.29.3.51	GetSubmeshCount	77
5.29.3.52	GetTextureCoords	77
5.29.3.53	GetVertexCount	77
5.29.3.54	GetVertices	77
5.29.3.55	HasAnimation	77
5.29.3.56	HasBone	77
5.29.3.57	HasTrackForBone	77
5.29.3.58	HideMesh	77
5.29.3.59	IsMeshVisible	77
5.29.3.60	RemoveAction	77
5.29.3.61	SelectMeshSubmesh	77
5.29.3.62	SetAnimationTime	77
5.29.3.63	SetCalModel	78
5.29.3.64	SetLODLevel	78
5.29.3.65	SetMaterialSet	78

5.29.3.66	ShowMesh	78
5.29.3.67	Update	78
5.29.3.68	UpdateAnimation	78
5.29.3.69	UpdateMorphTargetMixer	78
5.29.3.70	UpdatePhysique	78
5.29.3.71	UpdateSkeleton	78
5.29.3.72	UpdateSpringSystem	78
5.29.4	Member Data Documentation	78
5.29.4.1	NULL_BONE	78
5.30	CharacterFileHandler Class Reference	79
5.30.1	Detailed Description	81
5.30.2	Constructor & Destructor Documentation	82
5.30.2.1	CharacterFileHandler	82
5.30.2.2	~CharacterFileHandler	82
5.30.3	Member Function Documentation	82
5.30.3.1	characters	82
5.30.3.2	endDocument	82
5.30.3.3	endElement	82
5.30.3.4	endPrefixMapping	82
5.30.3.5	ignorableWhitespace	82
5.30.3.6	processingInstruction	82
5.30.3.7	setDocumentLocator	82
5.30.3.8	skippedEntity	82
5.30.3.9	startDocument	82
5.30.3.10	startElement	82
5.30.3.11	startPrefixMapping	82
5.30.4	Member Data Documentation	82
5.30.4.1	ANIMATION_ELEMENT	82
5.30.4.2	ANIMATION_NAME_ELEMENT	82
5.30.4.3	BASE_WEIGHT_ELEMENT	82
5.30.4.4	CHANNEL_ELEMENT	82
5.30.4.5	CHARACTER_ELEMENT	82
5.30.4.6	CHARACTER_XML_LOGGER	82
5.30.4.7	CHILD_ELEMENT	82
5.30.4.8	FADE_IN_ELEMENT	82
5.30.4.9	FADE_OUT_ELEMENT	82
5.30.4.10	FILENAME_ELEMENT	82
5.30.4.11	IS_ACTION_ELEMENT	82
5.30.4.12	IS_LOOPING_ELEMENT	82
5.30.4.13	LOD_ELEMENT	82
5.30.4.14	LOD_END_DISTANCE_ELEMENT	82

5.30.4.15	LOD_START_DISTANCE_ELEMENT	82
5.30.4.16	mAnimationChannels	82
5.30.4.17	mAnimations	82
5.30.4.18	mAnimationSequences	83
5.30.4.19	MATERIAL_ELEMENT	83
5.30.4.20	MAX_DURATION_ELEMENT	83
5.30.4.21	MAX_VISIBLE_DISTANCE_ELEMENT	83
5.30.4.22	MESH_ELEMENT	83
5.30.4.23	mFoundLODOptions	83
5.30.4.24	mFoundScale	83
5.30.4.25	mLODEndDistance	83
5.30.4.26	mLODMaxVisibleDistance	83
5.30.4.27	mLODStartDistance	83
5.30.4.28	mMaterials	83
5.30.4.29	mMeshes	83
5.30.4.30	mMorphAnimations	83
5.30.4.31	mName	83
5.30.4.32	MORPH_ANIMATION_ELEMENT	83
5.30.4.33	mPoseMeshFilename	83
5.30.4.34	mScale	83
5.30.4.35	mShaderGroup	83
5.30.4.36	mShaderMaxBones	83
5.30.4.37	mShaderName	83
5.30.4.38	mSkeletonFilename	83
5.30.4.39	NAME_ELEMENT	84
5.30.4.40	POSEMESH_ELEMENT	84
5.30.4.41	SCALE_ELEMENT	84
5.30.4.42	SCALE_FACTOR_ELEMENT	84
5.30.4.43	SEQUENCE_ELEMENT	84
5.30.4.44	SHADER_GROUP_ELEMENT	84
5.30.4.45	SHADER_MAX_BONES_ELEMENT	84
5.30.4.46	SHADER_NAME_ELEMENT	84
5.30.4.47	SKELETON_ELEMENT	84
5.30.4.48	SKINNING_SHADER_ELEMENT	84
5.30.4.49	SPEED_ELEMENT	84
5.30.4.50	START_DELAY_ELEMENT	84
5.31	CharacterWrapper Class Reference	85
5.31.1	Detailed Description	86
5.31.2	Member Typedef Documentation	86
5.31.2.1	BaseClass	86
5.31.3	Constructor & Destructor Documentation	86

5.31.3.1	CharacterWrapper	86
5.31.3.2	~CharacterWrapper	86
5.31.4	Member Function Documentation	86
5.31.4.1	ClearAllAnimations	86
5.31.4.2	ClearAnimation	86
5.31.4.3	GetAnimationHelper	86
5.31.4.4	GetAnimationHelper	87
5.31.4.5	GetHeading	87
5.31.4.6	GetHeightAboveGround	87
5.31.4.7	GetLocalOffset	87
5.31.4.8	GetRotationSpeed	87
5.31.4.9	GetSpeed	87
5.31.4.10	IsAnimationPlaying	87
5.31.4.11	PlayAnimation	87
5.31.4.12	RotateToHeading	87
5.31.4.13	RotateToPoint	87
5.31.4.14	SetGroundClamp	88
5.31.4.15	SetGroundClamp	88
5.31.4.16	SetHeading	88
5.31.4.17	SetHeightAboveGround	88
5.31.4.18	SetLocalOffset	88
5.31.4.19	SetRotationSpeed	88
5.31.4.20	SetSpeed	88
5.31.4.21	Update	88
5.32	CharDrawable Class Reference	89
5.32.1	Detailed Description	89
5.32.2	Constructor & Destructor Documentation	89
5.32.2.1	CharDrawable	89
5.32.2.2	~CharDrawable	89
5.32.3	Member Function Documentation	89
5.32.3.1	GetCal3DWrapper	89
5.32.3.2	GetNode	89
5.32.3.3	OnMessage	89
5.32.3.4	RebuildSubmeshes	89
5.32.3.5	SetCal3DWrapper	90
5.32.4	Member Data Documentation	90
5.32.4.1	mAnimator	90
5.32.4.2	mLastMeshCount	90
5.32.4.3	mNode	90
5.33	CloneFunctor Class Reference	91
5.33.1	Constructor & Destructor Documentation	91

5.33.1.1	CloneFunctor	91
5.33.2	Member Function Documentation	91
5.33.2.1	operator()	91
5.34	CreateGeometryDrawCallback Class Reference	92
5.34.1	Detailed Description	92
5.34.2	Constructor & Destructor Documentation	92
5.34.2.1	CreateGeometryDrawCallback	92
5.34.2.2	~CreateGeometryDrawCallback	92
5.34.3	Member Function Documentation	92
5.34.3.1	drawImplementation	92
5.34.4	Member Data Documentation	92
5.34.4.1	mCreatedNode	92
5.35	DeletePointer< PtrT > Struct Template Reference	93
5.35.1	Member Function Documentation	93
5.35.1.1	operator()	93
5.36	findWithCoreModel Struct Reference	94
5.36.1	Constructor & Destructor Documentation	94
5.36.1.1	findWithCoreModel	94
5.36.2	Member Function Documentation	94
5.36.2.1	operator()	94
5.36.3	Member Data Documentation	94
5.36.3.1	mModel	94
5.37	findWithFilename Struct Reference	95
5.37.1	Constructor & Destructor Documentation	95
5.37.1.1	findWithFilename	95
5.37.2	Member Function Documentation	95
5.37.2.1	operator()	95
5.37.3	Member Data Documentation	95
5.37.3.1	mFilename	95
5.38	HardwareSubmeshCallback Class Reference	96
5.38.1	Constructor & Destructor Documentation	96
5.38.1.1	HardwareSubmeshCallback	96
5.38.2	Member Function Documentation	96
5.38.2.1	update	96
5.39	HardwareSubmeshComputeBound Class Reference	97
5.39.1	Constructor & Destructor Documentation	97
5.39.1.1	HardwareSubmeshComputeBound	97
5.39.2	Member Function Documentation	97
5.39.2.1	computeBound	97
5.39.3	Member Data Documentation	97
5.39.3.1	mDefaultBox	97

5.40	HardwareSubmeshDrawable Class Reference	98
5.40.1	Constructor & Destructor Documentation	98
5.40.1.1	HardwareSubmeshDrawable	98
5.40.1.2	~HardwareSubmeshDrawable	98
5.40.2	Member Function Documentation	98
5.40.2.1	clone	98
5.40.2.2	cloneType	98
5.40.2.3	drawImplementation	98
5.40.2.4	SetBoundingBox	98
5.41	HotSpotDriver Class Reference	99
5.41.1	Detailed Description	99
5.41.2	Member Typedef Documentation	99
5.41.2.1	HotSpotContainer	99
5.41.3	Constructor & Destructor Documentation	99
5.41.3.1	HotSpotDriver	99
5.41.3.2	~HotSpotDriver	99
5.41.4	Member Function Documentation	99
5.41.4.1	AddHotSpot	99
5.41.4.2	GetHotSpots	99
5.41.4.3	RemoveHotSpot	99
5.41.4.4	SetWrapper	99
5.41.4.5	Update	99
5.42	ICal3DDriver Class Reference	100
5.42.1	Member Function Documentation	100
5.42.1.1	SetWrapper	100
5.42.1.2	Update	100
5.43	IsActor Class Reference	101
5.43.1	Constructor & Destructor Documentation	101
5.43.1.1	IsActor	101
5.43.2	Member Function Documentation	101
5.43.2.1	operator()	101
5.44	LODOptions Class Reference	102
5.44.1	Detailed Description	102
5.44.2	Constructor & Destructor Documentation	102
5.44.2.1	LODOptions	102
5.44.3	Member Function Documentation	102
5.44.3.1	GetEndDistance	102
5.44.3.2	GetMaxVisibleDistance	102
5.44.3.3	GetStartDistance	102
5.44.3.4	SetEndDistance	102
5.44.3.5	SetMaxVisibleDistance	102

5.44.3.6	SetStartDistance	102
5.45	MaterialStruct Struct Reference	103
5.45.1	Member Data Documentation	103
5.45.1.1	mFileName	103
5.45.1.2	mName	103
5.46	MeshStruct Struct Reference	104
5.46.1	Detailed Description	104
5.46.2	Member Data Documentation	104
5.46.2.1	mFileName	104
5.46.2.2	mName	104
5.47	MorphAnimationStruct Struct Reference	105
5.47.1	Detailed Description	105
5.47.2	Member Data Documentation	105
5.47.2.1	mFileName	105
5.47.2.2	mName	105
5.48	MorphDriver Class Reference	106
5.48.1	Constructor & Destructor Documentation	106
5.48.1.1	MorphDriver	106
5.48.1.2	~MorphDriver	106
5.48.2	Member Function Documentation	106
5.48.2.1	SetWrapper	106
5.48.2.2	Update	106
5.49	PhysiqueDriver Class Reference	107
5.49.1	Constructor & Destructor Documentation	107
5.49.1.1	PhysiqueDriver	107
5.49.1.2	~PhysiqueDriver	107
5.49.2	Member Function Documentation	107
5.49.2.1	SetWrapper	107
5.49.2.2	Update	107
5.50	PoseBuilderFunctor< ContainerT > Struct Template Reference	108
5.50.1	Constructor & Destructor Documentation	108
5.50.1.1	PoseBuilderFunctor	108
5.50.2	Member Function Documentation	108
5.50.2.1	operator()	108
5.51	PoseMesh Class Reference	109
5.51.1	Member Typedef Documentation	110
5.51.1.1	Barycentric2D	110
5.51.1.2	Barycentric2DVector	110
5.51.1.3	EdgeLineMap	110
5.51.1.4	MeshIndexPair	110
5.51.1.5	StringVector	110

5.51.1.6	TriangleEdgeVector	110
5.51.1.7	TriangleVector	110
5.51.1.8	VertexVector	110
5.51.2	Constructor & Destructor Documentation	110
5.51.2.1	PoseMesh	110
5.51.2.2	~PoseMesh	110
5.51.3	Member Function Documentation	110
5.51.3.1	FindPoseTriangleID	110
5.51.3.2	GetBarySpaces	110
5.51.3.3	GetEffectorForwardAxis	110
5.51.3.4	GetEffectorID	110
5.51.3.5	GetEffectorName	110
5.51.3.6	GetIndexPairsForTriangle	110
5.51.3.7	GetName	110
5.51.3.8	GetRootForwardAxis	110
5.51.3.9	GetSilhouette	110
5.51.3.10	GetTargetTriangleData	110
5.51.3.11	GetTriangles	111
5.51.3.12	GetVertices	111
5.52	PoseMeshData Struct Reference	112
5.52.1	Member Data Documentation	112
5.52.1.1	mAnimations	112
5.52.1.2	mEffectorForward	112
5.52.1.3	mEffectorName	112
5.52.1.4	mName	112
5.52.1.5	mRootForward	112
5.52.1.6	mRootName	112
5.53	PoseMeshDatabase Class Reference	113
5.53.1	Detailed Description	113
5.53.2	Member Typedef Documentation	113
5.53.2.1	PoseMeshList	113
5.53.3	Constructor & Destructor Documentation	113
5.53.3.1	PoseMeshDatabase	113
5.53.3.2	~PoseMeshDatabase	113
5.53.4	Member Function Documentation	113
5.53.4.1	GetMeshes	113
5.53.4.2	GetPoseMeshByName	113
5.53.4.3	LoadFromFile	113
5.54	PoseMeshFileHandler Class Reference	114
5.54.1	Member Typedef Documentation	114
5.54.1.1	NodeStack	114

5.54.1.2	PoseMeshDataVector	114
5.54.2	Member Enumeration Documentation	114
5.54.2.1	PoseNode	114
5.54.3	Constructor & Destructor Documentation	115
5.54.3.1	PoseMeshFileHandler	115
5.54.3.2	~PoseMeshFileHandler	115
5.54.4	Member Function Documentation	115
5.54.4.1	characters	115
5.54.4.2	endDocument	115
5.54.4.3	endElement	115
5.54.4.4	endPrefixMapping	115
5.54.4.5	GetData	115
5.54.4.6	ignorableWhitespace	115
5.54.4.7	processingInstruction	115
5.54.4.8	setDocumentLocator	115
5.54.4.9	skippedEntity	115
5.54.4.10	startDocument	115
5.54.4.11	startElement	115
5.54.4.12	startPrefixMapping	115
5.54.5	Member Data Documentation	115
5.54.5.1	ANIMATION_NODE	115
5.54.5.2	DEFAULT_VALUE	115
5.54.5.3	EFFECTOR_ATTRIBUTE	115
5.54.5.4	EFFECTOR_FORWARD_ATTRIBUTE	115
5.54.5.5	NAME_ATTRIBUTE	115
5.54.5.6	POSE_NODE	115
5.54.5.7	ROOT_ATTRIBUTE	115
5.54.5.8	ROOT_FORWARD_ATTRIBUTE	115
5.54.5.9	TRIANGLE_NODE	115
5.55	PoseMeshLoader Class Reference	116
5.55.1	Member Typedef Documentation	116
5.55.1.1	MeshDataContainer	116
5.55.2	Constructor & Destructor Documentation	116
5.55.2.1	PoseMeshLoader	116
5.55.2.2	~PoseMeshLoader	116
5.55.3	Member Function Documentation	116
5.55.3.1	Load	116
5.56	PoseMeshUtility Class Reference	117
5.56.1	Detailed Description	117
5.56.2	Member Typedef Documentation	117
5.56.2.1	BaseReferencePose	117

5.56.3	Constructor & Destructor Documentation	117
5.56.3.1	PoseMeshUtility	117
5.56.3.2	~PoseMeshUtility	117
5.56.4	Member Function Documentation	117
5.56.4.1	BlendPoses	117
5.56.4.2	ClearPoses	117
5.56.4.3	GetBaseReferenceBlend	118
5.56.4.4	SetBaseReferencePoses	118
5.57	PredicatePoseMeshName Struct Reference	119
5.57.1	Constructor & Destructor Documentation	119
5.57.1.1	PredicatePoseMeshName	119
5.57.1.2	PredicatePoseMeshName	119
5.57.2	Member Function Documentation	119
5.57.2.1	operator()	119
5.57.3	Member Data Documentation	119
5.57.3.1	mName	119
5.58	PropertyNames Struct Reference	120
5.58.1	Detailed Description	120
5.58.2	Member Data Documentation	120
5.58.2.1	ANIMATION_BLEND_DELAY	120
5.58.2.2	ANIMATION_BLEND_GROUP	120
5.58.2.3	ANIMATION_BLEND_ID	120
5.58.2.4	ANIMATION_BLEND_WEIGHT	120
5.58.2.5	ANIMATION_GROUP	120
5.58.2.6	ANIMATION_GROUP_LABEL	120
5.58.2.7	RENDER_MODE	120
5.58.2.8	RENDER_MODE_LABEL	120
5.59	RecalcFunctor Class Reference	121
5.59.1	Constructor & Destructor Documentation	121
5.59.1.1	RecalcFunctor	121
5.59.2	Member Function Documentation	121
5.59.2.1	GetEnd	121
5.59.2.2	operator()	121
5.60	RenderPrimitive Struct Reference	122
5.60.1	Detailed Description	122
5.60.2	Constructor & Destructor Documentation	122
5.60.2.1	RenderPrimitive	122
5.60.2.2	RenderPrimitive	122
5.60.3	Member Function Documentation	122
5.60.3.1	operator()	122
5.60.4	Member Data Documentation	122

5.60.4.1	mModelWrapper	122
5.61	SequenceMixer Class Reference	123
5.61.1	Detailed Description	124
5.61.2	Member Typedef Documentation	124
5.61.2.1	AnimationTable	124
5.61.2.2	TableKey	124
5.61.3	Constructor & Destructor Documentation	124
5.61.3.1	SequenceMixer	124
5.61.3.2	~SequenceMixer	124
5.61.4	Member Function Documentation	124
5.61.4.1	ClearActiveAnimations	124
5.61.4.2	ClearAnimation	124
5.61.4.3	ClearRegisteredAnimations	124
5.61.4.4	ForceRecalculate	124
5.61.4.5	GetActiveAnimation	124
5.61.4.6	GetActiveAnimation	124
5.61.4.7	GetRegisteredAnimation	124
5.61.4.8	GetRegisteredAnimations	125
5.61.4.9	IsAnimationPlaying	125
5.61.4.10	PlayAnimation	125
5.61.4.11	RegisterAnimation	125
5.61.4.12	RemoveRegisteredAnimation	125
5.61.4.13	Update	125
5.62	SkeletalDrawable Class Reference	126
5.62.1	Detailed Description	126
5.62.2	Constructor & Destructor Documentation	126
5.62.2.1	SkeletalDrawable	126
5.62.2.2	~SkeletalDrawable	126
5.62.3	Member Function Documentation	126
5.62.3.1	clone	126
5.62.3.2	cloneType	126
5.62.3.3	drawImplementation	126
5.63	SkeletonDriver Class Reference	127
5.63.1	Constructor & Destructor Documentation	127
5.63.1.1	SkeletonDriver	127
5.63.1.2	~SkeletonDriver	127
5.63.2	Member Function Documentation	127
5.63.2.1	SetWrapper	127
5.63.2.2	Update	127
5.64	SpringDriver Class Reference	128
5.64.1	Constructor & Destructor Documentation	128

5.64.1.1	SpringDriver	128
5.64.1.2	~SpringDriver	128
5.64.2	Member Function Documentation	128
5.64.2.1	SetWrapper	128
5.64.2.2	Update	128
5.65	SubmeshComputeBound Class Reference	129
5.65.1	Constructor & Destructor Documentation	129
5.65.1.1	SubmeshComputeBound	129
5.65.2	Member Function Documentation	129
5.65.2.1	computeBound	129
5.65.3	Member Data Documentation	129
5.65.3.1	mBoundingBox	129
5.66	SubmeshCullCallback Class Reference	130
5.66.1	Constructor & Destructor Documentation	130
5.66.1.1	SubmeshCullCallback	130
5.66.2	Member Function Documentation	130
5.66.2.1	cull	130
5.67	SubmeshDirtyCallback Class Reference	131
5.67.1	Member Function Documentation	131
5.67.1.1	update	131
5.68	SubmeshDrawable Class Reference	132
5.68.1	Detailed Description	132
5.68.2	Constructor & Destructor Documentation	132
5.68.2.1	SubmeshDrawable	132
5.68.2.2	~SubmeshDrawable	132
5.68.3	Member Function Documentation	132
5.68.3.1	accept	132
5.68.3.2	clone	133
5.68.3.3	cloneType	133
5.68.3.4	drawImplementation	133
5.68.3.5	GetCurrentLOD	133
5.68.3.6	GetLODOptions	133
5.68.3.7	GetLODOptions	133
5.68.3.8	GetMeshID	133
5.68.3.9	GetModelWrapper	133
5.68.3.10	GetModelWrapper	133
5.68.3.11	GetSubmeshID	133
5.68.3.12	InitVertexBuffers	133
5.68.3.13	SetBoundingBox	133
5.68.3.14	SetCurrentLOD	133
5.68.3.15	supports	133

5.68.4	Member Data Documentation	133
5.68.4.1	LOD_COUNT	133
5.69	SubmeshUserData Class Reference	134
5.69.1	Member Typedef Documentation	134
5.69.1.1	BaseClass	134
5.69.2	Member Function Documentation	134
5.69.2.1	className	134
5.69.2.2	clone	134
5.69.2.3	cloneType	134
5.69.2.4	isSameKindAs	134
5.69.2.5	libraryName	134
5.69.3	Member Data Documentation	134
5.69.3.1	mLOD	134
5.70	TargetTriangle Struct Reference	135
5.70.1	Member Data Documentation	135
5.70.1.1	mAzimuth	135
5.70.1.2	mElevation	135
5.70.1.3	mIsInside	135
5.70.1.4	mTriangleID	135
5.71	Triangle Struct Reference	136
5.71.1	Constructor & Destructor Documentation	136
5.71.1.1	Triangle	136
5.71.2	Member Data Documentation	136
5.71.2.1	mIndices	136
5.71.2.2	mVertices	136
5.72	TriangleEdge Struct Reference	137
5.72.1	Constructor & Destructor Documentation	137
5.72.1.1	TriangleEdge	137
5.72.1.2	TriangleEdge	137
5.72.2	Member Data Documentation	137
5.72.2.1	mEdge	137
5.72.2.2	mTriangleID	137
5.73	UpdateCallback Class Reference	138
5.73.1	Detailed Description	138
5.73.2	Constructor & Destructor Documentation	138
5.73.2.1	UpdateCallback	138
5.73.2.2	~UpdateCallback	138
5.73.3	Member Function Documentation	138
5.73.3.1	operator()	138
5.74	Vertex Struct Reference	139
5.74.1	Constructor & Destructor Documentation	139

5.74.1.1	Vertex	139
5.74.2	Member Data Documentation	139
5.74.2.1	mAnimID	139
5.74.2.2	mData	139
5.74.2.3	mDebugData	139
5.74.2.4	mDebugPrecision	139
5.74.2.5	mDebugRotation	139
6	File Documentation	141
6.1	animactorregistry.cpp File Reference	141
6.1.1	Function Documentation	141
6.1.1.1	CreatePluginRegistry	141
6.1.1.2	DestroyPluginRegistry	141
6.2	animactorregistry.h File Reference	142
6.3	animatable.cpp File Reference	143
6.4	animatable.h File Reference	144
6.5	animationchannel.cpp File Reference	145
6.6	animationchannel.h File Reference	146
6.7	animationcomponent.cpp File Reference	147
6.8	animationcomponent.h File Reference	148
6.9	animationgameactor.cpp File Reference	149
6.10	animationgameactor.h File Reference	150
6.11	animationhelper.cpp File Reference	151
6.12	animationhelper.h File Reference	152
6.13	animationsequence.cpp File Reference	153
6.14	animationsequence.h File Reference	154
6.15	animationwrapper.cpp File Reference	155
6.16	animationwrapper.h File Reference	156
6.17	animdriver.cpp File Reference	157
6.18	animdriver.h File Reference	158
6.19	animnodebuilder.cpp File Reference	159
6.20	animnodebuilder.h File Reference	160
6.21	attachmentcontroller.cpp File Reference	161
6.22	attachmentcontroller.h File Reference	162
6.23	cal3danimator.cpp File Reference	163
6.24	cal3danimator.h File Reference	164
6.25	cal3ddatabase.cpp File Reference	165
6.26	cal3ddatabase.h File Reference	166
6.27	cal3dgameactor.cpp File Reference	167
6.28	cal3dgameactor.h File Reference	168
6.29	cal3dloader.cpp File Reference	169

6.30	cal3dloader.h File Reference	170
6.31	cal3dmodeldata.cpp File Reference	171
6.32	cal3dmodeldata.h File Reference	172
6.33	cal3dmodelwrapper.cpp File Reference	173
6.34	cal3dmodelwrapper.h File Reference	174
6.35	characterfilehandler.cpp File Reference	175
	6.35.1 Variable Documentation	175
	6.35.1.1 XERCES_CPP_NAMESPACE_USE	175
6.36	characterfilehandler.h File Reference	176
6.37	characterwrapper.cpp File Reference	177
6.38	characterwrapper.h File Reference	178
6.39	chardrawable.cpp File Reference	179
6.40	chardrawable.h File Reference	180
6.41	dtanim.h File Reference	181
6.42	export.h File Reference	182
	6.42.1 Define Documentation	182
	6.42.1.1 DT_ANIM_EXPORT	182
6.43	hardwaresubmesh.cpp File Reference	183
	6.43.1 Define Documentation	183
	6.43.1.1 BUFFER_OFFSET	183
6.44	hardwaresubmesh.h File Reference	184
6.45	hotspotdriver.cpp File Reference	185
6.46	hotspotdriver.h File Reference	186
6.47	ical3ddriver.h File Reference	187
6.48	mainpage.h File Reference	188
	6.48.1 Detailed Description	188
6.49	morphdriver.cpp File Reference	189
6.50	morphdriver.h File Reference	190
6.51	physiquedriver.cpp File Reference	191
6.52	physiquedriver.h File Reference	192
6.53	posemath.cpp File Reference	193
6.54	posemath.h File Reference	194
6.55	posemesh.cpp File Reference	195
6.56	posemesh.h File Reference	196
	6.56.1 Define Documentation	196
	6.56.1.1 TRIANGLE_NOT_FOUND	196
6.57	posemeshdatabase.cpp File Reference	197
6.58	posemeshdatabase.h File Reference	198
6.59	posemeshloader.cpp File Reference	199
6.60	posemeshloader.h File Reference	200
6.61	posemeshutility.cpp File Reference	201

6.61.1	Function Documentation	201
6.61.1.1	BaseReferencePredicate	201
6.62	posemeshutility.h File Reference	202
6.63	posemeshxml.cpp File Reference	203
6.64	posemeshxml.h File Reference	204
6.65	sequencemixer.cpp File Reference	205
6.66	sequencemixer.h File Reference	206
6.67	skeletaldrawable.cpp File Reference	207
6.68	skeletaldrawable.h File Reference	208
6.69	skeletondriver.cpp File Reference	209
6.70	skeletondriver.h File Reference	210
6.71	springdriver.cpp File Reference	211
6.72	springdriver.h File Reference	212
6.73	submesh.cpp File Reference	213
6.73.1	Define Documentation	213
6.73.1.1	BUFFER_OFFSET	213
6.73.1.2	INDEX_OFFSET	213
6.74	submesh.h File Reference	214

Main Page

Delta3D is an Open Source engine which can be used for games, simulations, or other graphical applications.

The **Delta3D** framework exists as a number of modules, each sitting in its own library, enclosed within its own namespace. At the very core lies the **dtCore** library. This contains basic, low-level functionality which is mostly required for all 3D applications written in C++.

Around and alongside this sit other supporting libraries, such as **dtUtil** (containing reusable features which are useful for most applications), **dtTerrain** (for rendering terrain databases), **dtGame**, **dtNet**, etc.

Extensive online documentation is available from the Delta3D **Docs** section to help in using Delta3D.

The project's original reference guides generated by Doxygen from the source code may be viewed at the Delta3D **API Documentation** section.

To download source code, binaries, dependencies and sample datasets visit the Delta3D **Downloads** page.

For more about dependencies see the Delta3D **Dependencies** page.

The documentation you are looking at can be downloaded from www.3draum.ch.

Enjoy!

Todo List

Directory Documentation

3.1 inc/dtAnim/ Directory Reference

Files

- file [animactorregistry.h](#)
- file [animatable.h](#)
- file [animationchannel.h](#)
- file [animationcomponent.h](#)
- file [animationgameactor.h](#)
- file [animationhelper.h](#)
- file [animationsequence.h](#)
- file [animationwrapper.h](#)
- file [animdriver.h](#)
- file [animnodebuilder.h](#)
- file [attachmentcontroller.h](#)
- file [cal3danimator.h](#)
- file [cal3ddatabase.h](#)
- file [cal3dgameactor.h](#)
- file [cal3dloader.h](#)
- file [cal3dmodeldata.h](#)
- file [cal3dmodelwrapper.h](#)
- file [characterfilehandler.h](#)
- file [characterwrapper.h](#)
- file [chardrawable.h](#)
- file [dtanim.h](#)
- file [export.h](#)
- file [hardwaresubmesh.h](#)
- file [hotspotdriver.h](#)
- file [ical3ddriver.h](#)
- file [mainpage.h](#)
- file [morphdriver.h](#)
- file [physiquedriver.h](#)
- file [posemath.h](#)
- file [posemesh.h](#)
- file [posemeshdatabase.h](#)
- file [posemeshloader.h](#)
- file [posemeshutility.h](#)
- file [posemeshxml.h](#)
- file [sequencemixer.h](#)
- file [skeletaldrawable.h](#)
- file [skeletondriver.h](#)
- file [springdriver.h](#)
- file [submesh.h](#)

3.2 src/dtAnim/ Directory Reference

Files

- file [animactorregistry.cpp](#)
- file [animatable.cpp](#)
- file [animationchannel.cpp](#)
- file [animationcomponent.cpp](#)
- file [animationgameactor.cpp](#)
- file [animationhelper.cpp](#)
- file [animationsequence.cpp](#)
- file [animationwrapper.cpp](#)
- file [animdriver.cpp](#)
- file [animnodebuilder.cpp](#)
- file [attachmentcontroller.cpp](#)
- file [cal3danimator.cpp](#)
- file [cal3ddatabase.cpp](#)
- file [cal3dgameactor.cpp](#)
- file [cal3dloader.cpp](#)
- file [cal3dmodeldata.cpp](#)
- file [cal3dmodelwrapper.cpp](#)
- file [characterfilehandler.cpp](#)
- file [characterwrapper.cpp](#)
- file [chardrawable.cpp](#)
- file [hardwaresubmesh.cpp](#)
- file [hotspotdriver.cpp](#)
- file [morphdriver.cpp](#)
- file [physiquedriver.cpp](#)
- file [posemath.cpp](#)
- file [posemesh.cpp](#)
- file [posemeshdatabase.cpp](#)
- file [posemeshloader.cpp](#)
- file [posemeshutility.cpp](#)
- file [posemeshxml.cpp](#)
- file [sequencemixer.cpp](#)
- file [skeletaldrawable.cpp](#)
- file [skeletondriver.cpp](#)
- file [springdriver.cpp](#)
- file [submesh.cpp](#)

3.3 inc/ Directory Reference

Directories

- directory [dtAnim](#)

3.4 src/ Directory Reference

Directories

- directory [dtAnim](#)

Namespace Documentation

4.1 dtAnim Namespace Reference

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

Classes

- class [AnimActorRegistry](#)
- class [Animatable](#)

This class is used to specify the base class of an object which has semantics for animating.
- class [AnimationChannel](#)

AnimationChannel derives from Animatable and holds an AnimationWrapper, and contains semantics for playing an animation using the Cal3DModelWrapper API.
- class [AnimationComponent](#)
- class [AnimationGameActor](#)

This class is the game actor for an animated model.
- class [AnimationGameActorProxy](#)

This class is the proxy for an animated model game object.
- class [AnimationHelper](#)

The AnimationHelper class is a utility class to simplify adding animation to an articulated entity, it provides support for loading, rendering and animating.
- class [AnimationSequence](#)

AnimationSequence derives from Animatable and contains a child list of animations to play.
- class [AnimationWrapper](#)

The AnimationWrapper is meant to be a wrapper around a Cal3D animation.
- class [AnimDriver](#)
- class [AnimNodeBuilder](#)

Class used to generate the renderable geometry for an animated character.
- class [AnimSeqForceFade](#)
- struct [AnimSequenceUpdater](#)
- class [AttachmentController](#)

Stores a list of attachments for a cal model and can update their positions based each frame based on the position of the bones.
- class [AttachmentMover](#)

This is a helper functor for moving attachments.

- class [Cal3DAnimator](#)
- class [Cal3DDatabase](#)
- class [Cal3DGameActor](#)
 - This class is the game actor for an animated model.*
- class [Cal3DGameActorProxy](#)
 - This class is the proxy for an animated model game object.*
- class [Cal3DLoader](#)
 - Loads a animation definition file and returns a valid CalModel.*
- class [Cal3DModelData](#)
- class [Cal3DModelWrapper](#)
 - Wraps the Cal3D CalModel class.*
- class [CharacterFileHandler](#)
 - Simple Xerces XML handler that will store the data read from a character definition .xml file.*
- class [CharacterWrapper](#)
 - A Wrapper around an animated character that will perform basic steering.*
- class [CharDrawable](#)
 - A "view" of the cal3d animation state.*
- class [CloneFunctor](#)
- class [CreateGeometryDrawCallback](#)
 - Used to delay the building of the animated characters geometry until it is first rendered, at which point a valid OpenGL context should be valid.*
- struct [findWithCoreModel](#)
- struct [findWithFilename](#)
- class [HardwareSubmeshCallback](#)
- class [HardwareSubmeshComputeBound](#)
- class [HardwareSubmeshDrawable](#)
- class [HotSpotDriver](#)
 - updates the body offset value for HotSpot instances.*
- class [ICal3DDriver](#)
- class [IsActor](#)
- class [LODOptions](#)
 - A simple data class that stores the configuration options for level of detail.*
- class [MorphDriver](#)
- class [PhysiqueDriver](#)
- class [PoseMesh](#)
- struct [PoseMeshData](#)
- class [PoseMeshDatabase](#)
 - manager of the HotSpotData resources*
- class [PoseMeshFileHandler](#)
- class [PoseMeshLoader](#)
- class [PoseMeshUtility](#)
 - Convenience functionality for manipulating pose meshes.*
- class [RecalcFunctor](#)
- class [SequenceMixer](#)

The SequenceMixer's job is to manage animations and animation sequences.

- class [SkeletalDrawable](#)

Renders only the skeleton.

- class [SkeletonDriver](#)
- class [SpringDriver](#)
- class [SubmeshComputeBound](#)
- class [SubmeshCullCallback](#)
- class [SubmeshDirtyCallback](#)
- class [SubmeshDrawable](#)

Adapter that converts cal3d submeshes into osg::Drawables.

- class [SubmeshUserData](#)
- class [UpdateCallback](#)

Used to grab the created geometry from the [CreateGeometryDrawCallback](#) and add it as a child to the supplied Group.

Typedefs

- typedef dtUtil::Functor< void, TYPELIST_1(const dtAnim::Animatable &) > [AnimationCallback](#)
- typedef std::pair< dtCore::RefPtr< dtCore::Transformable >, dtUtil::HotSpotDefinition > [AttachmentPair](#)
- typedef std::vector< std::string > [StringVector](#)

Functions

- template<class T, class Array >
const Array::value_type::element_type * [FindWithFuncor](#) (Array a, T functor)
- void DT_ANIM_EXPORT [GetCelestialCoordinates](#) (osg::Vec3 targetDirection, const osg::Vec3 &lookForward, float &azimuth, float &elevation)
GetCelestialCoordinates - calculates the azimuth and elevation w.r.t.
- void DT_ANIM_EXPORT [GetCelestialDirection](#) (const float azimuth, const float elevation, const osg::Vec3 &forwardDirection, const osg::Vec3 &upDirection, osg::Vec3 &outDirection)
GetCelestialDirection - calculates the direction that a given azimuth and elevation points.
- void DT_ANIM_EXPORT [GetClosestPointOnSegment](#) (const osg::Vec3 &startPoint, const osg::Vec3 &endPoint, const osg::Vec3 &refPoint, osg::Vec3 &closestPoint)
GetClosestPointOnSegment - calculates the point on a segment that is closest to a reference point.
- bool DT_ANIM_EXPORT [IsPointBetweenVectors](#) (const osg::Vec3f &point, const osg::Vec3f &origin, const osg::Vec3f &A, const osg::Vec3f &B)
IsPointBetweenVectors - determines whether a points is between the vectors origin to A and origin to B.
- void DT_ANIM_EXPORT [MapCelestialToScreen](#) (float azimuth, float elevation, float maxDistance, float windowHeight, float windowWidth, const osg::Vec2 &screenOrigin, osg::Vec2 &outScreenPos)
MapCelestialToScreen - maps azimuth and elevation to a screen position.

4.1.1 Detailed Description

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

4.1.2 Typedef Documentation

4.1.2.1 `typedef dtUtil::Functor<void, TYPELIST_1(const dtAnim::Animatable&) > AnimationCallback`

4.1.2.2 `typedef std::pair<dtCore::RefPtr<dtCore::Transformable>, dtUtil::HotSpotDefinition > AttachmentPair`

4.1.2.3 `typedef std::vector<std::string> StringVector`

4.1.3 Function Documentation

4.1.3.1 `const Array::value_type::element_type* dtAnim::FindWithFunctor (Array a, T functor) [inline]`

4.1.3.2 `void GetCelestialCoordinates (osg::Vec3 targetDirection, const osg::Vec3 & lookForward, float & azimuth, float & elevation)`

GetCelestialCoordinates - calculates the azimuth and elevation w.r.t. a forward vector Parameters

targetDirection - a direction vector pointing to the point of interest

lookForward - current forward vector

azimuth - the horizontal angle between our forward and our target

elevation - the vertical angle between our forward and our target

4.1.3.3 `void GetCelestialDirection (const float azimuth, const float elevation, const osg::Vec3 & forwardDirection, const osg::Vec3 & upDirection, osg::Vec3 & outDirection)`

GetCelestialDirection - calculates the direction that a given azimuth and elevation points. Parameters

azimuth - the horizontal angle of interest

elevation - the vertical angle of interest

forwardDirection - the current forward direction the azimuth and elevation are w.r.t

outDirection - the direction that a given azimuth and elevation points

4.1.3.4 `void GetClosestPointOnSegment (const osg::Vec3 & startPoint, const osg::Vec3 & endPoint, const osg::Vec3 & refPoint, osg::Vec3 & closestPoint)`

GetClosestPointOnSegment - calculates the point on a segment that is closest to a reference point. Parameters

startPoint - the start point of a line segment

endPoint - the end point of a line segment

refPoint - the point whose closest distance to the segment we are interested in

closestPoint - the final calculated point that is closest to refPoint

4.1.3.5 `bool IsPointBetweenVectors (const osg::Vec3f & point, const osg::Vec3f & origin, const osg::Vec3f & A, const osg::Vec3f & B)`

IsPointBetweenVectors - determines whether a points is between the vectors origin to A and origin to B. Parameters

point - the point that we want to test

origin - the point which is shared by both vectors

A - the end point for the first vector

B - the end point for the second vector

4.1.3.6 void MapCelestialToScreen (float *azimuth*, float *elevation*, float *maxDistance*, float *windowWidth*, float *windowHeight*, const osg::Vec2 & *screenOrigin*, osg::Vec2 & *outScreenPos*)

MapCelestialToScreen - maps azimuth and elevation to a screen position. Parameters

azimuth - the azimuth to be mapped

elevation - the elevation to be mapped

windowWidth - the desired max width of the screen sub window

windowHeight - the desired max height of the screen sub window

screenOrigin - the position on the screen from which the sub window begins

outScreenPos - the transformed final position in normalized screen coordinates

4.2 dtCore Namespace Reference

4.3 dtDAL Namespace Reference

4.4 dtGame Namespace Reference

4.5 dtUtil Namespace Reference

Class Documentation

5.1 AnimActorRegistry Class Reference

```
#include <inc/dtAnim/animactorregistry.h>
```

Public Member Functions

- [AnimActorRegistry \(\)](#)
- virtual void [RegisterActorTypes \(\)](#)

Static Public Attributes

- static dtCore::RefPtr< dtDAL::ActorType > [ANIMATION_ACTOR_TYPE](#)
- static dtCore::RefPtr< dtDAL::ActorType > [CAL3D_ACTOR_TYPE](#)

5.1.1 Constructor & Destructor Documentation

5.1.1.1 AnimActorRegistry ()

5.1.2 Member Function Documentation

5.1.2.1 void RegisterActorTypes () [virtual]

5.1.3 Member Data Documentation

5.1.3.1 dtCore::RefPtr< dtDAL::ActorType > ANIMATION_ACTOR_TYPE [static]

5.1.3.2 dtCore::RefPtr< dtDAL::ActorType > CAL3D_ACTOR_TYPE [static]

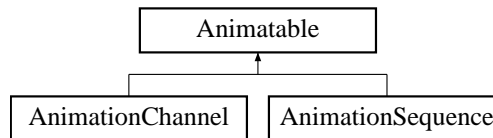
The documentation for this class was generated from the following files:

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

5.2 Animatable Class Reference

This class is used to specify the base class of an object which has semantics for animating.

#include <inc/dtAnim/animatable.h> Inheritance diagram for Animatable::



Public Member Functions

- [Animatable](#) (const [Animatable](#) &pAnim)
- [Animatable](#) ()
- virtual dtCore::RefPtr< [Animatable](#) > [Clone](#) ([Cal3DModelWrapper](#) *modelWrapper) const =0
This function is used to copy Animatables.
- virtual void [ForceFadeOut](#) (float time)=0
ForceFadeOut will ignore the EndTime and automatically fade out this animation over the time specified.
- float [GetBaseWeight](#) () const
The BaseWeight specifies the weight of this animation without being affected by blending.
- float [GetCurrentWeight](#) () const
The CurrentWeight is the weight of this animation in the current frame this is calculated from the BaseWeight, linear blending from fades, and the parent weight.
- float [GetElapsedTime](#) () const
The elapsed time is the time in seconds since this animation has been added to the play list.
- float [GetEndTime](#) () const
The end time is the time in seconds that this animation will start fading out relative to when it was added to an [AnimationSequence](#).
- float [GetFadeIn](#) () const
The FadeIn time, is the amount of time takes for an animation to blend linearly from a weight of 0 to the weight specified by BaseWeight after the animation starts playing specified by StartTime.
- float [GetFadeOut](#) () const
The FadeOut time is the amount of time it will take for an animation to blend linearly from its BaseWeight to 0 after the EndTime.
- const std::string & [GetName](#) () const
- float [GetSpeed](#) () const
The speed of an animation is the percentage relative to the actual speed of playback.
- float [GetStartDelay](#) () const
- float [GetStartTime](#) () const
The start time is the time in seconds this animation will start playing after it was added to an [AnimationSequence](#).
- bool [IsActive](#) () const
An animation is active if it is currently playing.
- [Animatable](#) & [operator=](#) (const [Animatable](#) &pAnim)
- virtual void [Prune](#) ()=0

This virtual function is called before this animation is removed from the system.

- virtual void [Recalculate](#) ()=0
- void [SetBaseWeight](#) (float f)
- void [SetCurrentWeight](#) (float weight)
- void [SetElapsedTime](#) (float t)
- void [SetEndCallback](#) ([AnimationCallback](#) callback)

Set the functor to be called when the animation ends.

- void [SetFadeIn](#) (float f)
- void [SetFadeOut](#) (float f)
- void [SetName](#) (const std::string &)
- void [SetSpeed](#) (float speed)
- void [SetStartDelay](#) (float t)

The start delay specifies how long in seconds this animation will delay once its parent has started playing.

- void [SetStartTime](#) (float t)

These functions are only to be used by AnimationController on [Update\(\)](#) and [Recalculate](#).

- bool [ShouldPrune](#) () const

This flag specifies whether or not this animation has stopped playing.

- virtual void [Update](#) (float dt)=0

The virtual update, should be called every frame.

Protected Member Functions

- virtual [~Animatable](#) ()
- void [SetActive](#) (bool b)
- void [SetEndTime](#) (float t)
- void [SetPrune](#) (bool b)

When this flag is set the parent sequence will call [Prune\(\)](#) at the end of its update and then delete this animation.

5.2.1 Detailed Description

This class is used to specify the base class of an object which has semantics for animating.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Animatable ()

5.2.2.2 Animatable (const Animatable & *pAnim*)

5.2.2.3 ~Animatable () [protected, virtual]

5.2.3 Member Function Documentation

5.2.3.1 virtual dtCore::RefPtr<Animatable> Clone (Cal3DModelWrapper * *modelWrapper*) const [pure virtual]

This function is used to copy Animatables.

Implemented in [AnimationChannel](#), and [AnimationSequence](#).

5.2.3.2 virtual void ForceFadeOut (float *time*) [pure virtual]

ForceFadeOut will ignore the EndTime and automatically fade out this animation over the time specified. Parameters

the time to fade out over

Implemented in [AnimationChannel](#), and [AnimationSequence](#).

5.2.3.3 float GetBaseWeight () const

The BaseWeight specifies the weight of this animation without being affected by blending.

5.2.3.4 float GetCurrentWeight () const

The CurrentWeight is the weight of this animation in the current frame this is calculated from the BaseWeight, linear blending from fades, and the parent weight.

5.2.3.5 float GetElapsedTime () const

The elapsed time is the time in seconds since this animation has been added to the play list.

5.2.3.6 float GetEndTime () const

The end time is the time in seconds that this animation will start fading out relative to when it was added to an [AnimationSequence](#).

5.2.3.7 float GetFadeIn () const

The FadeIn time, is the amount of time takes for an animation to blend linearly from a weight of 0 to the weight specified by BaseWeight after the animation starts playing specified by StartTime.

5.2.3.8 float GetFadeOut () const

The FadeOut time is the amount of time it will take for an animation to blend linearly from its BaseWeight to 0 after the EndTime.

5.2.3.9 const std::string & GetName () const

Returns the name of this animation

5.2.3.10 float GetSpeed () const

The speed of an animation is the percentage relative to the actual speed of playback. It defaults to 1.0, a speed of 2.0 would play twice as fast.

5.2.3.11 float GetStartDelay () const

5.2.3.12 float GetStartTime () const

The start time is the time in seconds this animation will start playing after it was added to an [AnimationSequence](#).

5.2.3.13 bool IsActive () const

An animation is active if it is currently playing.

5.2.3.14 Animatable & operator= (const Animatable & pAnim)

5.2.3.15 virtual void Prune () [pure virtual]

This virtual function is called before this animation is removed from the system.

Implemented in [AnimationChannel](#), and [AnimationSequence](#).

5.2.3.16 virtual void Recalculate () [pure virtual]

Implemented in [AnimationChannel](#), and [AnimationSequence](#).

5.2.3.17 void SetActive (bool b) [protected]

5.2.3.18 void SetBaseWeight (float f)

5.2.3.19 void SetCurrentWeight (float weight)

5.2.3.20 void SetElapsedTime (float f)

5.2.3.21 void SetEndCallback (AnimationCallback callback)

Set the functor to be called when the animation ends. Parameters

callback The functor to be called when this animation ends. The [Animatable](#) will be passed into the callback so that the callee can determine what animation has ended.

5.2.3.22 void SetEndTime (float *t*) [protected]

5.2.3.23 void SetFadeIn (float *f*)

5.2.3.24 void SetFadeOut (float *f*)

5.2.3.25 void SetName (const std::string & *pName*)

Parameters

the name to set this animation to

5.2.3.26 void SetPrune (bool *b*) [protected]

When this flag is set the parent sequence will call [Prune\(\)](#) at the end of its update and then delete this animation.

5.2.3.27 void SetSpeed (float *speed*)

5.2.3.28 void SetStartDelay (float *t*)

The start delay specifies how long in seconds this animation will delay once its parent has started playing.

5.2.3.29 void SetStartTime (float *t*)

These functions are only to be used by AnimationController on [Update\(\)](#) and Recalculate.

5.2.3.30 bool ShouldPrune () const

This flag specifies whether or not this animation has stopped playing.

5.2.3.31 virtual void Update (float *dt*) [pure virtual]

The virtual update, should be called every frame. Parameters

delta time

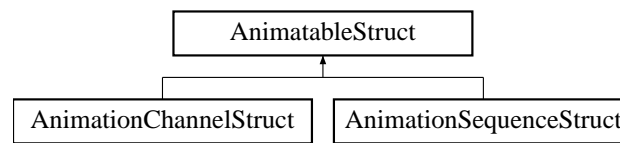
Implemented in [AnimationChannel](#), and [AnimationSequence](#).

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

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

5.3 AnimatableStruct Struct Reference

#include <inc/dtAnim/characterfilehandler.h> Inheritance diagram for AnimatableStruct::



Public Member Functions

- [AnimatableStruct \(\)](#)

Public Attributes

- float [mBaseWeight](#)
- float [mFadeIn](#)
- float [mFadeOut](#)
- std::string [mName](#)
The name of this animation channel.
- float [mSpeed](#)
- float [mStartDelay](#)

5.3.1 Constructor & Destructor Documentation

5.3.1.1 AnimatableStruct ()

5.3.2 Member Data Documentation

5.3.2.1 float mBaseWeight

5.3.2.2 float mFadeIn

5.3.2.3 float mFadeOut

5.3.2.4 std::string mName

The name of this animation channel.

5.3.2.5 float mSpeed

5.3.2.6 float mStartDelay

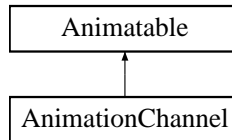
The documentation for this struct was generated from the following files:

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

5.4 AnimationChannel Class Reference

[AnimationChannel](#) derives from [Animatable](#) and holds an [AnimationWrapper](#), and contains semantics for playing an animation using the [Cal3DModelWrapper](#) API.

#include <inc/dtAnim/animationchannel.h> Inheritance diagram for AnimationChannel::



Public Member Functions

- [AnimationChannel](#) ([Cal3DModelWrapper](#) *pModelWrapper, [AnimationWrapper](#) *pAnimationWrapper)
- [AnimationChannel](#) ()
 - If you use the default constructor you must call [SetAnimation](#), and [SetModel](#).*
- virtual dtCore::RefPtr< [Animatable](#) > [Clone](#) ([Cal3DModelWrapper](#) *wrapper) const
 - This function copies the animation channel.*
- void [ForceFadeOut](#) (float time)
 - ForceFadeOut will ignore the EndTime and automatically fade out this animation over the time specified.*
- const [AnimationWrapper](#) * [GetAnimation](#) () const
- [AnimationWrapper](#) * [GetAnimation](#) ()
- float [GetMaxDuration](#) () const
 - The Max Duration will allow a user to set the amount of time this animation will play in seconds.*
- const [Cal3DModelWrapper](#) * [GetModel](#) () const
- [Cal3DModelWrapper](#) * [GetModel](#) ()
- bool [IsAction](#) () const
 - If an animation is an action it will play once and automatically delete itself it will also be weighted to override all other conflicting animations.*
- bool [IsLooping](#) () const
 - If an animation is not looping then it will be considered an action and the blend weights will be ignored.*
- void [Prune](#) ()
 - Prune is called before the animation is deleted.*
- void [Recalculate](#) ()
 - Recalculate is called on [PlayAnimation\(\)](#) it calculates the start and end times of our animation.*
- void [SetAction](#) (bool b)
- void [SetAnimation](#) ([AnimationWrapper](#) *pAnimation)
 - This function associates this channel with the supplied animation wrapper.*
- void [SetLooping](#) (bool b)
- void [SetMaxDuration](#) (float b)
- void [SetModel](#) ([Cal3DModelWrapper](#) *pWrapper)
 - This function sets the model wrapper used to make the calls to play the animation.*
- void [Update](#) (float dt)
 - The per frame update function.*

Protected Member Functions

- [AnimationChannel](#) (const [AnimationChannel](#) &)
Copy Constructor Only clone should call this.
- [~AnimationChannel](#) ()
- [AnimationChannel](#) & `operator=` (const [AnimationChannel](#) &)
Operator Equal Hide this to prevent assigning.

5.4.1 Detailed Description

[AnimationChannel](#) derives from [Animatable](#) and holds an [AnimationWrapper](#), and contains semantics for playing an animation using the [Cal3DModelWrapper](#) API.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 AnimationChannel ()

If you use the default constructor you must call `SetAnimation`, and `SetModel`.

5.4.2.2 AnimationChannel (Cal3DModelWrapper * pModelWrapper, AnimationWrapper * pAnimationWrapper)

Parameters

the model wrapper used to play animations with

the animation wrapper to specify which animation to play

5.4.2.3 ~AnimationChannel () [protected]

5.4.2.4 AnimationChannel (const AnimationChannel & pChannel) [protected]

Copy Constructor Only clone should call this.

5.4.3 Member Function Documentation

5.4.3.1 dtCore::RefPtr< Animatable > Clone (Cal3DModelWrapper * wrapper) const [virtual]

This function copies the animation channel. Parameters

Implements [Animatable](#).

5.4.3.2 void ForceFadeOut (float time) [virtual]

`ForceFadeOut` will ignore the `EndTime` and automatically fade out this animation over the time specified. Parameters

the time to fade out over

Implements [Animatable](#).

5.4.3.3 const AnimationWrapper * GetAnimation () const

5.4.3.4 AnimationWrapper * GetAnimation ()

5.4.3.5 float GetMaxDuration () const

The Max Duration will allow a user to set the amount of time this animation will play in seconds.

5.4.3.6 const Cal3DModelWrapper * GetModel () const

Returns the const model wrapper assigned to this channel.

5.4.3.7 Cal3DModelWrapper * GetModel ()

Returns the model wrapper assigned to this channel.

5.4.3.8 bool IsAction () const

If an animation is an action it will play once and automatically delete itself it will also be weighted to override all other conflicting animations.

5.4.3.9 bool IsLooping () const

If an animation is not looping then it will be considered an action and the blend weights will be ignored.

5.4.3.10 AnimationChannel & operator= (const AnimationChannel & pChannel) [protected]

Operator Equal Hide this to prevent assigning.

5.4.3.11 void Prune () [virtual]

Prune is called before the animation is deleted.

Implements [Animatable](#).

5.4.3.12 void Recalculate () [virtual]

Recalculate is called on PlayAnimation() it calculates the start and end times of our animation.

Implements [Animatable](#).

5.4.3.13 void SetAction (bool b)**5.4.3.14 void SetAnimation (AnimationWrapper * pAnimation)**

This function associates this channel with the supplied animation wrapper. Parameters

the animation wrapper this channel will play

5.4.3.15 void SetLooping (bool b)**5.4.3.16 void SetMaxDuration (float b)****5.4.3.17 void SetModel (Cal3DModelWrapper * pWrapper)**

This function sets the model wrapper used to make the calls to play the animation. Parameters

the associated model wrapper

5.4.3.18 void Update (float dt) [virtual]

The per frame update function. Parameters

delta time

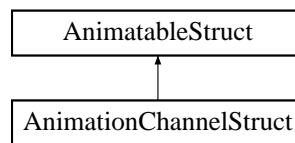
Implements [Animatable](#).

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

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

5.5 AnimationChannelStruct Struct Reference

#include <inc/dtAnim/characterfilehandler.h>Inheritance diagram for AnimationChannelStruct::



Public Member Functions

- [AnimationChannelStruct \(\)](#)

Public Attributes

- `std::string` [mAnimationName](#)
The name of the animation this references.
- `bool` [mIsAction](#)
- `bool` [mIsLooping](#)
- `float` [mMaxDuration](#)

5.5.1 Constructor & Destructor Documentation

5.5.1.1 AnimationChannelStruct ()

5.5.2 Member Data Documentation

5.5.2.1 `std::string` mAnimationName

The name of the animation this references.

5.5.2.2 `bool` mIsAction

5.5.2.3 `bool` mIsLooping

5.5.2.4 `float` mMaxDuration

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

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

5.6 AnimationComponent Class Reference

```
#include <inc/dtAnim/animationcomponent.h>
```

Public Types

- typedef AnimCompMap::iterator [AnimComplter](#)
- typedef std::map< dtCore::Uniqueld, dtCore::RefPtr< dtAnim::AnimationHelper > > [AnimCompMap](#)
- typedef AnimCompMap::allocator_type::value_type [AnimCompMapping](#)
- typedef dtGame::GMComponent [BaseClass](#)

Public Member Functions

- [AnimationComponent](#) (const std::string &name=[DEFAULT_NAME](#))
- const dtCore::Transformable * [GetEyePointActor](#) () const
- dtCore::Transformable * [GetEyePointActor](#) ()
- const dtGame::BaseGroundClamper & [GetGroundClamper](#) () const
- dtGame::BaseGroundClamper & [GetGroundClamper](#) ()
Get the ground clamper responsible for clamping animated objects.
- const dtAnim::AnimationHelper * [GetHelperForProxy](#) (dtGame::GameActorProxy &proxy) const
Gets the helper registered for an actor.
- const dtCore::Transformable * [GetTerrainActor](#) () const
- dtCore::Transformable * [GetTerrainActor](#) ()
- bool [IsRegisteredActor](#) (dtGame::GameActorProxy &gameActorProxy)
- void [ProcessMessage](#) (const dtGame::Message &message)
handles a processed a message
- void [RegisterActor](#) (dtGame::GameActorProxy &toRegister, dtAnim::AnimationHelper &helper)
Registers an actor with this component.
- void [SetEyePointActor](#) (dtCore::Transformable *newEyePointActor)
changes the actor to use for the terrain.
- void [SetGroundClamper](#) (dtGame::BaseGroundClamper &clamper)
Set the ground clamper responsible for clamping animated objects.
- void [SetTerrainActor](#) (dtCore::Transformable *newTerrain)
changes the actor to use for the terrain.
- void [UnregisterActor](#) (dtGame::GameActorProxy &toRegister)
Registers an actor with this component.

Static Public Attributes

- static const std::string [DEFAULT_NAME](#)
The default component name, used when looking it up on the GM.

Protected Member Functions

- virtual [~AnimationComponent](#) ()
- void [GroundClamp](#) ()
- virtual void [TickLocal](#) (float dt)

5.6.1 Member Typedef Documentation

5.6.1.1 **typedef AnimCompMap::iterator AnimComplter**

5.6.1.2 **typedef std::map<dtCore::Uniqueld, dtCore::RefPtr<dtAnim::AnimationHelper> > AnimCompMap**

5.6.1.3 **typedef AnimCompMap::allocator_type::value_type AnimCompMapping**

5.6.1.4 **typedef dtGame::GMComponent BaseClass**

5.6.2 Constructor & Destructor Documentation

5.6.2.1 **AnimationComponent (const std::string & *name* = DEFAULT_NAME)**

5.6.2.2 **~AnimationComponent () [protected, virtual]**

5.6.3 Member Function Documentation

5.6.3.1 **const dtCore::Transformable * GetEyePointActor () const**

Returns the actor to use as an eye point for ground clamping. This determines which LOD to clamp to.

5.6.3.2 **dtCore::Transformable * GetEyePointActor ()**

Returns the actor to use as an eye point for ground clamping. This determines which LOD to clamp to.

5.6.3.3 **const dtGame::BaseGroundClamper & GetGroundClamper () const**

5.6.3.4 **dtGame::BaseGroundClamper & GetGroundClamper ()**

Get the ground clamper responsible for clamping animated objects.

5.6.3.5 **const dtAnim::AnimationHelper * GetHelperForProxy (dtGame::GameActorProxy & *proxy*) const**

Gets the helper registered for an actor. Parameters

proxy The proxy to get the helper for

Returns A pointer to the helper, or NULL if the proxy is not registered

5.6.3.6 **const dtCore::Transformable * GetTerrainActor () const**

Returns the terrain actor using the given name. If it has not yet been queried, the query will run when this is called.

5.6.3.7 **dtCore::Transformable * GetTerrainActor ()**

Returns the terrain actor using the given name. If it has not yet been queried, the query will run when this is called.

5.6.3.8 **void GroundClamp () [protected]**

5.6.3.9 **bool IsRegisteredActor (dtGame::GameActorProxy & *gameActorProxy*)**

Returns true if the given actor is registered with this component.

5.6.3.10 **void ProcessMessage (const dtGame::Message & *message*)**

handles a processed a message See also dtGame::GMComponent::ProcessMessage

Parameters

The message

5.6.3.11 void RegisterActor (dtGame::GameActorProxy & *toRegister*, dtAnim::AnimationHelper & *helper*)

Registers an actor with this component. To simplify coding in the actor, specifically when it comes to setting properties on the helper, the actor should create it's own helper and pass it in when registering. Parameters

toRegister the actor to register.

helper the preconfigured helper object to use.

Exceptions

dtUtil::Exception if this actor is already registered with the component.

5.6.3.12 void SetEyePointActor (dtCore::Transformable * *newEyePointActor*)

changes the actor to use for the terrain.

5.6.3.13 void SetGroundClamper (dtGame::BaseGroundClamper & *clamper*)

Set the ground clamper responsible for clamping animated objects.

5.6.3.14 void SetTerrainActor (dtCore::Transformable * *newTerrain*)

changes the actor to use for the terrain.

5.6.3.15 void TickLocal (float *dt*) [protected, virtual]**5.6.3.16 void UnregisterActor (dtGame::GameActorProxy & *toRegister*)**

Registers an actor with this component. To simplify coding in the actor, specifically when it comes to setting properties on the helper, the actor should create it's own helper and pass it in when registering. Parameters

toRegister the actor to register.

helper the preconfigured helper object to use.

5.6.4 Member Data Documentation**5.6.4.1 const std::string DEFAULT_NAME [static]**

The default component name, used when looking it up on the GM.

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

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

5.7 AnimationController Class Reference

[AnimationController](#) is responsible for updating the sequences child Animatables.

```
#include <inc/dtAnim/animationsequence.h>
```

Public Member Functions

- [AnimationController](#) (const [AnimationController](#) &)
copy constructor
- [AnimationController](#) ([AnimationSequence](#) &)
Constructor.
- virtual dtCore::RefPtr< [AnimationController](#) > [Clone](#) () const
classes derived from this should implement a custom [Clone\(\)](#)
- const [AnimationSequence](#) & [GetParent](#) () const
- [AnimationSequence](#) & [GetParent](#) ()
- [AnimationController](#) & [operator=](#) (const [AnimationController](#) &)
operator equal
- virtual void [Recalculate](#) ()
The Recalculate function is responsible for calculating the start times of all the parent sequences children and then calling [Recalculate\(\)](#) on them so they can calculate their end times.
- void [SetParent](#) ([AnimationSequence](#) &)
Sets the parent [AnimationSequence](#).
- virtual void [Update](#) (float dt)
The Update function is responsible for updating all the parent sequences children.

Protected Member Functions

- virtual ~[AnimationController](#) ()
- void [SetComputeSpeed](#) ([Animatable](#) *pAnim)
Helper function to compute the speed of a child animatable using the parents speed.
- void [SetComputeWeight](#) ([Animatable](#) *pAnim)
Helper function to compute the weight of a child animatable using the parents weight.

5.7.1 Detailed Description

[AnimationController](#) is responsible for updating the sequences child Animatables. To change the update behavior, subclass [AnimationController](#) and add it to your [AnimationSequence](#).

5.7.2 Constructor & Destructor Documentation

5.7.2.1 AnimationController (AnimationSequence & pParent)

Constructor. Parameters

the parent AnimationSequence

5.7.2.2 AnimationController (const AnimationController & pConf)

copy constructor

5.7.2.3 `~AnimationController () [protected, virtual]`

5.7.3 Member Function Documentation

5.7.3.1 `dtCore::RefPtr< AnimationSequence::AnimationController > Clone () const [virtual]`

classes derived from this should implement a custom [Clone\(\)](#)

5.7.3.2 `const AnimationSequence & GetParent () const`

Returns the parent [AnimationSequence](#)

5.7.3.3 `AnimationSequence & GetParent ()`

Returns the parent [AnimationSequence](#)

5.7.3.4 `AnimationSequence::AnimationController & operator= (const AnimationController & pCont)`

operator equal

5.7.3.5 `void Recalculate () [virtual]`

The Recalculate function is responsible for calculating the start times of all the parent sequences children and then calling [Recalculate\(\)](#) on them so they can calculate their end times.

5.7.3.6 `void SetComputeSpeed (Animatable * pAnim) [protected]`

Helper function to compute the speed of a child animatable using the parents speed.

5.7.3.7 `void SetComputeWeight (Animatable * pAnim) [protected]`

Helper function to compute the weight of a child animatable using the parents weight.

5.7.3.8 `void SetParent (AnimationSequence & pParent)`

Sets the parent [AnimationSequence](#).

5.7.3.9 `void Update (float dt) [virtual]`

The Update function is responsible for updating all the parent sequences children. This update consists of incrementing their elapsed time, calculating their current weight, and call update on them if they are active.

Parameters

delta time

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

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

5.8 AnimationGameActor Class Reference

This class is the game actor for an animated model.

```
#include <inc/dtAnim/animationgameactor.h>
```

Public Member Functions

- [AnimationGameActor](#) (dtGame::GameActorProxy &proxy)
Constructs an [AnimationGameActor](#) actor.
- const [dtAnim::AnimationHelper](#) * [GetHelper](#) () const
- [dtAnim::AnimationHelper](#) * [GetHelper](#) ()
- virtual void [SetModel](#) (const std::string &modelFile)
Loads a model file.

Protected Member Functions

- virtual [~AnimationGameActor](#) ()
Destroys this actor.

Protected Attributes

- dtCore::RefPtr< [dtAnim::AnimationHelper](#) > [mHelper](#)

5.8.1 Detailed Description

This class is the game actor for an animated model. See also [GameActor](#) [AnimationGameActorProxy](#)

5.8.2 Constructor & Destructor Documentation

5.8.2.1 AnimationGameActor (dtGame::GameActorProxy & proxy)

Constructs an [AnimationGameActor](#) actor. Parameters

proxy The actor proxy owning this task actor.

5.8.2.2 ~AnimationGameActor () [protected, virtual]

Destroys this actor.

5.8.3 Member Function Documentation

5.8.3.1 const dtAnim::AnimationHelper * GetHelper () const

5.8.3.2 dtAnim::AnimationHelper * GetHelper ()

5.8.3.3 void SetModel (const std::string & modelFile) [virtual]

Loads a model file. Parameters

modelFile The filename of the model to load.

5.8.4 Member Data Documentation

5.8.4.1 dtCore::RefPtr<dtAnim::AnimationHelper> mHelper [protected]

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

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

5.9 AnimationGameActorProxy Class Reference

This class is the proxy for an animated model game object.

```
#include <inc/dtAnim/animationgameactor.h>
```

Public Member Functions

- [AnimationGameActorProxy \(\)](#)
Constructs the proxy.
- virtual void [BuildPropertyMap \(\)](#)
Builds the property map for the task actor proxy.
- virtual dtDAL::ActorProxyIcon * [GetBillBoardIcon \(\)](#)
Gets the billboard used to represent static mesh if this proxy's render mode is RenderMode::DRAW_BILLBOARD_ICON.
- virtual const dtDAL::ActorProxy::RenderMode & [GetRenderMode \(\)](#)
Gets the method by which this static mesh is rendered.

Protected Member Functions

- virtual [~AnimationGameActorProxy \(\)](#)
Destroys the proxy.
- virtual void [CreateActor \(\)](#)
Called by the game manager during creation of the proxy.

5.9.1 Detailed Description

This class is the proxy for an animated model game object. See also [GameActorProxy AnimationGameActor](#)

5.9.2 Constructor & Destructor Documentation

5.9.2.1 AnimationGameActorProxy ()

Constructs the proxy.

5.9.2.2 ~AnimationGameActorProxy () [protected, virtual]

Destroys the proxy.

5.9.3 Member Function Documentation

5.9.3.1 void BuildPropertyMap () [virtual]

Builds the property map for the task actor proxy. These properties wrap the specified properties located in the actor.

5.9.3.2 void CreateActor () [protected, virtual]

Called by the game manager during creation of the proxy. This method creates the real actor and returns it.

5.9.3.3 dtDAL::ActorProxyIcon * GetBillBoardIcon () [virtual]

Gets the billboard used to represent static mesh if this proxy's render mode is RenderMode::DRAW_BILLBOARD_ICON. Used by STAGE. Returns

5.9.3.4 const dtDAL::ActorProxy::RenderMode & GetRenderMode () [virtual]

Gets the method by which this static mesh is rendered. This is used by STAGE. Returns If there is no geometry currently assigned, this method will return RenderMode::DRAW_BILLBOARD_ICON. If there is geometry assigned to this static mesh, RenderMode::DRAW_ACTOR is returned.

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

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

5.10 AnimationHelper Class Reference

The [AnimationHelper](#) class is a utility class to simplify adding animation to an articulated entity, it provides support for loading, rendering and animating.

```
#include <inc/dtAnim/animationhelper.h>
```

Public Member Functions

- [AnimationHelper](#) ()

The constructor constructs a default [AnimNodeBuilder](#), the [Cal3DModelWrapper](#), and [AnimationController](#) are created on [LoadModel\(\)](#).
- void [ClearAll](#) (float fadeOutTime)

This function stops playing all currently active animations over the time specified by fade out.
- void [ClearAnimation](#) (const std::string &pAnim, float fadeOutTime)

This function stops playing an animation by name over the course of time specified by fade out.
- virtual void [GetActorProperties](#) (dtDAL::ActorProxy &pProxy, std::vector< dtCore::RefPtr< dtDAL::ActorProperty > > &pFillVector)

This function is used to create the proper actor properties for an animated entity after calling this function the user must iterate through the vector and add each property to its proxy.
- const [Cal3DAnimator](#) * [GetAnimator](#) () const
- [Cal3DAnimator](#) * [GetAnimator](#) ()
- [AttachmentController](#) & [GetAttachmentController](#) ()

The animation helper has an attachment controller that moves the Transformable attachments to match up to the bones.
- bool [GetGroundClamp](#) () const

This flag is used by the [AnimationComponent](#) to determine if this entity should be ground clamped.
- const [Cal3DModelWrapper](#) * [GetModelWrapper](#) () const
- [Cal3DModelWrapper](#) * [GetModelWrapper](#) ()
- const osg::Node * [GetNode](#) () const
- osg::Node * [GetNode](#) ()
- const [SequenceMixer](#) & [GetSequenceMixer](#) () const
- [SequenceMixer](#) & [GetSequenceMixer](#) ()
- bool [LoadModel](#) (const std::string &pFilename)

This function loads a character XML file from string, on loading it creates a [Cal3DAnimator](#) with the [Cal3DModelWrapper](#) and then calls [CreateGeode\(\)](#) on the [AnimNodeBuilder](#).
- void [PlayAnimation](#) (const std::string &pAnim)

This function plays the specified animation defined within the character XML.
- void [SetAttachmentController](#) ([AttachmentController](#) &newController)

Assigns a new [AttachmentController](#).
- void [SetGroundClamp](#) (bool b)

Set whether or not this entity should be ground clamped.
- virtual void [Update](#) (float dt)

The user should call [Update\(\)](#) on a per frame basis this function updates the sequence mixer and the [Cal3DAnimator](#).

Static Public Attributes

- static const std::string [PROPERTY_SKELETAL_MESH](#)

Protected Member Functions

- virtual [~AnimationHelper](#) ()

5.10.1 Detailed Description

The [AnimationHelper](#) class is a utility class to simplify adding animation to an articulated entity, it provides support for loading, rendering and animating.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 AnimationHelper ()

The constructor constructs a default [AnimNodeBuilder](#), the [Cal3DModelWrapper](#), and [AnimationController](#) are created on [LoadModel\(\)](#).

5.10.2.2 ~AnimationHelper () [protected, virtual]

5.10.3 Member Function Documentation

5.10.3.1 void ClearAll (float *fadeOutTime*)

This function stops playing all currently active animations over the time specified by fade out. Parameters

The amount of time to fade out over

5.10.3.2 void ClearAnimation (const std::string & *pAnim*, float *fadeOutTime*)

This function stops playing an animation by name over the course of time specified by fade out. Parameters

The name of the animation to clear

The amount of time to fade out over

5.10.3.3 void GetActorProperties (dtDAL::ActorProxy & *pProxy*, std::vector< dtCore::RefPtr< dtDAL::ActorProperty > > & *pFillVector*) [virtual]

This function is used to create the proper actor properties for an animated entity after calling this function the user must iterate through the vector and add each property to its proxy. Parameters

the actor proxy

an empty vector to fill of actor properties

5.10.3.4 const Cal3DAnimator * GetAnimator () const

Returns The [Cal3DAnimator](#) created on [LoadModel](#)

5.10.3.5 Cal3DAnimator * GetAnimator ()

Returns The [Cal3DAnimator](#) created on [LoadModel](#)

5.10.3.6 AttachmentController & GetAttachmentController ()

The animation helper has an attachment controller that moves the Transformable attachments to match up to the bones. Returns the currently assigned attachment controller.

5.10.3.7 bool GetGroundClamp () const

This flag is used by the [AnimationComponent](#) to determine if this entity should be ground clamped.

5.10.3.8 const Cal3DModelWrapper * GetModelWrapper () const

Returns The [Cal3DModelWrapper](#) held by the animator

5.10.3.9 Cal3DModelWrapper * GetModelWrapper ()

Returns The [Cal3DModelWrapper](#) held by the animator

5.10.3.10 const osg::Node * GetNode () const

Returns the osg::Node created by the builder on LoadModel

5.10.3.11 osg::Node * GetNode ()

Returns the osg::Node created by the builder on LoadModel

5.10.3.12 const SequenceMixer & GetSequenceMixer () const

Returns The [SequenceMixer](#) used to play, clear, and register new animations

5.10.3.13 SequenceMixer & GetSequenceMixer ()

Returns The [SequenceMixer](#) used to play, clear, and register new animations

5.10.3.14 bool LoadModel (const std::string & pFilename)

This function loads a character XML file from string, on loading it creates a [Cal3DAnimator](#) with the [Cal3DModelWrapper](#) and then calls CreateGeode() on the [AnimNodeBuilder](#). Parameters

the name of the file to load

Returns whether or not we successfully loaded the file

5.10.3.15 void PlayAnimation (const std::string & pAnim)

This function plays the specified animation defined within the character XML. Parameters

The name of the animation to play

5.10.3.16 void SetAttachmentController (AttachmentController & newController)

Assigns a new [AttachmentController](#). One is created by default, so this is provided to allow a developer to subclass the controller and assign the new one to the helper.

5.10.3.17 void SetGroundClamp (bool b)

Set whether or not this entity should be ground clamped.

5.10.3.18 void Update (float dt) [virtual]

The user should call [Update\(\)](#) on a per frame basis this function updates the sequence mixer and the [Cal3DAnimator](#).

5.10.4 Member Data Documentation**5.10.4.1 const std::string PROPERTY_SKELETAL_MESH [static]**

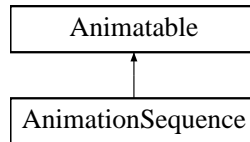
The documentation for this class was generated from the following files:

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

5.11 AnimationSequence Class Reference

[AnimationSequence](#) derives from [Animatable](#) and contains a child list of animations to play.

#include <inc/dtAnim/animationsequence.h> Inheritance diagram for AnimationSequence::



Classes

- class [AnimationController](#)
AnimationController is responsible for updating the sequences child Animatables.

Public Types

- typedef std::list< dtCore::RefPtr< [Animatable](#) > > [AnimationContainer](#)
- typedef AnimationContainer::allocator_type::value_type [ContainerType](#)

Public Member Functions

- [AnimationSequence](#) ([AnimationController](#) *)
- [AnimationSequence](#) ()
- void [AddAnimation](#) ([Animatable](#) *pAnimation)
Add an animation as a child of this sequence.
- void [ClearAnimation](#) (const std::string &pAnimName, float fadeTime)
Fade out an animation by name.
- virtual dtCore::RefPtr< [Animatable](#) > [Clone](#) ([Cal3DModelWrapper](#) *modelWrapper) const
This function copies sequence and all child Animatables.
- void [ForceFadeOut](#) (float time)
Force fade out will make this animation and all child animations fade out over time.
- const [Animatable](#) * [GetAnimation](#) (const std::string &pAnimName) const
- [Animatable](#) * [GetAnimation](#) (const std::string &pAnimName)
Get a child animation by name.
- const [AnimationContainer](#) & [GetChildAnimations](#) () const
- [AnimationContainer](#) & [GetChildAnimations](#) ()
Get a reference to the child animations.
- const [AnimationController](#) * [GetController](#) () const
- [AnimationController](#) * [GetController](#) ()
Get the controller for this sequence.
- bool [IsAnimationPlaying](#) (const std::string &pAnim) const
- void [Prune](#) ()
Prune is called before the animation is deleted.
- void [Recalculate](#) ()
Recalculate is called on PlayAnimation() it calculates the start and end times of our animation.

- void **SetController** ([AnimationController](#) *pController)
Override the default controller.
- void **Update** (float dt)
Our virtual update function.

Protected Member Functions

- **AnimationSequence** (const [AnimationSequence](#) &, [Cal3DModelWrapper](#) *wrapper)
No one should be calling the copy constructor except the clone method.
- **~AnimationSequence** ()
- **AnimationSequence** & **operator=** (const [AnimationSequence](#) &)
Hide this to prevent assigning.

5.11.1 Detailed Description

[AnimationSequence](#) derives from [Animatable](#) and contains a child list of animations to play. The animations can be AnimationChannels or other AnimationSequences.

5.11.2 Member Typedef Documentation

5.11.2.1 **typedef** std::list<dtCore::RefPtr<[Animatable](#)> > **AnimationContainer**

5.11.2.2 **typedef** [AnimationContainer](#)::allocator_type::value_type **ContainerType**

5.11.3 Constructor & Destructor Documentation

5.11.3.1 **AnimationSequence** ()

5.11.3.2 **AnimationSequence** ([AnimationController](#) * pController)

5.11.3.3 **~AnimationSequence** () [protected]

5.11.3.4 **AnimationSequence** (const [AnimationSequence](#) & pSeq, [Cal3DModelWrapper](#) * wrapper) [protected]

No one should be calling the copy constructor except the clone method.

5.11.4 Member Function Documentation

5.11.4.1 **void AddAnimation** ([Animatable](#) * pAnimation)

Add an animation as a child of this sequence. Parameters

the child animatable

5.11.4.2 **void ClearAnimation** (const std::string & pAnimName, float fadeTime)

Fade out an animation by name. Parameters

time in seconds to fade out over

5.11.4.3 **dtCore::RefPtr< Animatable > Clone** ([Cal3DModelWrapper](#) * modelWrapper) const [virtual]

This function copies sequence and all child Animatables.

Implements [Animatable](#).

5.11.4.4 void ForceFadeOut (float *time*) [virtual]

Force fade out will make this animation and all child animations fade out over time. Parameters

the time to fade out over

Implements [Animatable](#).

5.11.4.5 const Animatable * GetAnimation (const std::string & *pAnimName*) const**5.11.4.6 Animatable * GetAnimation (const std::string & *pAnimName*)**

Get a child animation by name. Returns the child animation, or 0 if it isnt a child

5.11.4.7 const AnimationSequence::AnimationContainer & GetChildAnimations () const**5.11.4.8 AnimationSequence::AnimationContainer & GetChildAnimations ()**

Get a reference to the child animations. Returns the container of child animations

5.11.4.9 const AnimationSequence::AnimationController * GetController () const**5.11.4.10 AnimationSequence::AnimationController * GetController ()**

Get the controller for this sequence. Returns the controller for this sequence

5.11.4.11 bool IsAnimationPlaying (const std::string & *pAnim*) const

Returns whether or not the specified animation is playing

5.11.4.12 AnimationSequence& operator= (const AnimationSequence &) [protected]

Hide this to prevent assigning.

5.11.4.13 void Prune () [virtual]

Prune is called before the animation is deleted.

Implements [Animatable](#).

5.11.4.14 void Recalculate () [virtual]

Recalculate is called on PlayAnimation() it calculates the start and end times of our animation.

Implements [Animatable](#).

5.11.4.15 void SetController (AnimationController * *pController*)

Override the default controller. Parameters

the new controller to use

5.11.4.16 void Update (float *dt*) [virtual]

Our virtual update function. Parameters

delta time

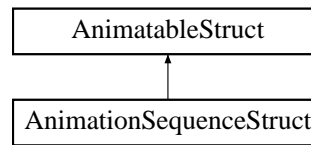
Implements [Animatable](#).

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

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

5.12 AnimationSequenceStruct Struct Reference

#include <inc/dtAnim/characterfilehandler.h>Inheritance diagram for AnimationSequenceStruct:



Public Member Functions

- [AnimationSequenceStruct \(\)](#)

Public Attributes

- `std::vector< std::string >` [mChildNames](#)

5.12.1 Constructor & Destructor Documentation

5.12.1.1 AnimationSequenceStruct ()

5.12.2 Member Data Documentation

5.12.2.1 `std::vector<std::string>` mChildNames

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

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

5.13 AnimationStruct Struct Reference

structure to contain all info related to an animation

```
#include <inc/dtAnim/characterfilehandler.h>
```

Public Attributes

- `std::string mFileName`
The filename of the cal3D animation.
- `std::string mName`
The user friendly name of this animation.

5.13.1 Detailed Description

structure to contain all info related to an animation

5.13.2 Member Data Documentation

5.13.2.1 `std::string mFileName`

The filename of the cal3D animation.

5.13.2.2 `std::string mName`

The user friendly name of this animation.

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

- [characterfilehandler.h](#)

5.14 AnimationWrapper Class Reference

The [AnimationWrapper](#) is meant to be a wrapper around a Cal3D animation.

```
#include <inc/dtAnim/animationwrapper.h>
```

Public Member Functions

- [AnimationWrapper](#) (const std::string &pName, int pAnimationID)
- float [GetDuration](#) () const
The duration of an animation is defined to be the amount of time it takes to play through a single cycle.
- int [GetID](#) () const
The ID of the Animation Wrapper refers to the Cal3D animation ID.
- const std::string & [GetName](#) () const
The name of this animation as it is defined in the XML file.
- float [GetSpeed](#) () const
The speed of an animation is defined to be a percentage of the export speed, defaulting to 1.0.
- void [SetDuration](#) (float duration)
- void [SetName](#) (const std::string &pName)
- void [SetSpeed](#) (float pSpeed)

Protected Member Functions

- virtual [~AnimationWrapper](#) ()

5.14.1 Detailed Description

The [AnimationWrapper](#) is meant to be a wrapper around a Cal3D animation. The wrapper is used to keep our API from passing Cal3D specific objects and to attach additional information to the animation.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 [AnimationWrapper](#) (const std::string &pName, int pAnimationID)

5.14.2.2 [~AnimationWrapper](#) () [protected, virtual]

5.14.3 Member Function Documentation

5.14.3.1 float [GetDuration](#) () const

The duration of an animation is defined to be the amount of time it takes to play through a single cycle.

5.14.3.2 int [GetID](#) () const

The ID of the Animation Wrapper refers to the Cal3D animation ID. Returns the Cal3D animation ID

5.14.3.3 const std::string & [GetName](#) () const

The name of this animation as it is defined in the XML file.

5.14.3.4 float [GetSpeed](#) () const

The speed of an animation is defined to be a percentage of the export speed, defaulting to 1.0.

5.14.3.5 void SetDuration (float *duration*)

5.14.3.6 void SetName (const std::string & *pName*)

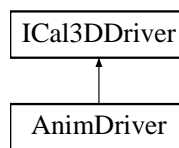
5.14.3.7 void SetSpeed (float *pSpeed*)

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

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

5.15 AnimDriver Class Reference

#include <inc/dtAnim/animdriver.h>Inheritance diagram for AnimDriver::



Public Member Functions

- [AnimDriver](#) ([Cal3DModelWrapper](#) *pWrapper)
- void [SetWrapper](#) ([Cal3DModelWrapper](#) *)
- void [Update](#) (double dt)

Protected Member Functions

- virtual [~AnimDriver](#) ()

5.15.1 Constructor & Destructor Documentation

5.15.1.1 [AnimDriver](#) ([Cal3DModelWrapper](#) * *pWrapper*)

5.15.1.2 [~AnimDriver](#) () [[protected](#), [virtual](#)]

5.15.2 Member Function Documentation

5.15.2.1 void [SetWrapper](#) ([Cal3DModelWrapper](#) * *pWrapper*) [[virtual](#)]

Implements [ICal3DDriver](#).

5.15.2.2 void [Update](#) (double *dt*) [[virtual](#)]

Implements [ICal3DDriver](#).

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

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

5.16 AnimNodeBuilder Class Reference

Class used to generate the renderable geometry for an animated character.

```
#include <inc/dtAnim/animnodebuilder.h>
```

Classes

- class **Array**
- class [Cal3DBoundingBoxSphereCalculator](#)

Public Types

- typedef dtUtil::Functor< dtCore::RefPtr< osg::Node >, TYPELIST_1([Cal3DModelWrapper](#) *) [CreateFunc](#) >

Prototype of the Create method.

Public Member Functions

- [AnimNodeBuilder](#) (const [CreateFunc](#) &pCreate)
- [AnimNodeBuilder](#) ()
- virtual dtCore::RefPtr< osg::Node > [CreateHardware](#) ([Cal3DModelWrapper](#) *pWrapper)
- dtCore::RefPtr< osg::Node > [CreateNode](#) ([Cal3DModelWrapper](#) *pWrapper)
Create the node that holds the animated character's geometry.
- virtual dtCore::RefPtr< osg::Node > [CreateNULL](#) ([Cal3DModelWrapper](#) *pWrapper)
- virtual dtCore::RefPtr< osg::Node > [CreateSoftware](#) ([Cal3DModelWrapper](#) *pWrapper)
- virtual dtCore::RefPtr< osg::Node > [CreateSoftwareNoVBO](#) ([Cal3DModelWrapper](#) *pWrapper)
- [CreateFunc](#) & [GetCreate](#) ()
- void [SetCreate](#) (const [CreateFunc](#) &pCreate)
Set a custom CreateFunc for the [AnimNodeBuilder](#).
- bool [SupportsHardware](#) () const
Does the hardware support hardware skinning?
- bool [SupportsSoftware](#) () const
Does the hardware support software skinning?

Protected Member Functions

- [AnimNodeBuilder](#) (const [AnimNodeBuilder](#) &)
- virtual [~AnimNodeBuilder](#) ()
- virtual dtCore::RefPtr< osg::Node > [CreateSoftwareInternal](#) ([Cal3DModelWrapper](#) *pWrapper, bool vbo)
- dtCore::ShaderProgram * [LoadShaders](#) ([Cal3DModelData](#) &modelData, osg::Geode &geode) const
- [AnimNodeBuilder](#) & [operator=](#) (const [AnimNodeBuilder](#) &)

5.16.1 Detailed Description

Class used to generate the renderable geometry for an animated character. This class makes use of a pointer to a function which will do the actual building of the geometry. By default, [AnimNodeBuilder](#) will try to use the default [CreateSoftware\(\)](#), [CreateHardware](#), or [CreateNULL\(\)](#) methods. Call [SetCreate\(\)](#) to supply a custom create method.

5.16.2 Member Typedef Documentation

5.16.2.1 typedef dtUtil::Functor<dtCore::RefPtr<osg::Node>, TYPELIST_1(Cal3DModelWrapper*) CreateFunc)

Prototype of the Create method. Returns the Node containing the animated character's geometry.

```
dtCore::RefPtr<osg::Node> MyCreateFunc(osg::Geode& geode, Cal3DModelWrapper* wrapper);
```

See also [CreateNode\(\)](#)

5.16.3 Constructor & Destructor Documentation

5.16.3.1 AnimNodeBuilder ()

5.16.3.2 AnimNodeBuilder (const CreateFunc & pCreate)

5.16.3.3 ~AnimNodeBuilder () [protected, virtual]

5.16.3.4 AnimNodeBuilder (const AnimNodeBuilder &) [protected]

5.16.4 Member Function Documentation

5.16.4.1 dtCore::RefPtr< osg::Node > CreateHardware (Cal3DModelWrapper * pWrapper) [virtual]

5.16.4.2 dtCore::RefPtr< osg::Node > CreateNode (Cal3DModelWrapper * pWrapper)

Create the node that holds the animated character's geometry. Will return a valid Node which contains temporary geometry. Actual character geometry might not be present until the returned node is rendered in a Scene. Parameters

pWrapper : Pointer to the [Cal3DModelWrapper](#) to be used when building the geometry.

Returns : RefPtr of a osg::Node which will contain the renderable geometry. Temporary geometry is a osg::Group with a child osg::Geode which contains one osg::Drawable that has a Drawcallback assigned to it.

Add a temporary rendered shape with a draw callback to a Group. The callback will postpone the creation of the real geometry until a valid openGL context is available.

5.16.4.3 dtCore::RefPtr< osg::Node > CreateNULL (Cal3DModelWrapper * pWrapper) [virtual]

5.16.4.4 dtCore::RefPtr< osg::Node > CreateSoftware (Cal3DModelWrapper * pWrapper) [virtual]

5.16.4.5 dtCore::RefPtr< osg::Node > CreateSoftwareInternal (Cal3DModelWrapper * pWrapper, bool vbo) [protected, virtual]

5.16.4.6 dtCore::RefPtr< osg::Node > CreateSoftwareNoVBO (Cal3DModelWrapper * pWrapper) [virtual]

5.16.4.7 AnimNodeBuilder::CreateFunc & GetCreate ()

Returns the create function

5.16.4.8 dtCore::ShaderProgram * LoadShaders (Cal3DModelData & modelData, osg::Geode & geode) const [protected]

5.16.4.9 AnimNodeBuilder& operator= (const AnimNodeBuilder &) [protected]

5.16.4.10 void SetCreate (const CreateFunc & pCreate)

Set a custom CreateFunc for the [AnimNodeBuilder](#). Function must remove the temporary geometry and drawcallback supplied by CreateNode. Parameters

pCreate : the custom create function.

5.16.4.11 bool SupportsHardware () const

Does the hardware support hardware skinning?

5.16.4.12 bool SupportsSoftware () const

Does the hardware support software skinning?

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

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

5.17 AnimSeqForceFade Class Reference

Public Member Functions

- [AnimSeqForceFade](#) (float time)
- `template<typename T >`
void [operator\(\)](#) (T &pChild)

5.17.1 Constructor & Destructor Documentation

5.17.1.1 [AnimSeqForceFade](#) (float *time*) [`inline`]

5.17.2 Member Function Documentation

5.17.2.1 `void operator()` (T &*pChild*) [`inline`]

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

- [animationsequence.cpp](#)

5.18 AnimSequenceUpdater Struct Reference

Public Member Functions

- [AnimSequenceUpdater](#) (float dt, float parent_weight)
- `template<typename T >`
void [operator\(\)](#) (T &pChild)

5.18.1 Constructor & Destructor Documentation

5.18.1.1 [AnimSequenceUpdater](#) (float *dt*, float *parent_weight*) [`inline`]

5.18.2 Member Function Documentation

5.18.2.1 `void operator()` (T &*pChild*) [`inline`]

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

- [animationsequence.cpp](#)

5.19 AttachmentController Class Reference

Stores a list of attachments for a cal model and can update their positions based each frame based on the position of the bones.

```
#include <inc/dtAnim/attachmentcontroller.h>
```

Public Types

- typedef std::vector< [AttachmentPair](#) > [AttachmentContainer](#)

Public Member Functions

- [AttachmentController](#) ()
- void [AddAttachment](#) (dtCore::Transformable &actor, dtUtil::HotSpotDefinition &spot)
Adds a hot spot attachment to the skeleton.
- const [AttachmentContainer](#) & [GetAttachments](#) () const
- void [RemoveAttachment](#) (const dtCore::Transformable &actor)
Removes a previously added hot spot attachment.
- virtual void [Update](#) ([Cal3DModelWrapper](#) &model)
Update the attachments to the new positions based on the model.

Protected Member Functions

- virtual [~AttachmentController](#) ()

5.19.1 Detailed Description

Stores a list of attachments for a cal model and can update their positions based each frame based on the position of the bones. It may be subclassed to add additional features.

5.19.2 Member Typedef Documentation

5.19.2.1 typedef std::vector<AttachmentPair> AttachmentContainer

5.19.3 Constructor & Destructor Documentation

5.19.3.1 AttachmentController ()

5.19.3.2 ~AttachmentController () [protected, virtual]

5.19.4 Member Function Documentation

5.19.4.1 void AddAttachment (dtCore::Transformable & actor, dtUtil::HotSpotDefinition & spot)

Adds a hot spot attachment to the skeleton. This will move the attachment each time the skeleton is updated. See also dtCore::HotSpotAttachment

5.19.4.2 const AttachmentController::AttachmentContainer & GetAttachments () const

Returns an immutable container holding the current set of hot spots.

5.19.4.3 void RemoveAttachment (const dtCore::Transformable & actor)

Removes a previously added hot spot attachment. If the spot is not in the container, this call is a no-op. See also dtCore::HotSpotAttachment

5.19.4.4 void Update (Cal3DModelWrapper & *model*) [virtual]

Update the attachments to the new positions based on the model. This may be overridden in a subclass to modify or update the behavior.

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

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

5.20 AttachmentMover Class Reference

This is a helper functor for moving attachments.

```
#include <inc/dtAnim/attachmentcontroller.h>
```

Public Member Functions

- [AttachmentMover](#) (const [AttachmentMover](#) &same)
- [AttachmentMover](#) (const [dtAnim::Cal3DModelWrapper](#) &model)
- void [operator\(\)](#) ([AttachmentPair](#) &attachment)
For use with AttachmentPair typedef above using for_each or similar.
- void [operator\(\)](#) (dtCore::RefPtr< dtCore::HotSpotAttachment > &attachment)
For use with dtCore::HotSpotAttachments using for_each or similar This method takes a ref ptr because containers tend to hold then, and it is for use with for_each or something similar.
- [AttachmentMover](#) & [operator=](#) (const [AttachmentMover](#) &same)

5.20.1 Detailed Description

This is a helper functor for moving attachments. It's exposed to allow use in other classes.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 AttachmentMover (const dtAnim::Cal3DModelWrapper & model)

5.20.2.2 AttachmentMover (const AttachmentMover & same)

5.20.3 Member Function Documentation

5.20.3.1 void operator() (AttachmentPair & attachment)

For use with AttachmentPair typedef above using for_each or similar. This method technically passes a refptr as a parameter, but it won't be copied because its a member of the pair which is passed by reference.

5.20.3.2 void operator() (dtCore::RefPtr< dtCore::HotSpotAttachment > & attachment)

For use with dtCore::HotSpotAttachments using for_each or similar This method takes a ref ptr because containers tend to hold then, and it is for use with for_each or something similar. The ref ptr is takes by reference, so it is not copied.

5.20.3.3 AttachmentMover & operator= (const AttachmentMover & same)

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

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

5.21 BaseReferenceBlend Struct Reference

```
#include <inc/dtAnim/posemeshutility.h>
```

Public Attributes

- float [blendAlpha](#)
- int [endAnimID](#)
- osg::Vec3 [endDirection](#)
- int [startAnimID](#)
- osg::Vec3 [startDirection](#)

5.21.1 Member Data Documentation

5.21.1.1 float blendAlpha

5.21.1.2 int endAnimID

5.21.1.3 osg::Vec3 endDirection

5.21.1.4 int startAnimID

5.21.1.5 osg::Vec3 startDirection

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

- [posemeshutility.h](#)

5.22 Cal3DAnimator Class Reference

```
#include <inc/dtAnim/cal3d animator.h>
```

Public Member Functions

- [Cal3DAnimator](#) ([Cal3DModelWrapper](#) *pWrapper)
- [ICal3DDriver](#) * [GetAnimationDriver](#) () const
- [ICal3DDriver](#) * [GetMorphTargetDriver](#) () const
- [ICal3DDriver](#) * [GetPhysiqueDriver](#) () const
- [ICal3DDriver](#) * [GetPostDriver](#) () const
- [ICal3DDriver](#) * [GetPreDriver](#) () const
- [ICal3DDriver](#) * [GetSkeletonDriver](#) () const
- [ICal3DDriver](#) * [GetSpringDriver](#) () const
- const [Cal3DModelWrapper](#) * [GetWrapper](#) () const
- [Cal3DModelWrapper](#) * [GetWrapper](#) ()
- void [SetAnimationDriver](#) ([ICal3DDriver](#) *pDriver)
- void [SetMorphTargetDriver](#) ([ICal3DDriver](#) *pDriver)
- void [SetPhysiqueDriver](#) ([ICal3DDriver](#) *pDriver)
- void [SetPostDriver](#) ([ICal3DDriver](#) *pDriver)
- void [SetPreDriver](#) ([ICal3DDriver](#) *pDriver)
- void [SetSkeletonDriver](#) ([ICal3DDriver](#) *pDriver)
- void [SetSpringDriver](#) ([ICal3DDriver](#) *pDriver)
- void [SetWrapper](#) ([Cal3DModelWrapper](#) *wrapper)

Give the Animator a new [Cal3DModelWrapper](#) to operate on.

- void [Update](#) (double dt)

Protected Member Functions

- virtual [~Cal3DAnimator](#) ()

5.22.1 Constructor & Destructor Documentation

5.22.1.1 **Cal3DAnimator** (**Cal3DModelWrapper** * *pWrapper*)

5.22.1.2 **~Cal3DAnimator** () [**protected**, **virtual**]

5.22.2 Member Function Documentation

5.22.2.1 **ICal3DDriver*** **GetAnimationDriver** () **const** [**inline**]

5.22.2.2 **ICal3DDriver*** **GetMorphTargetDriver** () **const** [**inline**]

5.22.2.3 **ICal3DDriver*** **GetPhysiqueDriver** () **const** [**inline**]

5.22.2.4 **ICal3DDriver*** **GetPostDriver** () **const** [**inline**]

5.22.2.5 **ICal3DDriver*** **GetPreDriver** () **const** [**inline**]

5.22.2.6 **ICal3DDriver*** **GetSkeletonDriver** () **const** [**inline**]

5.22.2.7 **ICal3DDriver*** **GetSpringDriver** () **const** [**inline**]

5.22.2.8 **const Cal3DModelWrapper** * **GetWrapper** () **const**

5.22.2.9 **Cal3DModelWrapper** * **GetWrapper** ()

5.22.2.10 **void** **SetAnimationDriver** (**ICal3DDriver** * *pDriver*)

5.22.2.11 **void** **SetMorphTargetDriver** (**ICal3DDriver** * *pDriver*)

5.22.2.12 **void** **SetPhysiqueDriver** (**ICal3DDriver** * *pDriver*)

5.22.2.13 **void** **SetPostDriver** (**ICal3DDriver** * *pDriver*)

5.22.2.14 **void** **SetPreDriver** (**ICal3DDriver** * *pDriver*)

5.22.2.15 **void** **SetSkeletonDriver** (**ICal3DDriver** * *pDriver*)

5.22.2.16 **void** **SetSpringDriver** (**ICal3DDriver** * *pDriver*)

5.22.2.17 **void** **SetWrapper** (**Cal3DModelWrapper** * *wrapper*)

Give the Animator a new [Cal3DModelWrapper](#) to operate on. Will pass the wrapper unto any ICal3DDrivers that may be present.

Parameters

wrapper : the new [Cal3DModelWrapper](#) this instance should use.

5.22.2.18 **void** **Update** (**double** *dt*)

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

- [cal3d animator.h](#)
- [cal3d animator.cpp](#)

5.23 Cal3DBoundingSphereCalculator Class Reference

```
#include <inc/dtAnim/animnodebuilder.h>
```

Public Member Functions

- [Cal3DBoundingSphereCalculator](#) ([Cal3DModelWrapper](#) &wrapper)
- [osg::BoundingSphere](#) [computeBound](#) (const [osg::Node](#) &) const

5.23.1 Constructor & Destructor Documentation

5.23.1.1 Cal3DBoundingSphereCalculator (Cal3DModelWrapper & wrapper)

5.23.2 Member Function Documentation

5.23.2.1 [osg::BoundingSphere](#) [computeBound](#) (const [osg::Node](#) &) const

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

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

5.24 Cal3DDatabase Class Reference

```
#include <inc/dtAnim/cal3ddatabase.h>
```

Public Types

- typedef std::vector< dtCore::RefPtr< [Cal3DModelData](#) > > [ModelDataArray](#)

Public Member Functions

- [Cal3DModelData](#) * [GetModelData](#) (const [Cal3DModelWrapper](#) &wrapper)
Get the model data associated with this model wrapper.
- const [Cal3DModelData](#) * [GetModelData](#) (const [Cal3DModelWrapper](#) &wrapper) const
Get the model data associated with this model wrapper.
- [AnimNodeBuilder](#) & [GetNodeBuilder](#) ()
- dtCore::RefPtr< [Cal3DModelWrapper](#) > [Load](#) (const std::string &filename)
Load an animated entity definition file and return the [Cal3DModelWrapper](#).
- void [PurgeLoaderCaches](#) ()
- void [TruncateDatabase](#) ()

Static Public Member Functions

- static [Cal3DDatabase](#) & [GetInstance](#) ()
Sigh, yes it's a singleton.

Protected Member Functions

- [Cal3DDatabase](#) ()
- virtual ~[Cal3DDatabase](#) ()
- const [Cal3DModelData](#) * [Find](#) (const [CalCoreModel](#) *coreModel) const
- const [Cal3DModelData](#) * [Find](#) (const std::string &filename) const
- [Cal3DModelData](#) * [Find](#) (const [CalCoreModel](#) *coreModel)
- [Cal3DModelData](#) * [Find](#) (const std::string &filename)

Protected Attributes

- dtCore::RefPtr< [Cal3DLoader](#) > [mFileLoader](#)
- [ModelDataArray](#) [mModelData](#)
- dtCore::RefPtr< [AnimNodeBuilder](#) > [mNodeBuilder](#)

Static Protected Attributes

- static dtCore::RefPtr< [Cal3DDatabase](#) > [mInstance](#)

5.24.1 Member Typedef Documentation

5.24.1.1 `typedef std::vector<dtCore::RefPtr<Cal3DModelData> > ModelDataArray`

5.24.2 Constructor & Destructor Documentation

5.24.2.1 `Cal3DDatabase ()` [protected]

5.24.2.2 `~Cal3DDatabase ()` [protected, virtual]

5.24.3 Member Function Documentation

5.24.3.1 `const Cal3DModelData * Find (const CalCoreModel * coreModel) const` [protected]

5.24.3.2 `const Cal3DModelData * Find (const std::string & filename) const` [protected]

5.24.3.3 `Cal3DModelData * Find (const CalCoreModel * coreModel)` [protected]

5.24.3.4 `Cal3DModelData * Find (const std::string & filename)` [protected]

5.24.3.5 `Cal3DDatabase & GetInstance ()` [static]

Sigh, yes it's a singleton.

5.24.3.6 `Cal3DModelData * GetModelData (const Cal3DModelWrapper & wrapper)`

Get the model data associated with this model wrapper.

5.24.3.7 `const Cal3DModelData * GetModelData (const Cal3DModelWrapper & wrapper) const`

Get the model data associated with this model wrapper.

5.24.3.8 `AnimNodeBuilder & GetNodeBuilder ()`

Returns the node builder for this database.

5.24.3.9 `dtCore::RefPtr< Cal3DModelWrapper > Load (const std::string & filename)`

Load an animated entity definition file and return the [Cal3DModelWrapper](#).

5.24.3.10 `void PurgeLoaderCaches ()`

5.24.3.11 `void TruncateDatabase ()`

5.24.4 Member Data Documentation

5.24.4.1 `dtCore::RefPtr<Cal3DLoader> mFileLoader` [protected]

5.24.4.2 `dtCore::RefPtr< Cal3DDatabase > mInstance` [static, protected]

5.24.4.3 `ModelDataArray mModelData` [protected]

5.24.4.4 `dtCore::RefPtr<AnimNodeBuilder> mNodeBuilder` [protected]

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

- [cal3ddatabase.h](#)
- [cal3ddatabase.cpp](#)

5.25 Cal3DGameActor Class Reference

This class is the game actor for an animated model.

```
#include <inc/dtAnim/cal3dgameactor.h>
```

Classes

- struct [PropertyNames](#)
string constants for this actor

Public Types

- enum [RenderModeBits](#) { [RENDER_MODE_NONE](#) = 0, [RENDER_MODE_SKIN](#) = 1<<0, [RENDER_MODE_BONES](#) = 1<<1 }
- used to describe what to render.*

Public Member Functions

- [Cal3DGameActor](#) (dtGame::GameActorProxy &proxy)
Constructs a [Cal3DGameActor](#) actor.
- virtual void [AddedToScene](#) (dtCore::Scene *scene)
Called when the actor has been added to the game manager.
- void [ApplyAnimationGroup](#) (const dtDAL::NamedGroupParameter &prop)
- const dtAnim::Cal3DAnimator * [GetAnimator](#) () const
- dtAnim::Cal3DAnimator * [GetAnimator](#) ()
- int [GetRenderMode](#) () const
- dtCore::RefPtr< dtDAL::NamedGroupParameter > [MakeAnimationGroup](#) ()
- virtual void [OnEnteredWorld](#) ()
- virtual void [OnTickLocal](#) (const dtGame::TickMessage &tickMessage)
- virtual void [SetModel](#) (const std::string &modelFile)
Loads a model file.
- void [SetRenderMode](#) (int bits)

Protected Types

- typedef unsigned char [RenderModeBitContainer](#)

Protected Member Functions

- virtual [~Cal3DGameActor](#) ()
Destroys this actor.

Protected Attributes

- dtCore::RefPtr< dtAnim::Cal3DAnimator > [mAnimator](#)
- dtCore::RefPtr< osg::Geode > [mModelGeode](#)
- dtCore::RefPtr< dtAnim::Cal3DDatabase > [mModelLoader](#)
- [RenderModeBitContainer](#) [mRenderModeBits](#)
- dtCore::RefPtr< osg::Geode > [mSkeletalGeode](#)
support to visualize the skeletal.

5.25.1 Detailed Description

This class is the game actor for an animated model. See also [GameActor](#)
[Cal3DGameActorProxy](#)

5.25.2 Member Typedef Documentation

5.25.2.1 `typedef unsigned char RenderModeBitContainer [protected]`

5.25.3 Member Enumeration Documentation

5.25.3.1 `enum RenderModeBits`

used to describe what to render.

Todo

should this be a `dtUtil::Enumeration`?

Enumerator:

RENDER_MODE_NONE
RENDER_MODE_SKIN
RENDER_MODE_BONES

5.25.4 Constructor & Destructor Documentation

5.25.4.1 `Cal3DGameActor (dtGame::GameActorProxy & proxy)`

Constructs a [Cal3DGameActor](#) actor. Parameters

proxy The actor proxy owning this task actor.

5.25.4.2 `~Cal3DGameActor () [protected, virtual]`

Destroys this actor.

5.25.5 Member Function Documentation

5.25.5.1 `void AddedToScene (dtCore::Scene * scene) [virtual]`

Called when the actor has been added to the game manager.

5.25.5.2 `void ApplyAnimationGroup (const dtDAL::NamedGroupParameter & prop)`

5.25.5.3 `const dtAnim::Cal3DAnimator * GetAnimator () const`

5.25.5.4 `dtAnim::Cal3DAnimator * GetAnimator ()`

5.25.5.5 `int GetRenderMode () const`

5.25.5.6 `dtCore::RefPtr< dtDAL::NamedGroupParameter > MakeAnimationGroup ()`

Todo

add all animations currently being blended

5.25.5.7 `void OnEnteredWorld () [virtual]`

5.25.5.8 `void OnTickLocal (const dtGame::TickMessage & tickMessage) [virtual]`

5.25.5.9 `void SetModel (const std::string & modelFile) [virtual]`

Loads a model file. Parameters

modelFile The filename of the model to load.

Force generation of first mesh

5.25.5.10 void SetRenderMode (int *bits*)

5.25.6 Member Data Documentation

5.25.6.1 dtCore::RefPtr<dtAnim::Cal3DAnimator> mAnimator [protected]

5.25.6.2 dtCore::RefPtr<osg::Geode> mModelGeode [protected]

5.25.6.3 dtCore::RefPtr<dtAnim::Cal3DDatabase> mModelLoader [protected]

5.25.6.4 RenderModeBitContainer mRenderModeBits [protected]

5.25.6.5 dtCore::RefPtr<osg::Geode> mSkeletalGeode [protected]

support to visualize the skeletal.

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

- [cal3dgameactor.h](#)
- [cal3dgameactor.cpp](#)

5.26 Cal3DGameActorProxy Class Reference

This class is the proxy for an animated model game object.

```
#include <inc/dtAnim/cal3dgameactor.h>
```

Public Member Functions

- [Cal3DGameActorProxy \(\)](#)
Constructs the proxy.
- virtual void [BuildInvokables \(\)](#)
Registers any invokables used by the proxy.
- virtual void [BuildPropertyMap \(\)](#)
Builds the property map for the task actor proxy.
- virtual dtDAL::ActorProxyIcon * [GetBillBoardIcon \(\)](#)
Gets the billboard used to represent static mesh if this proxy's render mode is RenderMode::DRAW_BILLBOARD_ICON.
- virtual const dtDAL::ActorProxy::RenderMode & [GetRenderMode \(\)](#)
Gets the method by which this static mesh is rendered.

Protected Member Functions

- virtual [~Cal3DGameActorProxy \(\)](#)
Destroys the proxy.
- virtual void [CreateActor \(\)](#)
Called by the game manager during creation of the proxy.

5.26.1 Detailed Description

This class is the proxy for an animated model game object. See also [GameActorProxy](#) [Cal3DGameActor](#)

5.26.2 Constructor & Destructor Documentation

5.26.2.1 Cal3DGameActorProxy ()

Constructs the proxy.

5.26.2.2 ~Cal3DGameActorProxy () [protected, virtual]

Destroys the proxy.

5.26.3 Member Function Documentation

5.26.3.1 void BuildInvokables () [virtual]

Registers any invokables used by the proxy. The invokables allow the actor to hook into the game manager messages system.

5.26.3.2 void BuildPropertyMap () [virtual]

Builds the property map for the task actor proxy. These properties wrap the specified properties located in the actor.

Todo

make a UChar actor property and use it here.

5.26.3.3 void CreateActor () [protected, virtual]

Called by the game manager during creation of the proxy. This method creates the real actor and returns it.

5.26.3.4 dtDAL::ActorProxyIcon * GetBillBoardIcon () [virtual]

Gets the billboard used to represent static mesh if this proxy's render mode is RenderMode::DRAW_BILLBOARD_ICON. Used by STAGE. Returns

5.26.3.5 const dtDAL::ActorProxy::RenderMode & GetRenderMode () [virtual]

Gets the method by which this static mesh is rendered. This is used by STAGE. Returns If there is no geometry currently assigned, this method will return RenderMode::DRAW_BILLBOARD_ICON. If there is geometry assigned to this static mesh, RenderMode::DRAW_ACTOR is returned.

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

- [cal3dgameactor.h](#)
- [cal3dgameactor.cpp](#)

5.27 Cal3DLoader Class Reference

Loads a animation definition file and returns a valid CalModel.

```
#include <inc/dtAnim/cal3dloader.h>
```

Public Member Functions

- [Cal3DLoader](#) ()
- [bool Load](#) (const std::string &filename, [Cal3DModelData](#) *&data_in)
Load an animated entity definition file and return the [Cal3DModelWrapper](#).
- [void PurgeAllCaches](#) ()
empty all containers of CalCoreModels and the stored textures

Protected Member Functions

- virtual [~Cal3DLoader](#) ()

5.27.1 Detailed Description

Loads a animation definition file and returns a valid CalModel. Caches the CalCoreModel defined by the file to make it faster to create additional instances of CalModels. If you call [Load\(\)](#) with the same filename twice, it actually only loads once.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 [Cal3DLoader](#) ()

5.27.2.2 [~Cal3DLoader](#) () [protected, virtual]

5.27.3 Member Function Documentation

5.27.3.1 [bool Load](#) (const std::string & filename, [Cal3DModelData](#) *& data_in)

Load an animated entity definition file and return the [Cal3DModelWrapper](#). Will use the Delta3D search paths to find the supplied filename.

Will create a new [Cal3DModelWrapper](#), but you're responsible for deleting it. Note The animations are named with their filenames by default, or by an optional name attribute in the .xml file. Returns A fully defined CalModel wrapped by a [Cal3DModelWrapper](#). RefPtr could be not valid (wrapper->valid()==false) if the file didn't load correctly. See also [SetDataFilePathList\(\)](#)

Exceptions

SAXParseException If the file wasn't formatted correctly

5.27.3.2 [void PurgeAllCaches](#) ()

empty all containers of CalCoreModels and the stored textures Use if you want to completely start over with no history of previous animated entities that have been created.

This will allow you to reload files for a second time. Note : currently this will remove reference to all created osg Textures as well, which might cause the texture to be deleted.

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

- [cal3dloader.h](#)
- [cal3dloader.cpp](#)

5.28 Cal3DModelData Class Reference

```
#include <inc/dtAnim/cal3dmodeldata.h>
```

Public Types

- typedef std::vector< dtCore::RefPtr< [Animatable](#) > > [AnimatableArray](#)
- typedef std::vector< dtCore::RefPtr< [AnimationWrapper](#) > > [AnimationWrapperArray](#)

Public Member Functions

- [Cal3DModelData](#) (CalCoreModel *coreModel, const std::string &filename)
- void [Add](#) ([Animatable](#) *)
- void [Add](#) ([AnimationWrapper](#) *)
- const [AnimatableArray](#) & [GetAnimatables](#) () const
- [AnimatableArray](#) & [GetAnimatables](#) ()
- const [AnimationWrapperArray](#) & [GetAnimationWrappers](#) () const
- [AnimationWrapperArray](#) & [GetAnimationWrappers](#) ()
- const CalCoreModel * [GetCoreModel](#) () const
- CalCoreModel * [GetCoreModel](#) ()
- const std::string & [GetFilename](#) () const
- unsigned [GetIndexVBO](#) () const
- const [LODOptions](#) & [GetLODOptions](#) () const
- [LODOptions](#) & [GetLODOptions](#) ()
- const std::string & [GetPoseMeshFilename](#) () const
- const std::string & [GetShaderGroupName](#) () const
- unsigned [GetShaderMaxBones](#) () const
- const std::string & [GetShaderName](#) () const
- unsigned [GetVertexVBO](#) () const
- void [Remove](#) ([Animatable](#) *)
- void [Remove](#) ([AnimationWrapper](#) *)
- void [SetIndexVBO](#) (unsigned)
 - *Sets the id of the of the Index Vertex Buffer Object being used with this character core model.*
- void [SetPoseMeshFilename](#) (const std::string &name)
- void [SetShaderGroupName](#) (const std::string &groupName)
 - *Sets the shader group name.*
- void [SetShaderMaxBones](#) (unsigned maxBones)
 - *Sets the maximum number of bones the shader supports.*
- void [SetShaderName](#) (const std::string &name)
 - *Sets the shader group name.*
- void [SetVertexVBO](#) (unsigned)
 - *Sets the id of the of the Vertex Buffer Object being used with this character core model.*

Protected Member Functions

- [Cal3DModelData](#) (const [Cal3DModelData](#) &)
- virtual [~Cal3DModelData](#) ()
- [Cal3DModelData](#) & [operator=](#) (const [Cal3DModelData](#) &)

5.28.1 Member Typedef Documentation

5.28.1.1 `typedef std::vector<dtCore::RefPtr<Animatable> > AnimatableArray`

5.28.1.2 `typedef std::vector<dtCore::RefPtr<AnimationWrapper> > AnimationWrapperArray`

5.28.2 Constructor & Destructor Documentation

5.28.2.1 `Cal3DModelData (CalCoreModel * coreModel, const std::string & filename)`

5.28.2.2 `~Cal3DModelData ()` [protected, virtual]

5.28.2.3 `Cal3DModelData (const Cal3DModelData &)` [protected]

5.28.3 Member Function Documentation

5.28.3.1 `void Add (Animatable * anim)`

5.28.3.2 `void Add (AnimationWrapper * wrapper)`

5.28.3.3 `const Cal3DModelData::AnimatableArray & GetAnimatables () const`

5.28.3.4 `Cal3DModelData::AnimatableArray & GetAnimatables ()`

5.28.3.5 `const Cal3DModelData::AnimationWrapperArray & GetAnimationWrappers () const`

5.28.3.6 `Cal3DModelData::AnimationWrapperArray & GetAnimationWrappers ()`

5.28.3.7 `const CalCoreModel * GetCoreModel () const`

5.28.3.8 `CalCoreModel * GetCoreModel ()`

5.28.3.9 `const std::string & GetFilename () const`

5.28.3.10 `unsigned GetIndexVBO () const`

Returns the id of the of the Index Vertex Buffer Object being used with this character core model, or 0 for none

5.28.3.11 `const LODOptions& GetLODOptions () const` [inline]

5.28.3.12 `LODOptions& GetLODOptions ()` [inline]

5.28.3.13 `const std::string & GetPoseMeshFilename () const`

5.28.3.14 `const std::string & GetShaderGroupName () const`

See also `dtCore::ShaderManager` Returns the shader group used to lookup the shader for this character model.

5.28.3.15 `unsigned GetShaderMaxBones () const`

Returns the maximum number of bones the skinning shader supports.

5.28.3.16 `const std::string & GetShaderName () const`

See also `dtCore::ShaderManager`

[GetShaderGroupName](#) Returns the name of the shader within the shader group to use.

5.28.3.17 `unsigned GetVertexVBO () const`

Returns the id of the of the Vertex Buffer Object being used with this character core model, or 0 for none.

5.28.3.18 `Cal3DModelData& operator= (const Cal3DModelData &)` [protected]

5.28.3.19 `void Remove (Animatable * anim)`

5.28.3.20 `void Remove (AnimationWrapper * wrapper)`

5.28.3.21 `void SetIndexVBO (unsigned vbo)`

Sets the id of the of the Index Vertex Buffer Object being used with this character core model.

5.28.3.22 void SetPoseMeshFilename (const std::string & *name*)

5.28.3.23 void SetShaderGroupName (const std::string & *groupName*)

Sets the shader group name.

5.28.3.24 void SetShaderMaxBones (unsigned *maxBones*)

Sets the maximum number of bones the shader supports.

5.28.3.25 void SetShaderName (const std::string & *name*)

Sets the shader group name.

5.28.3.26 void SetVertexVBO (unsigned *vbo*)

Sets the id of the of the Vertex Buffer Object being used with this character core model.

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

- [cal3dmodeldata.h](#)
- [cal3dmodeldata.cpp](#)

5.29 Cal3DModelWrapper Class Reference

Wraps the Cal3D CalModel class.

```
#include <inc/dtAnim/cal3dmodelwrapper.h>
```

Public Member Functions

- [Cal3DModelWrapper](#) (CalModel *model)
- void [ApplyCoreModelScaleFactor](#) (float scaleFactor) const
Apply a scaling factor to the core model.
- bool [AttachMesh](#) (int meshID)
- bool [BeginRenderingQuery](#) ()
- bool [BlendCycle](#) (int id, float weight, float delay)
- void [ClearAll](#) (float delay=0.0)
Remove all existing animations from the mixer.
- bool [ClearCycle](#) (int id, float delay)
- bool [DetachMesh](#) (int meshID)
- void [EndRenderingQuery](#) ()
- bool [ExecuteAction](#) (int id, float delayIn, float delayOut, float weightTgt=1.0f, bool autoLock=false)
Perform a one time animation.
- void [GetAmbientColor](#) (unsigned char *colorBuffer)
- float [GetAnimationTime](#) ()
- osg::Quat [GetBoneAbsoluteRotation](#) (unsigned int boneID) const
- osg::Quat [GetBoneAbsoluteRotationForKeyFrame](#) (int animid, int boneid, unsigned int keyframeindex) const
Get the Cal3D rotation values.
- osg::Vec3 [GetBoneAbsoluteTranslation](#) (unsigned int boneID) const
Get the current translation for the CalBone.
- osg::Quat [GetBoneRelativeRotation](#) (unsigned int boneID) const
- osg::BoundingBox [GetBoundingBox](#) ()
Get a bounding box the encompasses the character in its default pose.
- const CalModel * [GetCalModel](#) () const
Get a const pointer to the internal CalModel.
- CalModel * [GetCalModel](#) ()
Get a pointer to the internal CalModel.
- int [GetCoreAnimationCount](#) () const
- float [GetCoreAnimationDuration](#) (int animID) const
Get the duration of this animation (seconds?).
- int [GetCoreAnimationIDByName](#) (const std::string &name) const
- unsigned int [GetCoreAnimationKeyframeCount](#) (int animID) const
Get the total number of keyframes in this animation.
- unsigned int [GetCoreAnimationKeyframeCountForTrack](#) (int animID, int boneID) const
Get the number of keyframes in the animation for a particular bone.
- const std::string & [GetCoreAnimationName](#) (int animID) const
Get the name that equates to the supplied animation ID.

- unsigned int [GetCoreAnimationTrackCount](#) (int animID) const
Get the number of tracks this animation uses.
- void [GetCoreBoneChildrenIDs](#) (int parentCoreBoneID, std::vector< int > &toFill) const
Get the bone IDs of all children for a parent bone.
- int [GetCoreBoneID](#) (const std::string &name) const
Get the Cal3D CoreBone ID.
- void [GetCoreBoneNames](#) (std::vector< std::string > &toFill) const
Get all bone IDs within the CoreSkeleton.
- CalCoreMaterial * [GetCoreMaterial](#) (int matID)
- osg::Vec4 [GetCoreMaterialAmbient](#) (int matID) const
Get the core material ambient color (rgba 0-255).
- int [GetCoreMaterialCount](#) () const
- osg::Vec4 [GetCoreMaterialDiffuse](#) (int matID) const
Get the core material diffuse color (rgba 0-255).
- const std::string & [GetCoreMaterialName](#) (int matID) const
Get the name associated with the material using the supplied material ID.
- float [GetCoreMaterialShininess](#) (int matID) const
Get the core material shininess.
- osg::Vec4 [GetCoreMaterialSpecular](#) (int matID) const
Get the core material specular color (rgba 0-255).
- int [GetCoreMeshCount](#) () const
- const std::string & [GetCoreMeshName](#) (int meshID) const
Get the name for the mesh using the supplied meshID.
- osg::Quat [GetCoreTrackKeyFrameQuat](#) (unsigned int animID, unsigned int boneID, unsigned int keyframeindex) const
Get the Cal3D rotation values.
- void [GetDiffuseColor](#) (unsigned char *colorBuffer)
- int [GetFaceCount](#) ()
- int [GetFaces](#) (int *faces)
- int [GetMapCount](#) ()
- void * [GetMapUserData](#) (int mapID)
- int [GetMeshCount](#) ()
- int [GetNormals](#) (float *normals, int stride=0)
- CalHardwareModel * [GetOrCreateCalHardwareModel](#) ()
Get the CAL3D CalHardwareModel representation for this CalModel.
- int [GetParentBoneID](#) (unsigned int boneID) const
Get the id of the parent to boneID.
- void [GetRootBoneIDs](#) (std::vector< int > &toFill) const
Get all the root bone IDs.
- float [GetShininess](#) ()
- void [GetSpecularColor](#) (unsigned char *colorBuffer)

- int [GetSubmeshCount](#) (int submeshID)
- int [GetTextureCoords](#) (int mapID, float *coords, int stride=0)
- int [GetVertexCount](#) ()
- int [GetVertices](#) (float *vertBuffer, int stride=0)
- bool [HasAnimation](#) (int animID) const
- bool [HasBone](#) (int boneID) const
- bool [HasTrackForBone](#) (unsigned int animID, int boneID) const
- void [HideMesh](#) (int meshID)
- bool [IsMeshVisible](#) (int meshID)
- bool [RemoveAction](#) (int id)

Remove an existing one-time animation from the mixer.

- bool [SelectMeshSubmesh](#) (int meshID, int submeshID)
- void [SetAnimationTime](#) (float time)

sets the offset time used in synchronized looping animations.

- void [SetCalModel](#) (CalModel *model)
- void [SetLODLevel](#) (float level)
- void [SetMaterialSet](#) (int materialSetID)
- void [ShowMesh](#) (int meshID)
- void [Update](#) (float deltaTime)

Update the Cal3D system using the CalModel's update.

- void [UpdateAnimation](#) (float deltaTime)

Update just the Cal3D's animation using the mixer.

- void [UpdateMorphTargetMixer](#) (float deltaTime)

Update the CalModel's morph target mixer.

- void [UpdatePhysique](#) ()

Update the CalModel's physique.

- void [UpdateSkeleton](#) ()

Update just Cal3D's skeleton using the mixer.

- void [UpdateSpringSystem](#) (float deltaTime)

Update the CalModel's spring system.

Static Public Attributes

- static const int [NULL_BONE](#) = -1

Protected Member Functions

- virtual [~Cal3DModelWrapper](#) ()

5.29.1 Detailed Description

Wraps the Cal3D CalModel class. It is expected that users will use the [Cal3DModelWrapper](#) instead of using the CalModel class directly. To create a new [Cal3DModelWrapper](#):

```
dtCore::RefPtr<Cal3DModelWrapper> wrap = new Cal3DModelWrapper( calModel );
```

5.29.2 Constructor & Destructor Documentation

5.29.2.1 Cal3DModelWrapper (CalModel * *model*)

5.29.2.2 ~Cal3DModelWrapper () [protected, virtual]

5.29.3 Member Function Documentation

5.29.3.1 void ApplyCoreModelScaleFactor (float *scaleFactor*) const

Apply a scaling factor to the core model. May need to rebuild any local geometry after calling this. Parameters

scaleFactor : amount to scale the character (2.0 = double the size)

5.29.3.2 bool AttachMesh (int *meshID*)

5.29.3.3 bool BeginRenderingQuery () [inline]

5.29.3.4 bool BlendCycle (int *id*, float *weight*, float *delay*)

Parameters

id : a valid ID of an animation (0 based)

weight : the strength of this animation in relation to the other animations already being blended.

delay : how long it takes for this animation to become full strength (seconds)

Returns true if successful, false if an error happened.

5.29.3.5 void ClearAll (float *delay* = 0.0)

Remove all existing animations from the mixer.

5.29.3.6 bool ClearCycle (int *id*, float *delay*)

Parameters

id : a valid ID of an animation already being blended (0 based)

delay : how long it takes to fade this animation out (seconds)

Returns true if successful, false if an error happened.

5.29.3.7 bool DetachMesh (int *meshID*)

5.29.3.8 void EndRenderingQuery () [inline]

5.29.3.9 bool ExecuteAction (int *id*, float *delayIn*, float *delayOut*, float *weightTgt* = 1.0f, bool *autoLock* = false)

Perform a one time animation. Parameters

id : a valid ID of a animation to perform one-time (0 based)

delayIn : how long it takes to fade in this animation to full strength (seconds)

delayOut : how long it takes to fade out this animation (seconds)

weightTgt : the strength of this animation

autoLock : true prevents the action from being reset and removed on the last key frame

Returns true if successful, false if an error happened.

5.29.3.10 void GetAmbientColor (unsigned char * *colorBuffer*) [inline]

5.29.3.11 float GetAnimationTime ()

Returns the offset time used when playing looping animations.

5.29.3.12 osg::Quat GetBoneAbsoluteRotation (unsigned int *boneID*) const

Parameters

boneID the ID for the CalBone instance.

5.29.3.13 osg::Quat GetBoneAbsoluteRotationForKeyFrame (int *animid*, int *boneid*, unsigned int *keyframeindex*) const

Get the Cal3D rotation values. Parameters

animid the core animation of interest.

boneid the core bone within the animation, identifying the track.

keyframeindex the keyframe array index of interest for the animation track.

Returns the rotation values that cal3d is using, converted into right hand coordinate frame.

5.29.3.14 osg::Vec3 GetBoneAbsoluteTranslation (unsigned int *boneID*) const

Get the current translation for the CalBone. Parameters

boneID the ID for the CalBone of interest.

Returns the translation vector in a right-hand coordinate system.

5.29.3.15 osg::Quat GetBoneRelativeRotation (unsigned int *boneID*) const

Parameters

boneID the ID for the CalBone instance.

5.29.3.16 osg::BoundingBox GetBoundingBox ()

Get a bounding box the encompasses the character in its default pose.

5.29.3.17 const CalModel * GetCalModel () const

Get a const pointer to the internal CalModel. For advanced users only!

Warning! This violates the protective services brought to you by the wrapper. Only modify the CalModel if you know how it will impact the rest of the Delta3D animation system. You have been warned. Returns A const pointer to the internal CalModel this class operates on.

5.29.3.18 CalModel * GetCalModel ()

Get a pointer to the internal CalModel. For advanced users only!

Warning! This violates the protective services brought to you by the wrapper. Only modify the CalModel if you know how it will impact the rest of the Delta3D animation system. You have been warned. Returns A pointer to the internal CalModel this class operates on.

5.29.3.19 int GetCoreAnimationCount () const**5.29.3.20 float GetCoreAnimationDuration (int *animID*) const**

Get the duration of this animation (seconds?).

5.29.3.21 int GetCoreAnimationIDByName (const std::string & *name*) const**5.29.3.22 unsigned int GetCoreAnimationKeyframeCount (int *animID*) const**

Get the total number of keyframes in this animation.

5.29.3.23 unsigned int GetCoreAnimationKeyframeCountForTrack (int *animID*, int *boneID*) const

Get the number of keyframes in the animation for a particular bone.

5.29.3.24 const std::string & GetCoreAnimationName (int *animID*) const

Get the name that equates to the supplied animation ID.

5.29.3.25 unsigned int GetCoreAnimationTrackCount (int *animID*) const

Get the number of tracks this animation uses.

5.29.3.26 void GetCoreBoneChildrenIDs (int *parentCoreBoneID*, std::vector< int > & *toFill*) const

Get the bone IDs of all children for a parent bone. Parameters

parentCoreBoneID the bone ID for the parent bone.

toFill a vector to be filled with the bone IDs for all child bones.

5.29.3.27 int GetCoreBoneID (const std::string & *name*) const

Get the Cal3D CoreBone ID. Returns the ID for the CoreBone.

5.29.3.28 void GetCoreBoneNames (std::vector< std::string > & *toFill*) const

Get all bone IDs within the CoreSkeleton. Parameters

toFill a vector to be filled with the bone IDs for the entire CoreSkeleton.

5.29.3.29 CalCoreMaterial* GetCoreMaterial (int *matID*) [inline]**5.29.3.30 osg::Vec4 GetCoreMaterialAmbient (int *matID*) const**

Get the core material ambient color (rgba 0-255).

5.29.3.31 int GetCoreMaterialCount () const [inline]**5.29.3.32 osg::Vec4 GetCoreMaterialDiffuse (int *matID*) const**

Get the core material diffuse color (rgba 0-255).

5.29.3.33 const std::string& GetCoreMaterialName (int *matID*) const [inline]

Get the name associated with the material using the supplied material ID.

5.29.3.34 float GetCoreMaterialShininess (int *matID*) const

Get the core material shininess.

5.29.3.35 osg::Vec4 GetCoreMaterialSpecular (int *matID*) const

Get the core material specular color (rgba 0-255).

5.29.3.36 int GetCoreMeshCount () const [inline]**5.29.3.37 const std::string & GetCoreMeshName (int *meshID*) const**

Get the name for the mesh using the supplied meshID.

5.29.3.38 osg::Quat GetCoreTrackKeyFrameQuat (unsigned int *animID*, unsigned int *boneID*, unsigned int *keyframeindex*) const

Get the Cal3D rotation values. Parameters

animid the core animation of interest.

boneid the core bone within the animation, identifying the track.

keyframeindex the keyframe array index of interest for the animation track.

Returns the rotation values that cal3d is using, converted into right hand coordinate frame.

5.29.3.39 void GetDiffuseColor (unsigned char * *colorBuffer*) [inline]

5.29.3.40 int GetFaceCount () [inline]

5.29.3.41 int GetFaces (int * *faces*) [inline]

5.29.3.42 int GetMapCount () [inline]

5.29.3.43 void* GetMapUserData (int *mapID*) [inline]

5.29.3.44 int GetMeshCount () [inline]

5.29.3.45 int GetNormals (float * *normals*, int *stride* = 0) [inline]

5.29.3.46 CalHardwareModel * GetOrCreateCalHardwareModel ()

Get the CAL3D CalHardwareModel representation for this CalModel. Warning! This violates the protective services brought to you by the wrapper; use with caution. Returns The hardware model represented by this CalModel

5.29.3.47 int GetParentBoneID (unsigned int *boneID*) const

Get the id of the parent to boneID.

5.29.3.48 void GetRootBoneIDs (std::vector< int > & *toFill*) const

Get all the root bone IDs. Parameters

toFill a container of bone IDs consisting only of root bones.

5.29.3.49 float GetShininess () [inline]

5.29.3.50 void GetSpecularColor (unsigned char * *colorBuffer*) [inline]

5.29.3.51 int GetSubmeshCount (int *submeshID*) [inline]

5.29.3.52 int GetTextureCoords (int *mapID*, float * *coords*, int *stride* = 0) [inline]

5.29.3.53 int GetVertexCount () [inline]

5.29.3.54 int GetVertices (float * *vertBuffer*, int *stride* = 0) [inline]

5.29.3.55 bool HasAnimation (int *animID*) const

5.29.3.56 bool HasBone (int *boneID*) const

5.29.3.57 bool HasTrackForBone (unsigned int *animID*, int *boneID*) const

5.29.3.58 void HideMesh (int *meshID*)

5.29.3.59 bool IsMeshVisible (int *meshID*)

5.29.3.60 bool RemoveAction (int *id*)

Remove an existing one-time animation from the mixer. Parameters

id : a valid ID of a one-time animation already playing (0 based)

Returns true if successful, false if an error happened or animation doesn't exist.

5.29.3.61 bool SelectMeshSubmesh (int *meshID*, int *submeshID*) [inline]

5.29.3.62 void SetAnimationTime (float *time*)

sets the offset time used in synchronized looping animations.

5.29.3.63 void SetCalModel (CalModel * *model*)

5.29.3.64 void SetLODLevel (float *level*) [inline]

5.29.3.65 void SetMaterialSet (int *materialSetID*) [inline]

5.29.3.66 void ShowMesh (int *meshID*)

5.29.3.67 void Update (float *deltaTime*) [inline]

Update the Cal3D system using the CalModel's update.

5.29.3.68 void UpdateAnimation (float *deltaTime*) [inline]

Update just the Cal3D's animation using the mixer.

5.29.3.69 void UpdateMorphTargetMixer (float *deltaTime*) [inline]

Update the CalModel's morph target mixer.

5.29.3.70 void UpdatePhysique () [inline]

Update the CalModel's physique.

5.29.3.71 void UpdateSkeleton () [inline]

Update just Cal3D's skeleton using the mixer.

5.29.3.72 void UpdateSpringSystem (float *deltaTime*) [inline]

Update the CalModel's spring system.

5.29.4 Member Data Documentation

5.29.4.1 const int NULL_BONE = -1 [static]

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

- [cal3dmodelwrapper.h](#)
- [cal3dmodelwrapper.cpp](#)

5.30 CharacterFileHandler Class Reference

Simple Xerces XML handler that will store the data read from a character definition .xml file.

```
#include <inc/dtAnim/characterfilehandler.h>
```

Classes

- struct [AnimatableStruct](#)
- struct [AnimationChannelStruct](#)
- struct [AnimationSequenceStruct](#)
- struct [AnimationStruct](#)
structure to contain all info related to an animation
- struct [MaterialStruct](#)
- struct [MeshStruct](#)
structure to contain all info related to a mesh
- struct [MorphAnimationStruct](#)
structure to contain all info related to an animation

Public Member Functions

- [CharacterFileHandler](#) ()
- [~CharacterFileHandler](#) ()
- virtual void [characters](#) (const XMLCh *const chars, const unsigned int length)
- virtual void [endDocument](#) ()
- virtual void [endElement](#) (const XMLCh *const uri, const XMLCh *const localname, const XMLCh *const qname)
- virtual void [endPrefixMapping](#) (const XMLCh *const)
- virtual void [ignorableWhitespace](#) (const XMLCh *const , const unsigned int)
- virtual void [processingInstruction](#) (const XMLCh *const , const XMLCh *const)
- virtual void [setDocumentLocator](#) (const XERCES_CPP_NAMESPACE_QUALIFIER Locator *const)
- virtual void [skippedEntity](#) (const XMLCh *const)
- virtual void [startDocument](#) ()
- virtual void [startElement](#) (const XMLCh *const uri, const XMLCh *const localname, const XMLCh *const qname, const XERCES_CPP_NAMESPACE_QUALIFIER Attributes &attrs)
- virtual void [startPrefixMapping](#) (const XMLCh *const , const XMLCh *const)

Public Attributes

- std::vector< [AnimationChannelStruct](#) > [mAnimationChannels](#)
The preconfigured playable animations.
- std::vector< [AnimationStruct](#) > [mAnimations](#)
Container of animation structs.
- std::vector< [AnimationSequenceStruct](#) > [mAnimationSequences](#)
The preconfigured playable animations.
- bool [mFoundLODOptions](#)
- bool [mFoundScale](#)
- double [mLODEndDistance](#)
- double [mLODMaxVisibleDistance](#)
- double [mLODStartDistance](#)
- std::vector< [MaterialStruct](#) > [mMaterials](#)
Container of material structs.

- `std::vector< MeshStruct > mMeshes`
Container of mesh structs.
- `std::vector< MorphAnimationStruct > mMorphAnimations`
Container of MorphAnimation structs.
- `std::string mName`
Character Data.
- `std::string mPoseMeshFilename`
- `float mScale`
The scaling factor.
- `std::string mShaderGroup`
Shader information for hardware skinning. these value work with the shader manager.
- `unsigned mShaderMaxBones`
- `std::string mShaderName`
- `std::string mSkeletonFilename`
The one skeleton filename.

Static Public Attributes

- `static const std::string ANIMATION_ELEMENT`
- `static const std::string ANIMATION_NAME_ELEMENT`
- `static const std::string BASE_WEIGHT_ELEMENT`
- `static const std::string CHANNEL_ELEMENT`
- `static const std::string CHARACTER_ELEMENT`
- `static const std::string CHARACTER_XML_LOGGER`
- `static const std::string CHILD_ELEMENT`
- `static const std::string FADE_IN_ELEMENT`
- `static const std::string FADE_OUT_ELEMENT`
- `static const std::string FILENAME_ELEMENT`
- `static const std::string IS_ACTION_ELEMENT`
- `static const std::string IS_LOOPING_ELEMENT`
- `static const std::string LOD_ELEMENT`
- `static const std::string LOD_END_DISTANCE_ELEMENT`
- `static const std::string LOD_START_DISTANCE_ELEMENT`
- `static const std::string MATERIAL_ELEMENT`
- `static const std::string MAX_DURATION_ELEMENT`
- `static const std::string MAX_VISIBLE_DISTANCE_ELEMENT`
- `static const std::string MESH_ELEMENT`
- `static const std::string MORPH_ANIMATION_ELEMENT`
- `static const std::string NAME_ELEMENT`
- `static const std::string POSEMESH_ELEMENT`
- `static const std::string SCALE_ELEMENT`
- `static const std::string SCALE_FACTOR_ELEMENT`
- `static const std::string SEQUENCE_ELEMENT`
- `static const std::string SHADER_GROUP_ELEMENT`
- `static const std::string SHADER_MAX_BONES_ELEMENT`
- `static const std::string SHADER_NAME_ELEMENT`
- `static const std::string SKELETON_ELEMENT`
- `static const std::string SKINNING_SHADER_ELEMENT`
- `static const std::string SPEED_ELEMENT`
- `static const std::string START_DELAY_ELEMENT`

5.30.1 Detailed Description

Simple Xerces XML handler that will store the data read from a character definition .xml file. into a series of containers of strings. Usage:

```
dtUtil::XercesParser parse;
dtAnim::CharacterHandler handler;
parser.Parse(filename, handler);

std::string skeletonFilename = handler.mSkeletonFilename;
...
```

```
<character>
  <skeleton fileName="skel.csf" />
  <animation fileName="anim1.xaf" />
  <animation ...
  <mesh fileName="mesh1.cmf" />
  <mesh ...
  <material fileName="mat1.crf" />
  <material ...
  <animationChannel>
    <name>Run</name>
    <animationName>Run</animationName>
    <startDelay>0.0</startDelay>
    <fadeIn>0.0</fadeIn>
    <fadeOut>0.0</fadeOut>
    <speed>1.0</speed>
    <baseWeight>1.0</baseWeight>
    <maxDuration>0.0</maxDuration>
    <isAction>0</isAction>
    <isLooping>1</isLooping>
  </animationChannel>
</character>
```

5.30.2 Constructor & Destructor Documentation

5.30.2.1 `CharacterFileHandler ()`

5.30.2.2 `~CharacterFileHandler ()`

5.30.3 Member Function Documentation

5.30.3.1 `void characters (const XMLCh *const chars, const unsigned int length) [virtual]`

5.30.3.2 `void endDocument () [virtual]`

5.30.3.3 `void endElement (const XMLCh *const uri, const XMLCh *const localname, const XMLCh *const qname) [virtual]`

5.30.3.4 `virtual void endPrefixMapping (const XMLCh * const) [inline, virtual]`

5.30.3.5 `virtual void ignorableWhitespace (const XMLCh * const, const unsigned int) [inline, virtual]`

5.30.3.6 `virtual void processingInstruction (const XMLCh * const, const XMLCh * const) [inline, virtual]`

5.30.3.7 `virtual void setDocumentLocator (const XERCES_CPP_NAMESPACE_QUALIFIER Locator * const) [inline, virtual]`

5.30.3.8 `virtual void skippedEntity (const XMLCh * const) [inline, virtual]`

5.30.3.9 `void startDocument () [virtual]`

5.30.3.10 `void startElement (const XMLCh *const uri, const XMLCh *const localname, const XMLCh *const qname, const XERCES_CPP_NAMESPACE_QUALIFIER Attributes & attrs) [virtual]`

5.30.3.11 `virtual void startPrefixMapping (const XMLCh * const, const XMLCh * const) [inline, virtual]`

5.30.4 Member Data Documentation

5.30.4.1 `const std::string ANIMATION_ELEMENT [static]`

5.30.4.2 `const std::string ANIMATION_NAME_ELEMENT [static]`

5.30.4.3 `const std::string BASE_WEIGHT_ELEMENT [static]`

5.30.4.4 `const std::string CHANNEL_ELEMENT [static]`

5.30.4.5 `const std::string CHARACTER_ELEMENT [static]`

5.30.4.6 `const std::string CHARACTER_XML_LOGGER [static]`

5.30.4.7 `const std::string CHILD_ELEMENT [static]`

5.30.4.8 `const std::string FADE_IN_ELEMENT [static]`

5.30.4.9 `const std::string FADE_OUT_ELEMENT [static]`

5.30.4.10 `const std::string FILENAME_ELEMENT [static]`

5.30.4.11 `const std::string IS_ACTION_ELEMENT [static]`

5.30.4.12 `const std::string IS_LOOPING_ELEMENT [static]`

5.30.4.13 `const std::string LOD_ELEMENT [static]`

5.30.4.14 `const std::string LOD_END_DISTANCE_ELEMENT [static]`

5.30.4.15 `const std::string LOD_START_DISTANCE_ELEMENT [static]`

5.30.4.16 `std::vector<AnimationChannelStruct> mAnimationChannels`

The preconfigured playable animations.

5.30.4.17 `std::vector<AnimationStruct> mAnimations`

Container of animation structs.

5.30.4.18 std::vector<AnimationSequenceStruct> mAnimationSequences

The preconfigured playable animations.

5.30.4.19 const std::string MATERIAL_ELEMENT [static]**5.30.4.20 const std::string MAX_DURATION_ELEMENT [static]****5.30.4.21 const std::string MAX_VISIBLE_DISTANCE_ELEMENT [static]****5.30.4.22 const std::string MESH_ELEMENT [static]****5.30.4.23 bool mFoundLOOptions****5.30.4.24 bool mFoundScale****5.30.4.25 double mLODEndDistance****5.30.4.26 double mLODMaxVisibleDistance****5.30.4.27 double mLODStartDistance****5.30.4.28 std::vector<MaterialStruct> mMaterials**

Container of material structs.

5.30.4.29 std::vector<MeshStruct> mMeshes

Container of mesh structs.

5.30.4.30 std::vector<MorphAnimationStruct> mMorphAnimations

Container of MorphAnimation structs.

5.30.4.31 std::string mName

Character Data. The name of this animated entity

5.30.4.32 const std::string MORPH_ANIMATION_ELEMENT [static]**5.30.4.33 std::string mPoseMeshFilename****5.30.4.34 float mScale**

The scaling factor.

5.30.4.35 std::string mShaderGroup

Shader information for hardware skinning. these value work with the shader manager.

5.30.4.36 unsigned mShaderMaxBones**5.30.4.37 std::string mShaderName****5.30.4.38 std::string mSkeletonFilename**

The one skeleton filename.

- 5.30.4.39 `const std::string NAME_ELEMENT [static]`
- 5.30.4.40 `const std::string POSEMESH_ELEMENT [static]`
- 5.30.4.41 `const std::string SCALE_ELEMENT [static]`
- 5.30.4.42 `const std::string SCALE_FACTOR_ELEMENT [static]`
- 5.30.4.43 `const std::string SEQUENCE_ELEMENT [static]`
- 5.30.4.44 `const std::string SHADER_GROUP_ELEMENT [static]`
- 5.30.4.45 `const std::string SHADER_MAX_BONES_ELEMENT [static]`
- 5.30.4.46 `const std::string SHADER_NAME_ELEMENT [static]`
- 5.30.4.47 `const std::string SKELETON_ELEMENT [static]`
- 5.30.4.48 `const std::string SKINNING_SHADER_ELEMENT [static]`
- 5.30.4.49 `const std::string SPEED_ELEMENT [static]`
- 5.30.4.50 `const std::string START_DELAY_ELEMENT [static]`

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

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

5.31 CharacterWrapper Class Reference

A Wrapper around an animated character that will perform basic steering.

```
#include <inc/dtAnim/characterwrapper.h>
```

Public Types

- typedef dtCore::Transformable [BaseClass](#)

Public Member Functions

- [CharacterWrapper](#) (const std::string &filename)
- void [ClearAllAnimations](#) (float fadeOutTime)
Just a wrapper interface to the animation helper.
- void [ClearAnimation](#) (const std::string &pAnim, float fadeOutTime)
Just a wrapper interface to the animation helper.
- const [AnimationHelper](#) & [GetAnimationHelper](#) () const
Get the internal [AnimationHelper](#) this instance is using.
- [AnimationHelper](#) & [GetAnimationHelper](#) ()
Get the internal [AnimationHelper](#) this instance is using.
- float [GetHeading](#) () const
Gets the current heading of the character in degrees.
- float [GetHeightAboveGround](#) () const
The height above ground is used for setting the camera height after ground clamping.
- dtCore::Transform [GetLocalOffset](#) () const
The Local Offset is an additional matrix between the character and the loaded art model.
- float [GetRotationSpeed](#) () const
The rotation speed of the character is defined to be a rotation about the Z axis in degrees/sec.
- float [GetSpeed](#) () const
The Speed of the character is defined to be the forward translation in m/s, the character is moved in the update.
- bool [IsAnimationPlaying](#) (const std::string &name) const
Searches the play list for the desired animation, returns true if the animation is playing.
- void [PlayAnimation](#) (const std::string &pAnim)
Just a wrapper interface to the animation helper.
- void [RotateToHeading](#) (float headingInDegrees, float dt)
Rotate to Heading will rotate the character about the Z axis to match the given heading.
- void [RotateToPoint](#) (const osg::Vec3 &point, float dt)
Rotate to Point will rotate the character about the Z axis to match the given direction.
- void [SetGroundClamp](#) (dtCore::Scene *sceneNode, float heightAboveGround)
Initiates ground clamping using a delta transformable, the height above ground is determined to be the height above the ground intersection to set as an offset translation.
- void [SetGroundClamp](#) (dtCore::Transformable *nodeToClampTo, float heightAboveGround)

Initiates ground clamping using a delta transformable, the height above ground is determined to be the height above the ground intersection to set as an offset translation.

- void [SetHeading](#) (float degrees)
Gets the current heading of the character in degrees.
- void [SetHeightAboveGround](#) (float heightAboveGround)
The height above ground is used for setting the camera height after ground clamping.
- void [SetLocalOffset](#) (const dtCore::Transform &localOffset)
The Local Offset is an additional matrix between the character and the loaded art model.
- void [SetRotationSpeed](#) (float degreesPerSecond)
The rotation speed of the character is defined to be a rotation about the Z axis in degrees/sec.
- void [SetSpeed](#) (float metersPerSecond)
The Speed of the character is defined to be the forward translation in m/s, the character is moved in the update.
- void [Update](#) (float dt)
This is a per-frame call which updates the animation helper, and applies rotation and translation from Speed and RotationSpeed.

Protected Member Functions

- [~CharacterWrapper](#) ()

5.31.1 Detailed Description

A Wrapper around an animated character that will perform basic steering.

5.31.2 Member Typedef Documentation

5.31.2.1 typedef dtCore::Transformable BaseClass

5.31.3 Constructor & Destructor Documentation

5.31.3.1 CharacterWrapper (const std::string & filename)

5.31.3.2 ~CharacterWrapper () [protected]

5.31.4 Member Function Documentation

5.31.4.1 void ClearAllAnimations (float fadeOutTime)

Just a wrapper interface to the animation helper. Clears all animations playing. Parameters

the fade out time

5.31.4.2 void ClearAnimation (const std::string & pAnim, float fadeOutTime)

Just a wrapper interface to the animation helper. Parameters

the animation to clear from the play list

the desired time to fade out the animation

5.31.4.3 const AnimationHelper & GetAnimationHelper () const

Get the internal [AnimationHelper](#) this instance is using. Returns The [AnimationHelper](#) used to manage the animations

5.31.4.4 AnimationHelper & GetAnimationHelper ()

Get the internal [AnimationHelper](#) this instance is using. Returns The [AnimationHelper](#) used to manage the animations

5.31.4.5 float GetHeading () const

Gets the current heading of the character in degrees. Returns heading in degrees

5.31.4.6 float GetHeightAboveGround () const

The height above ground is used for setting the camera height after ground clamping. Returns the current translational offset in the Z direction

5.31.4.7 dtCore::Transform GetLocalOffset () const

The Local Offset is an additional matrix between the character and the loaded art model. This offset matrix can be used to rotate the model to face 'forwards' in model space, or apply a scale value that will not be lost by calling `SetTransform()`.

Returns the current local offset (starts as identity)

5.31.4.8 float GetRotationSpeed () const

The rotation speed of the character is defined to be a rotation about the Z axis in degrees/sec. It controls the heading, and is contributed to the per-frame update.

Returns current rotation speed

5.31.4.9 float GetSpeed () const

The Speed of the character is defined to be the forward translation in m/s, the character is moved in the update. Returns the current speed of the character

5.31.4.10 bool IsAnimationPlaying (const std::string & name) const

Searches the play list for the desired animation, returns true if the animation is playing. Parameters

the name of the animation

5.31.4.11 void PlayAnimation (const std::string & pAnim)

Just a wrapper interface to the animation helper. Parameters

the animation to play from the loaded model

5.31.4.12 void RotateToHeading (float headingInDegrees, float dt)

Rotate to Heading will rotate the character about the Z axis to match the given heading. A time step parameter can be used to give a smooth rotation and control the speed at which the character rotates. A time step of 1.0 will give a full rotation to the heading specified. It should be noted that this is an immediate rotation and is independent of the per frame update function.

Parameters

desired heading to rotate towards

current time step

5.31.4.13 void RotateToPoint (const osg::Vec3 & point, float dt)

Rotate to Point will rotate the character about the Z axis to match the given direction. A time step parameter can be used to give a smooth rotation and control the speed at which the character rotates. A time step of 1.0 will give a full rotation to the heading specified. It should be noted that this is an immediate rotation and is independent of the per frame update function.

Parameters

desired point to rotate towards

current time step

5.31.4.14 void SetGroundClamp (dtCore::Scene * sceneNode, float heightAboveGround)

Initiates ground clamping using a delta transformable, the height above ground is determined to be the height above the ground intersection to set as an offset translation. Parameters

a scene to perform ground clamping on

the height above ground

5.31.4.15 void SetGroundClamp (dtCore::Transformable * nodeToClampTo, float heightAboveGround)

Initiates ground clamping using a delta transformable, the height above ground is determined to be the height above the ground intersection to set as an offset translation. Parameters

the node to ground clamp to

the height above ground

5.31.4.16 void SetHeading (float degrees)

Gets the current heading of the character in degrees. Parameters

heading in degrees

5.31.4.17 void SetHeightAboveGround (float heightAboveGround)

The height above ground is used for setting the camera height after ground clamping. It is defined as a translational offset in the Z direction. Note to initiate ground clamping you must call [SetGroundClamp\(\)](#).

Parameters

the desired translational offset in the Z direction

5.31.4.18 void SetLocalOffset (const dtCore::Transform & localOffset)

The Local Offset is an additional matrix between the character and the loaded art model. This offset matrix can be used to rotate the model to face 'forwards' in model space, or apply a scale value that will not be lost by calling [SetTransform\(\)](#).

Parameters

the desired local offset (starts as identity)

5.31.4.19 void SetRotationSpeed (float degreesPerSecond)

The rotation speed of the character is defined to be a rotation about the Z axis in degrees/sec. It controls the heading, and is contributed to the per-frame update.

Returns desired rotation speed

5.31.4.20 void SetSpeed (float metersPerSecond)

The Speed of the character is defined to be the forward translation in m/s, the character is moved in the update. Parameters

the desired speed of the character

5.31.4.21 void Update (float dt)

This is a per-frame call which updates the animation helper, and applies rotation and translation from Speed and RotationSpeed. Parameters

time step

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

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

5.32 CharDrawable Class Reference

A "view" of the cal3d animation state.

```
#include <inc/dtAnim/chardrawable.h>
```

Public Member Functions

- [CharDrawable](#) ([Cal3DModelWrapper](#) *wrapper)
- [~CharDrawable](#) ()
- [Cal3DModelWrapper](#) * [GetCal3DWrapper](#) ()
- [osg::Node](#) * [GetNode](#) () const
Get the Node representing the geometry.
- void [OnMessage](#) ([dtCore::Base::MessageData](#) *data)
- [osg::Node](#) * [RebuildSubmeshes](#) ()
Delete and rebuild all the SubMeshDrawables required, based on the CalRenderer.
- void [SetCal3DWrapper](#) ([Cal3DModelWrapper](#) *wrapper)
change the data this class is viewing.

Protected Attributes

- [dtCore::RefPtr](#)< [Cal3DAnimator](#) > [mAnimator](#)
- int [mLastMeshCount](#)
- [dtCore::RefPtr](#)< [osg::Node](#) > [mNode](#)

5.32.1 Detailed Description

A "view" of the cal3d animation state. Simple class that wraps up a dtAnim::Model so it can be added to the Scene.

Usage:

```
dtCore::RefPtr<dtAnim::CharDrawable> char = new dtAnim::CharDrawable();
dtCore::RefPtr<dtAnim::CoreModel> core = new dtAnim::CoreModel();
core->LoadSkeleton(...);
core->LoadAnimation(...);
char->Create( *core );
myScene->AddDrawable( *char );
```

5.32.2 Constructor & Destructor Documentation

5.32.2.1 CharDrawable (Cal3DModelWrapper * wrapper)

5.32.2.2 ~CharDrawable ()

5.32.3 Member Function Documentation

5.32.3.1 Cal3DModelWrapper * GetCal3DWrapper ()

5.32.3.2 osg::Node* GetNode () const [inline]

Get the Node representing the geometry. Note This Node comes from the [AnimNodeBuilder](#). There is no knowledge of the returned Node's hierarchy in this class. Use with caution. GetOSGNode() is the "normal" method for getting a handle to the geometry. See also GetOSGNode()

5.32.3.3 void OnMessage (dtCore::Base::MessageData * data)

5.32.3.4 osg::Node * RebuildSubmeshes ()

Delete and rebuild all the SubMeshDrawables required, based on the CalRenderer. Returns : the new Node which contains the newly scaled geometry. Note : This will generate brand new geometry. The node previously returned by [GetNode\(\)](#) will be invalid after calling this.

5.32.3.5 void SetCal3DWrapper (Cal3DModelWrapper * *wrapper*)

change the data this class is viewing.

5.32.4 Member Data Documentation**5.32.4.1 dtCore::RefPtr<Cal3DAnimator> mAnimator [protected]****5.32.4.2 int mLastMeshCount [protected]****5.32.4.3 dtCore::RefPtr<osg::Node> mNode [protected]**

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

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

5.33 CloneFunctor Class Reference

Public Member Functions

- [CloneFunctor](#) ([AnimationSequence](#) *pSeq, [Cal3DModelWrapper](#) *pWrapper)
- `template<typename T >`
void [operator\(\)](#) (T &pChild)

5.33.1 Constructor & Destructor Documentation

5.33.1.1 [CloneFunctor](#) ([AnimationSequence](#) * *pSeq*, [Cal3DModelWrapper](#) * *pWrapper*) [`inline`]

5.33.2 Member Function Documentation

5.33.2.1 void [operator\(\)](#) (T & *pChild*) [`inline`]

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

- [animationsequence.cpp](#)

5.34 CreateGeometryDrawCallback Class Reference

Used to delay the building of the animated characters geometry until it is first rendered, at which point a valid OpenGL context should be valid.

Public Member Functions

- [CreateGeometryDrawCallback](#) ([AnimNodeBuilder::CreateFunc](#) &func, [Cal3DModelWrapper](#) *wrapper)
- [~CreateGeometryDrawCallback](#) ()
- virtual void [drawImplementation](#) (osg::RenderInfo &, const osg::Drawable *) const

Public Attributes

- dtCore::RefPtr< osg::Node > [mCreatedNode](#)

5.34.1 Detailed Description

Used to delay the building of the animated characters geometry until it is first rendered, at which point a valid OpenGL context should be valid.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 [CreateGeometryDrawCallback](#) ([AnimNodeBuilder::CreateFunc](#) & *func*, [Cal3DModelWrapper](#) * *wrapper*) [[inline](#)]

5.34.2.2 [~CreateGeometryDrawCallback](#) () [[inline](#)]

5.34.3 Member Function Documentation

5.34.3.1 virtual void [drawImplementation](#) (osg::RenderInfo &, const osg::Drawable *) const [[inline](#), [virtual](#)]

5.34.4 Member Data Documentation

5.34.4.1 dtCore::RefPtr<osg::Node> [mCreatedNode](#)

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

- [animnodebuilder.cpp](#)

5.35 DeletePointer< PtrT > Struct Template Reference

Public Member Functions

- void [operator\(\)](#) (PtrT ptr)

`template<typename PtrT> struct DeletePointer< PtrT >`

5.35.1 Member Function Documentation

5.35.1.1 void operator() (PtrT *ptr*) [inline]

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

- [posemeshdatabase.cpp](#)

5.36 findWithCoreModel Struct Reference

Public Member Functions

- [findWithCoreModel](#) (const CalCoreModel *model)
- bool [operator\(\)](#) (Cal3DModelData *data)

Public Attributes

- const CalCoreModel * [mModel](#)

5.36.1 Constructor & Destructor Documentation

5.36.1.1 [findWithCoreModel](#) (const CalCoreModel * *model*) [[inline](#)]

5.36.2 Member Function Documentation

5.36.2.1 [bool operator\(\)](#) (Cal3DModelData * *data*) [[inline](#)]

5.36.3 Member Data Documentation

5.36.3.1 [const CalCoreModel*](#) [mModel](#)

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

- [cal3ddatabase.cpp](#)

5.37 findWithFilename Struct Reference

Public Member Functions

- [findWithFilename](#) (const std::string &filename)
- bool [operator\(\)](#) ([Cal3DModelData](#) *data)

Public Attributes

- const std::string & [mFilename](#)

5.37.1 Constructor & Destructor Documentation

5.37.1.1 [findWithFilename](#) (const std::string & *filename*) [[inline](#)]

5.37.2 Member Function Documentation

5.37.2.1 [bool operator\(\)](#) ([Cal3DModelData](#) * *data*) [[inline](#)]

5.37.3 Member Data Documentation

5.37.3.1 [const std::string& mFilename](#)

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

- [cal3ddatabase.cpp](#)

5.38 HardwareSubmeshCallback Class Reference

Public Member Functions

- [HardwareSubmeshCallback](#) ([Cal3DModelWrapper](#) &wrapper, [CalHardwareModel](#) &model, [osg::Uniform](#) &boneTrans, unsigned mesh)
- virtual void [update](#) ([osg::NodeVisitor](#) *, [osg::Drawable](#) *drawable)
do customized update code.

5.38.1 Constructor & Destructor Documentation

- 5.38.1.1** [HardwareSubmeshCallback](#) ([Cal3DModelWrapper](#) & *wrapper*, [CalHardwareModel](#) & *model*, [osg::Uniform](#) & *boneTrans*, unsigned *mesh*) [[inline](#)]

5.38.2 Member Function Documentation

- 5.38.2.1** virtual void [update](#) ([osg::NodeVisitor](#) *, [osg::Drawable](#) * *drawable*) [[inline](#), [virtual](#)]

do customized update code.

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

- [hardwaresubmesh.cpp](#)

5.39 HardwareSubmeshComputeBound Class Reference

Public Member Functions

- [HardwareSubmeshComputeBound](#) (const osg::BoundingBox &defaultBox)
- osg::BoundingBox [computeBound](#) (const osg::Drawable &) const

Public Attributes

- const osg::BoundingBox & [mDefaultBox](#)

5.39.1 Constructor & Destructor Documentation

5.39.1.1 [HardwareSubmeshComputeBound](#) (const osg::BoundingBox & *defaultBox*) [inline]

5.39.2 Member Function Documentation

5.39.2.1 [osg::BoundingBox computeBound](#) (const osg::Drawable &) const [inline]

5.39.3 Member Data Documentation

5.39.3.1 [const osg::BoundingBox& mDefaultBox](#)

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

- [hardwaresubmesh.cpp](#)

5.40 HardwareSubmeshDrawable Class Reference

```
#include <inc/dtAnim/hardwaresubmesh.h>
```

Public Member Functions

- [HardwareSubmeshDrawable](#) ([Cal3DModelWrapper](#) *wrapper, [CalHardwareModel](#) *model, const std::string &boneUniformName, unsigned numBones, unsigned mesh, unsigned vertexVBO, unsigned indexVBO)
- virtual [osg::Object](#) * [clone](#) (const [osg::CopyOp](#) &) const
- virtual [osg::Object](#) * [cloneType](#) () const
- virtual void [drawImplementation](#) ([osg::RenderInfo](#) &renderInfo) const
- void [SetBoundingBox](#) (const [osg::BoundingBox](#) &boundingBox)

Protected Member Functions

- [~HardwareSubmeshDrawable](#) ()

5.40.1 Constructor & Destructor Documentation

5.40.1.1 [HardwareSubmeshDrawable](#) ([Cal3DModelWrapper](#) * *wrapper*, [CalHardwareModel](#) * *model*, const std::string & *boneUniformName*, unsigned *numBones*, unsigned *mesh*, unsigned *vertexVBO*, unsigned *indexVBO*)

5.40.1.2 [~HardwareSubmeshDrawable](#) (void) [protected]

5.40.2 Member Function Documentation

5.40.2.1 [osg::Object](#) * [clone](#) (const [osg::CopyOp](#) &) const [virtual]

5.40.2.2 [osg::Object](#) * [cloneType](#) () const [virtual]

5.40.2.3 void [drawImplementation](#) ([osg::RenderInfo](#) & *renderInfo*) const [virtual]

5.40.2.4 void [SetBoundingBox](#) (const [osg::BoundingBox](#) & *boundingBox*)

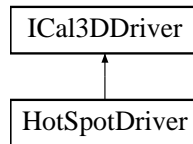
The documentation for this class was generated from the following files:

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

5.41 HotSpotDriver Class Reference

updates the body offset value for HotSpot instances.

#include <inc/dtAnim/hotspotdriver.h> Inheritance diagram for HotSpotDriver::



Public Types

- typedef std::vector< dtCore::RefPtr< dtCore::HotSpotAttachment > > [HotSpotContainer](#)

Public Member Functions

- [HotSpotDriver](#) (const dtAnim::Cal3DModelWrapper *model)
- void [AddHotSpot](#) (dtCore::HotSpotAttachment *spot)
- const [HotSpotContainer](#) & [GetHotSpots](#) () const
- void [RemoveHotSpot](#) (const dtCore::HotSpotAttachment *spot)
- void [SetWrapper](#) (dtAnim::Cal3DModelWrapper *model)
- void [Update](#) (double dt)

Protected Member Functions

- [~HotSpotDriver](#) ()

5.41.1 Detailed Description

updates the body offset value for HotSpot instances.

5.41.2 Member Typedef Documentation

5.41.2.1 typedef std::vector<dtCore::RefPtr<dtCore::HotSpotAttachment>> [HotSpotContainer](#)

5.41.3 Constructor & Destructor Documentation

5.41.3.1 [HotSpotDriver](#) (const dtAnim::Cal3DModelWrapper * *model*)

5.41.3.2 [~HotSpotDriver](#) () [protected]

5.41.4 Member Function Documentation

5.41.4.1 void [AddHotSpot](#) (dtCore::HotSpotAttachment * *spot*)

5.41.4.2 const [HotSpotDriver::HotSpotContainer](#) & [GetHotSpots](#) () const

5.41.4.3 void [RemoveHotSpot](#) (const dtCore::HotSpotAttachment * *spot*)

5.41.4.4 void [SetWrapper](#) (dtAnim::Cal3DModelWrapper * *model*) [virtual]

Implements [ICal3DDriver](#).

5.41.4.5 void [Update](#) (double *dt*) [virtual]

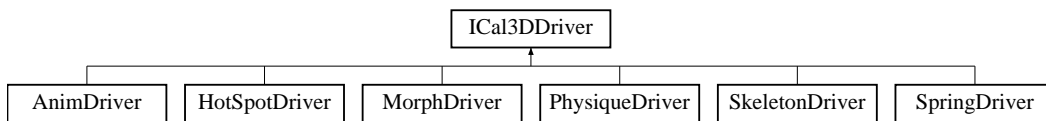
Implements [ICal3DDriver](#).

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

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

5.42 ICal3DDriver Class Reference

#include <inc/dtAnim/ical3ddriver.h> Inheritance diagram for ICal3DDriver::



Public Member Functions

- virtual void [SetWrapper](#) ([Cal3DModelWrapper](#) *)=0
- virtual void [Update](#) (double dt)=0

5.42.1 Member Function Documentation

5.42.1.1 virtual void [SetWrapper](#) ([Cal3DModelWrapper](#) *) [pure virtual]

Implemented in [AnimDriver](#), [HotSpotDriver](#), [MorphDriver](#), [PhysiqueDriver](#), [SkeletonDriver](#), and [SpringDriver](#).

5.42.1.2 virtual void [Update](#) (double *dt*) [pure virtual]

Implemented in [AnimDriver](#), [HotSpotDriver](#), [MorphDriver](#), [PhysiqueDriver](#), [SkeletonDriver](#), and [SpringDriver](#).

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

- [ical3ddriver.h](#)

5.43 IsActor Class Reference

Public Member Functions

- [IsActor](#) (const dtCore::Transformable &actor)
- bool [operator\(\)](#) (const [AttachmentPair](#) &val)

5.43.1 Constructor & Destructor Documentation

5.43.1.1 [IsActor](#) (const dtCore::Transformable & *actor*) [[inline](#)]

5.43.2 Member Function Documentation

5.43.2.1 bool [operator\(\)](#) (const [AttachmentPair](#) & *va*) [[inline](#)]

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

- [attachmentcontroller.cpp](#)

5.44 LODOptions Class Reference

A simple data class that stores the configuration options for level of detail.

```
#include <inc/dtAnim/cal3dmodeldata.h>
```

Public Member Functions

- [LODOptions](#) ()
- double [GetEndDistance](#) () const
- double [GetMaxVisibleDistance](#) () const
- double [GetStartDistance](#) () const
- void [SetEndDistance](#) (double newDistance)
- void [SetMaxVisibleDistance](#) (double newDistance)
- void [SetStartDistance](#) (double newDistance)

5.44.1 Detailed Description

A simple data class that stores the configuration options for level of detail.

5.44.2 Constructor & Destructor Documentation

5.44.2.1 LODOptions ()

5.44.3 Member Function Documentation

5.44.3.1 double GetEndDistance () const [inline]

Returns the distance at which the level of detail should be the minimum.

5.44.3.2 double GetMaxVisibleDistance () const [inline]

Returns the maximum distance that the model will be drawn at all.

5.44.3.3 double GetStartDistance () const [inline]

Returns the distance at which to start decreasing the level of detail.

5.44.3.4 void SetEndDistance (double newDistance)

5.44.3.5 void SetMaxVisibleDistance (double newDistance)

5.44.3.6 void SetStartDistance (double newDistance)

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

- [cal3dmodeldata.h](#)
- [cal3dmodeldata.cpp](#)

5.45 MaterialStruct Struct Reference

```
#include <inc/dtAnim/characterfilehandler.h>
```

Public Attributes

- `std::string mFileName`
The filename of the Cal3D material.
- `std::string mName`
The user friendly name of this material.

5.45.1 Member Data Documentation

5.45.1.1 `std::string mFileName`

The filename of the Cal3D material.

5.45.1.2 `std::string mName`

The user friendly name of this material.

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

- [characterfilehandler.h](#)

5.46 MeshStruct Struct Reference

structure to contain all info related to a mesh

```
#include <inc/dtAnim/characterfilehandler.h>
```

Public Attributes

- `std::string mFileName`
The filename of the Cal3D mesh.
- `std::string mName`
The user friendly name of this mesh.

5.46.1 Detailed Description

structure to contain all info related to a mesh

5.46.2 Member Data Documentation

5.46.2.1 `std::string mFileName`

The filename of the Cal3D mesh.

5.46.2.2 `std::string mName`

The user friendly name of this mesh.

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

- [characterfilehandler.h](#)

5.47 MorphAnimationStruct Struct Reference

structure to contain all info related to an animation

```
#include <inc/dtAnim/characterfilehandler.h>
```

Public Attributes

- `std::string mFileName`
The filename of the cal3D morph animation.
- `std::string mName`
The user friendly name of this morph animation.

5.47.1 Detailed Description

structure to contain all info related to an animation

5.47.2 Member Data Documentation

5.47.2.1 `std::string mFileName`

The filename of the cal3D morph animation.

5.47.2.2 `std::string mName`

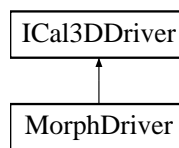
The user friendly name of this morph animation.

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

- [characterfilehandler.h](#)

5.48 MorphDriver Class Reference

#include <inc/dtAnim/morphdriver.h> Inheritance diagram for MorphDriver::



Public Member Functions

- [MorphDriver](#) ([Cal3DModelWrapper](#) *pWrapper)
- void [SetWrapper](#) ([Cal3DModelWrapper](#) *)
- void [Update](#) (double dt)

Protected Member Functions

- virtual [~MorphDriver](#) ()

5.48.1 Constructor & Destructor Documentation

5.48.1.1 [MorphDriver](#) ([Cal3DModelWrapper](#) * *pWrapper*)

5.48.1.2 [~MorphDriver](#) () [[protected](#), [virtual](#)]

5.48.2 Member Function Documentation

5.48.2.1 void [SetWrapper](#) ([Cal3DModelWrapper](#) * *pWrapper*) [[virtual](#)]

Implements [ICal3DDriver](#).

5.48.2.2 void [Update](#) (double *dt*) [[virtual](#)]

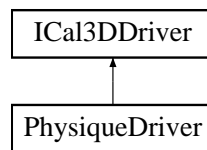
Implements [ICal3DDriver](#).

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

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

5.49 PhysiqueDriver Class Reference

#include <inc/dtAnim/physiquedriver.h> Inheritance diagram for PhysiqueDriver::



Public Member Functions

- [PhysiqueDriver](#) ([Cal3DModelWrapper](#) *pWrapper)
- void [SetWrapper](#) ([Cal3DModelWrapper](#) *)
- void [Update](#) (double dt)

Protected Member Functions

- virtual [~PhysiqueDriver](#) ()

5.49.1 Constructor & Destructor Documentation

5.49.1.1 [PhysiqueDriver](#) ([Cal3DModelWrapper](#) * *pWrapper*)

5.49.1.2 [~PhysiqueDriver](#) () [protected, virtual]

5.49.2 Member Function Documentation

5.49.2.1 void [SetWrapper](#) ([Cal3DModelWrapper](#) * *pWrapper*) [virtual]

Implements [ICal3DDriver](#).

5.49.2.2 void [Update](#) (double *dt*) [virtual]

Implements [ICal3DDriver](#).

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

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

5.50 PoseBuilderFuncutor< ContainerT > Struct Template Reference

Public Member Functions

- [PoseBuilderFuncutor](#) (dtAnim::Cal3DModelWrapper *model, ContainerT *container)
- void [operator\(\)](#) (const [PoseMeshData](#) &meshData)

```
template<typename ContainerT> struct PoseBuilderFuncutor< ContainerT >
```

5.50.1 Constructor & Destructor Documentation

5.50.1.1 [PoseBuilderFuncutor](#) (dtAnim::Cal3DModelWrapper * *model*, ContainerT * *container*) [inline]

5.50.2 Member Function Documentation

5.50.2.1 void [operator\(\)](#) (const [PoseMeshData](#) & *meshData*) [inline]

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

- [posemeshdatabase.cpp](#)

5.51 PoseMesh Class Reference

```
#include <inc/dtAnim/posemesh.h>
```

Classes

- struct [TargetTriangle](#)
- struct [Triangle](#)
- struct [TriangleEdge](#)
- struct [Vertex](#)

Public Types

- typedef dtUtil::BarycentricSpace< osg::Vec3 > [Barycentric2D](#)
- typedef std::vector< [Barycentric2D](#) * > [Barycentric2DVector](#)
- typedef std::map< [MeshIndexPair](#), osg::ref_ptr< osg::Geometry > > [EdgeLineMap](#)
- typedef std::pair< unsigned short, unsigned short > [MeshIndexPair](#)
- typedef std::vector< std::string > [StringVector](#)
- typedef std::vector< [TriangleEdge](#) > [TriangleEdgeVector](#)
- typedef std::vector< [Triangle](#) > [TriangleVector](#)
- typedef std::vector< [Vertex](#) * > [VertexVector](#)

Public Member Functions

- [PoseMesh](#) (dtAnim::Cal3DModelWrapper *model, const [PoseMeshData](#) &meshData)
- [~PoseMesh](#) ()
- int [FindPoseTriangleID](#) (float azimuth, float elevation) const
FindCelestialTriangleID Looks up a celestial triangle from a mesh using azimuth and elevation.
- const [Barycentric2DVector](#) & [GetBarySpaces](#) () const
- const osg::Vec3 & [GetEffectorForwardAxis](#) () const
- int [GetEffectorID](#) () const
- const std::string & [GetEffectorName](#) () const
- void [GetIndexPairsForTriangle](#) (int triangleID, [MeshIndexPair](#) &pair0, [MeshIndexPair](#) &pair1, [MeshIndexPair](#) &pair2) const
GetIndexPairsForTriangle Look up the indices for a triangle and create pairs corresponding to its edges.
- const std::string & [GetName](#) () const
- const osg::Vec3 & [GetRootForwardAxis](#) () const
- const [TriangleEdgeVector](#) [GetSilhouette](#) () const
- void [GetTargetTriangleData](#) (const float azimuth, const float elevation, [TargetTriangle](#) &outTriangle) const
GetTargetTriangleData Finds the triangle in the mesh for the given azimuth elevation if it exists, otherwise it returns the closest triangle and its coordinates.
- const [TriangleVector](#) & [GetTriangles](#) () const
- const [VertexVector](#) & [GetVertices](#) () const

5.51.1 Member Typedef Documentation

5.51.1.1 `typedef dtUtil::BarycentricSpace<osg::Vec3> Barycentric2D`

5.51.1.2 `typedef std::vector<Barycentric2D*> Barycentric2DVector`

5.51.1.3 `typedef std::map<MeshIndexPair, osg::ref_ptr<osg::Geometry> > EdgeLineMap`

5.51.1.4 `typedef std::pair<unsigned short, unsigned short> MeshIndexPair`

5.51.1.5 `typedef std::vector<std::string> StringVector`

5.51.1.6 `typedef std::vector<TriangleEdge> TriangleEdgeVector`

5.51.1.7 `typedef std::vector<Triangle> TriangleVector`

5.51.1.8 `typedef std::vector<Vertex*> VertexVector`

5.51.2 Constructor & Destructor Documentation

5.51.2.1 `PoseMesh (dtAnim::Cal3DModelWrapper * model, const PoseMeshData & meshData)`

Todo

now also build the barycentric array

5.51.2.2 `~PoseMesh ()`

5.51.3 Member Function Documentation

5.51.3.1 `int FindPoseTriangleID (float azimuth, float elevation) const`

FindCelestialTriangleID Looks up a celestial triangle from a mesh using azimuth and elevation. Algorithm in detail at <http://www.blackpawn.com/texts/pointinpoly/default.html>.

Parameters

azimuth the horizontal angle of interest

elevation the vertical angle of interest

5.51.3.2 `const Barycentric2DVector& GetBarySpaces () const [inline]`

5.51.3.3 `const osg::Vec3& GetEffectorForwardAxis () const [inline]`

5.51.3.4 `int GetEffectorID () const [inline]`

5.51.3.5 `const std::string& GetEffectorName () const [inline]`

5.51.3.6 `void GetIndexPairsForTriangle (int triangleID, MeshIndexPair & pair0, MeshIndexPair & pair1, MeshIndexPair & pair2) const`

GetIndexPairsForTriangle Look up the indices for a triangle and create pairs corresponding to its edges. Parameters

triangleID the triangle to extract pairs from

pair0 the first pair(edge)

pair1 the second pair(edge)

pair2 the thris pair(edge)

5.51.3.7 `const std::string& GetName () const [inline]`

5.51.3.8 `const osg::Vec3& GetRootForwardAxis () const [inline]`

5.51.3.9 `const TriangleEdgeVector GetSilhouette () const [inline]`

5.51.3.10 `void GetTargetTriangleData (const float azimuth, const float elevation, TargetTriangle & outTriangle) const`

GetTargetTriangleData Finds the triangle in the mesh for the given azimuth elevation if it exists, otherwise it returns the closest triangle and its coordinates. Parameters

azimuth the horizontal angle between our forward and our target

elevation the vertical angle between our forward and our target

Returns outTriangle struct containing the nearest triangle and it's location

5.51.3.11 `const TriangleVector& GetTriangles () const [inline]`

5.51.3.12 `const VertexVector& GetVertices () const [inline]`

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

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

5.52 PoseMeshData Struct Reference

```
#include <inc/dtAnim/posemeshxml.h>
```

Public Attributes

- [StringVector mAnimations](#)
a list of name triples for animation triangles
- `osg::Vec3` [mEffectorForward](#)
the direction that is forward in the end effector's bone's space
- `std::string` [mEffectorName](#)
an identifier for the end effector bone
- `std::string` [mName](#)
an identifier for this pose mesh instance
- `osg::Vec3` [mRootForward](#)
the direction that end effectors are relative to
- `std::string` [mRootName](#)
an identifier for the bone that will give the global forward

5.52.1 Member Data Documentation

5.52.1.1 StringVector mAnimations

a list of name triples for animation triangles

5.52.1.2 osg::Vec3 mEffectorForward

the direction that is forward in the end effector's bone's space

5.52.1.3 std::string mEffectorName

an identifier for the end effector bone

5.52.1.4 std::string mName

an identifier for this pose mesh instance

5.52.1.5 osg::Vec3 mRootForward

the direction that end effectors are relative to

5.52.1.6 std::string mRootName

an identifier for the bone that will give the global forward

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

- [posemeshxml.h](#)

5.53 PoseMeshDatabase Class Reference

manager of the HotSpotData resources

```
#include <inc/dtAnim/posemeshdatabase.h>
```

Public Types

- typedef std::vector< [PoseMesh](#) * > [PoseMeshList](#)

Public Member Functions

- [PoseMeshDatabase](#) (dtAnim::Cal3DModelWrapper *model)
- [~PoseMeshDatabase](#) ()
- [PoseMeshList](#) & [GetMeshes](#) ()
- [PoseMesh](#) * [GetPoseMeshByName](#) (const std::string &name)
- bool [LoadFromFile](#) (const std::string &file)

5.53.1 Detailed Description

manager of the HotSpotData resources

5.53.2 Member Typedef Documentation

5.53.2.1 typedef std::vector<[PoseMesh](#)*> [PoseMeshList](#)

5.53.3 Constructor & Destructor Documentation

5.53.3.1 [PoseMeshDatabase](#) (dtAnim::Cal3DModelWrapper * *model*)

5.53.3.2 [~PoseMeshDatabase](#) ()

5.53.4 Member Function Documentation

5.53.4.1 [PoseMeshList](#)& [GetMeshes](#) () [inline]

5.53.4.2 [PoseMesh](#) * [GetPoseMeshByName](#) (const std::string & *name*)

5.53.4.3 bool [LoadFromFile](#) (const std::string & *file*)

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

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

5.54 PoseMeshFileHandler Class Reference

```
#include <inc/dtAnim/posemeshxml.h>
```

Public Types

- typedef std::stack< [PoseNode](#) > [NodeStack](#)
- typedef std::vector< [PoseMeshData](#) > [PoseMeshDataVector](#)
- enum [PoseNode](#) { [NODE_UNKNOWN](#), [NODE_POSEMESH](#), [NODE_TRIANGLE](#), [NODE_ANIMATION](#) }

Public Member Functions

- [PoseMeshFileHandler](#) ()
- [~PoseMeshFileHandler](#) ()
- void [characters](#) (const XMLCh *const chars, const unsigned int length)
- void [endDocument](#) ()
- void [endElement](#) (const XMLCh *const uri, const XMLCh *const localname, const XMLCh *const qname)
- void [endPrefixMapping](#) (const XMLCh *const prefix)
- const [PoseMeshDataVector](#) & [GetData](#) ()
- void [ignorableWhitespace](#) (const XMLCh *const chars, const unsigned int length)
- void [processingInstruction](#) (const XMLCh *const target, const XMLCh *const data)
- void [setDocumentLocator](#) (const XERCES_CPP_NAMESPACE_QUALIFIER Locator *const locator)
- void [skippedEntity](#) (const XMLCh *const name)
- void [startDocument](#) ()
- void [startElement](#) (const XMLCh *const uri, const XMLCh *const localname, const XMLCh *const qname, const XERCES_CPP_NAMESPACE_QUALIFIER Attributes &attrs)
- void [startPrefixMapping](#) (const XMLCh *const prefix, const XMLCh *const uri)

Static Public Attributes

- static const char [ANIMATION_NODE](#) [] = { "Animation\0" }
- static const char [DEFAULT_VALUE](#) [] = { "default\0" }
- static const char [EFFECTOR_ATTRIBUTE](#) [] = { "effector\0" }
- static const char [EFFECTOR_FORWARD_ATTRIBUTE](#) [] = { "effectorForwardAxis\0" }
- static const char [NAME_ATTRIBUTE](#) [] = { "name\0" }
- static const char [POSE_NODE](#) [] = { "PoseMesh\0" }
- static const char [ROOT_ATTRIBUTE](#) [] = { "root\0" }
- static const char [ROOT_FORWARD_ATTRIBUTE](#) [] = { "rootForwardAxis\0" }
- static const char [TRIANGLE_NODE](#) [] = { "Triangle\0" }

5.54.1 Member Typedef Documentation

5.54.1.1 typedef std::stack<[PoseNode](#)> [NodeStack](#)

5.54.1.2 typedef std::vector<[PoseMeshData](#)> [PoseMeshDataVector](#)

5.54.2 Member Enumeration Documentation

5.54.2.1 enum [PoseNode](#)

Enumerator:

NODE_UNKNOWN

NODE_POSEMESH

NODE_TRIANGLE

NODE_ANIMATION

5.54.3 Constructor & Destructor Documentation

5.54.3.1 PoseMeshFileHandler ()

5.54.3.2 ~PoseMeshFileHandler ()

5.54.4 Member Function Documentation

5.54.4.1 void characters (const XMLCh *const *chars*, const unsigned int *length*)

5.54.4.2 void endDocument () [inline]

5.54.4.3 void endElement (const XMLCh *const *uri*, const XMLCh *const *localname*, const XMLCh *const *qname*)

5.54.4.4 void endPrefixMapping (const XMLCh *const *prefix*) [inline]

5.54.4.5 const PoseMeshDataVector& GetData () [inline]

5.54.4.6 void ignorableWhitespace (const XMLCh *const *chars*, const unsigned int *length*) [inline]

5.54.4.7 void processingInstruction (const XMLCh *const *target*, const XMLCh *const *data*) [inline]

5.54.4.8 void setDocumentLocator (const XERCES_CPP_NAMESPACE_QUALIFIER Locator *const *locator*) [inline]

5.54.4.9 void skippedEntity (const XMLCh *const *name*) [inline]

5.54.4.10 void startDocument () [inline]

5.54.4.11 void startElement (const XMLCh *const *uri*, const XMLCh *const *localname*, const XMLCh *const *qname*, const XERCES_CPP_NAMESPACE_QUALIFIER Attributes & *attrs*)

5.54.4.12 void startPrefixMapping (const XMLCh *const *prefix*, const XMLCh *const *uri*) [inline]

5.54.5 Member Data Documentation

5.54.5.1 const char ANIMATION_NODE = { "Animation\0" } [static]

5.54.5.2 const char DEFAULT_VALUE = { "default\0" } [static]

5.54.5.3 const char EFFECTOR_ATTRIBUTE = { "effector\0" } [static]

5.54.5.4 const char EFFECTOR_FORWARD_ATTRIBUTE = { "effectorForwardAxis\0" } [static]

5.54.5.5 const char NAME_ATTRIBUTE = { "name\0" } [static]

5.54.5.6 const char POSE_NODE = { "PoseMesh\0" } [static]

5.54.5.7 const char ROOT_ATTRIBUTE = { "root\0" } [static]

5.54.5.8 const char ROOT_FORWARD_ATTRIBUTE = { "rootForwardAxis\0" } [static]

5.54.5.9 const char TRIANGLE_NODE = { "Triangle\0" } [static]

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

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

5.55 PoseMeshLoader Class Reference

```
#include <inc/dtAnim/posemeshloader.h>
```

Public Types

- typedef std::vector< [PoseMeshData](#) > [MeshDataContainer](#)

Public Member Functions

- [PoseMeshLoader](#) ()
- [~PoseMeshLoader](#) ()
- bool [Load](#) (const std::string &file, [MeshDataContainer](#) &toFill)

5.55.1 Member Typedef Documentation

5.55.1.1 typedef std::vector<[PoseMeshData](#)> [MeshDataContainer](#)

5.55.2 Constructor & Destructor Documentation

5.55.2.1 [PoseMeshLoader](#) ()

5.55.2.2 [~PoseMeshLoader](#) ()

5.55.3 Member Function Documentation

5.55.3.1 bool [Load](#) (const std::string & *file*, [MeshDataContainer](#) & *toFill*)

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

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

5.56 PoseMeshUtility Class Reference

Convenience functionality for manipulating pose meshes.

```
#include <inc/dtAnim/posemeshutility.h>
```

Classes

- struct [BaseReferenceBlend](#)

Public Types

- typedef std::pair< int, float > [BaseReferencePose](#)

Public Member Functions

- [PoseMeshUtility](#) ()
- [~PoseMeshUtility](#) ()
- void [BlendPoses](#) (const [PoseMesh](#) *poseMesh, [dtAnim::Cal3DModelWrapper](#) *model, const [PoseMesh::TargetTriangle](#) &targetTriangle, float blendDelay)
BlendPoses Applies a 3 animation blend to a model based on TargetTriangle IK data.
- void [ClearPoses](#) (const [PoseMesh](#) *poseMesh, [dtAnim::Cal3DModelWrapper](#) *model, float delay)
ClearPoses Removes all animations associated with a pose mesh from the mixer.
- bool [GetBaseReferenceBlend](#) (float overallAlpha, [BaseReferenceBlend](#) &outFinalBlend)
- void [SetBaseReferencePoses](#) (std::vector< [BaseReferencePose](#) > *poseList, const [dtAnim::Cal3DModelWrapper](#) *model)

5.56.1 Detailed Description

Convenience functionality for manipulating pose meshes.

5.56.2 Member Typedef Documentation

5.56.2.1 typedef std::pair<int, float> BaseReferencePose

5.56.3 Constructor & Destructor Documentation

5.56.3.1 PoseMeshUtility ()

5.56.3.2 ~PoseMeshUtility ()

5.56.4 Member Function Documentation

5.56.4.1 void BlendPoses (const PoseMesh * poseMesh, dtAnim::Cal3DModelWrapper * model, const PoseMesh::TargetTriangle & targetTriangle, float blendDelay)

BlendPoses Applies a 3 animation blend to a model based on TargetTriangle IK data. Parameters

model the model where the animations will be applied

targetTriangle the 3 triangle points and a point located within

5.56.4.2 void ClearPoses (const PoseMesh * poseMesh, dtAnim::Cal3DModelWrapper * model, float delay)

ClearPoses Removes all animations associated with a pose mesh from the mixer. Parameters

poseMesh the mesh containing all animation to 'clear'

model the model where animations will be 'cleared' from

delay the amount of time before the start of the 'clear'

5.56.4.3 **bool** GetBaseReferenceBlend (float *overallAlpha*, BaseReferenceBlend & *outFinalBlend*)

5.56.4.4 **void** SetBaseReferencePoses (std::vector< BaseReferencePose > * *poseList*, const dtAnim::CaI3DModelWrapper * *model*)

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

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

5.57 PredicatePoseMeshName Struct Reference

Public Member Functions

- [PredicatePoseMeshName](#) (const [PredicatePoseMeshName](#) &same)
- [PredicatePoseMeshName](#) (const std::string &name)
- bool [operator\(\)](#) (const [PoseMesh](#) *mesh) const

Public Attributes

- std::string [mName](#)

5.57.1 Constructor & Destructor Documentation

5.57.1.1 [PredicatePoseMeshName](#) (const std::string & *name*) [inline]

5.57.1.2 [PredicatePoseMeshName](#) (const [PredicatePoseMeshName](#) & *same*) [inline]

5.57.2 Member Function Documentation

5.57.2.1 bool [operator\(\)](#) (const [PoseMesh](#) * *mesh*) const [inline]

5.57.3 Member Data Documentation

5.57.3.1 std::string [mName](#)

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

- [posemeshdatabase.cpp](#)

5.58 PropertyNames Struct Reference

string constants for this actor

```
#include <inc/dtAnim/cal3dgameactor.h>
```

Static Public Attributes

- static const dtUtil::RefString [ANIMATION_BLEND_DELAY](#)
- static const dtUtil::RefString [ANIMATION_BLEND_GROUP](#)
- static const dtUtil::RefString [ANIMATION_BLEND_ID](#)
- static const dtUtil::RefString [ANIMATION_BLEND_WEIGHT](#)
- static const dtUtil::RefString [ANIMATION_GROUP](#)
- static const dtUtil::RefString [ANIMATION_GROUP_LABEL](#)
- static const dtUtil::RefString [RENDER_MODE](#)
- static const dtUtil::RefString [RENDER_MODE_LABEL](#)

5.58.1 Detailed Description

string constants for this actor

5.58.2 Member Data Documentation

5.58.2.1 const dtUtil::RefString [ANIMATION_BLEND_DELAY](#) [static]

5.58.2.2 const dtUtil::RefString [ANIMATION_BLEND_GROUP](#) [static]

5.58.2.3 const dtUtil::RefString [ANIMATION_BLEND_ID](#) [static]

5.58.2.4 const dtUtil::RefString [ANIMATION_BLEND_WEIGHT](#) [static]

5.58.2.5 const dtUtil::RefString [ANIMATION_GROUP](#) [static]

5.58.2.6 const dtUtil::RefString [ANIMATION_GROUP_LABEL](#) [static]

5.58.2.7 const dtUtil::RefString [RENDER_MODE](#) [static]

5.58.2.8 const dtUtil::RefString [RENDER_MODE_LABEL](#) [static]

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

- [cal3dgameactor.h](#)
- [cal3dgameactor.cpp](#)

5.59 RecalcFuncor Class Reference

Public Member Functions

- [RecalcFuncor](#) (float pStart)
- float [GetEnd](#) () const
- template<typename T >
void [operator](#)() (T &pChild)

5.59.1 Constructor & Destructor Documentation

5.59.1.1 [RecalcFuncor \(float pStart\)](#) [inline]

5.59.2 Member Function Documentation

5.59.2.1 [float GetEnd \(\) const](#) [inline]

5.59.2.2 [void operator\(\) \(T & pChild\)](#) [inline]

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

- [animationsequence.cpp](#)

5.60 RenderPrimitive Struct Reference

a functor to be used in a loop algorithm

```
#include <inc/dtAnim/skeletaldrawable.h>
```

Public Member Functions

- [RenderPrimitive](#) (const [RenderPrimitive](#) &same)
- [RenderPrimitive](#) (const [Cal3DModelWrapper](#) *model)
- void [operator\(\)](#) (const dtAnim::SkeletalDrawable::IPrimitiveRenderObject *ptr) const

Public Attributes

- dtCore::RefPtr< const [Cal3DModelWrapper](#) > [mModelWrapper](#)

5.60.1 Detailed Description

a functor to be used in a loop algorithm

5.60.2 Constructor & Destructor Documentation

5.60.2.1 [RenderPrimitive](#) (const [Cal3DModelWrapper](#) * *model*) [[inline](#)]

5.60.2.2 [RenderPrimitive](#) (const [RenderPrimitive](#) & *same*) [[inline](#)]

5.60.3 Member Function Documentation

5.60.3.1 void [operator\(\)](#) (const dtAnim::SkeletalDrawable::IPrimitiveRenderObject * *ptr*) const [[inline](#)]

5.60.4 Member Data Documentation

5.60.4.1 dtCore::RefPtr<const [Cal3DModelWrapper](#)> [mModelWrapper](#)

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

- [skeletaldrawable.h](#)

5.61 SequenceMixer Class Reference

The SequenceMixer's job is to manage animations and animation sequences.

```
#include <inc/dtAnim/sequencemixer.h>
```

Public Types

- typedef std::map< std::string, dtCore::RefPtr< const Animatable > > AnimationTable
- typedef AnimationTable::allocator_type::value_type TableKey

Public Member Functions

- [SequenceMixer](#) ()
- void [ClearActiveAnimations](#) (float time)

This function will clear all currently playing animations from the mixer over the fade out time specified.
- void [ClearAnimation](#) (const std::string &pAnim, float time)

This function will clear an animation from the play list by name and fade it out over the time specified.
- void [ClearRegisteredAnimations](#) ()

This function will clear all animations registered with the mixer.
- void [ForceRecalculate](#) ()

This function forces all non-active animations to recalculate their start and end times.
- const Animatable * [GetActiveAnimation](#) (const std::string &pAnim) const

This function returns a pointer to the active animation specified by name.
- Animatable * [GetActiveAnimation](#) (const std::string &pAnim)

This function returns a pointer to the active animation specified by name.
- const Animatable * [GetRegisteredAnimation](#) (const std::string &pAnim) const

This function returns a pointer to the registered animation specified by name.
- void [GetRegisteredAnimations](#) (std::vector< const Animatable * > &toFill) const

Fills the given vector with all of the animatables registered with the mixer.
- bool [IsAnimationPlaying](#) (const std::string &pAnim) const
- void [PlayAnimation](#) (Animatable *pAnim)

PlayAnimation adds the given animation to active play list.
- void [RegisterAnimation](#) (const Animatable *pAnimation)

This function registers and animation within the system.
- void [RemoveRegisteredAnimation](#) (const std::string &pAnim)

This function will remove a registered animation by name from the mixer.
- void [Update](#) (float dt)

The Update() must be call every frame, this is done automatically using an AnimationHelper.

Protected Member Functions

- virtual [~SequenceMixer](#) ()

5.61.1 Detailed Description

The SequenceMixer's job is to manage animations and animation sequences. Animations are registered with the [SequenceMixer](#) using an [AnimationHelper](#) on LoadModel(). Once an animation or sequence is registered the user can call [PlayAnimation\(\)](#) by name. Alternatively the user can create an [Animatable](#) using an [AnimationWrapper](#) and call PlayAnimation using that. To clear the animation use the name specified by Animatable* GetName(), or the name used to play the animation.

5.61.2 Member Typedef Documentation

5.61.2.1 typedef std::map<std::string, dtCore::RefPtr<const Animatable> > AnimationTable

5.61.2.2 typedef AnimationTable::allocator_type::value_type TableKey

5.61.3 Constructor & Destructor Documentation

5.61.3.1 SequenceMixer ()

5.61.3.2 ~SequenceMixer () [protected, virtual]

5.61.4 Member Function Documentation

5.61.4.1 void ClearActiveAnimations (float *time*)

This function will clear all currently playing animations from the mixer over the fade out time specified. Parameters

the time to fade out over

5.61.4.2 void ClearAnimation (const std::string & *pAnim*, float *time*)

This function will clear an animation from the play list by name and fade it out over the time specified. Parameters

the name of the animation to fade out

the time to fade out over

5.61.4.3 void ClearRegisteredAnimations ()

This function will clear all animations registered with the mixer.

5.61.4.4 void ForceRecalculate ()

This function forces all non-active animations to recalculate their start and end times.

5.61.4.5 const Animatable * GetActiveAnimation (const std::string & *pAnim*) const

This function returns a pointer to the active animation specified by name. Parameters

the name of the active animation

Returns the active animation within the system, 0 if this animation does not exist

5.61.4.6 Animatable * GetActiveAnimation (const std::string & *pAnim*)

This function returns a pointer to the active animation specified by name. Parameters

the name of the active animation

Returns the active animation within the system, 0 if this animation does not exist

5.61.4.7 const Animatable * GetRegisteredAnimation (const std::string & *pAnim*) const

This function returns a pointer to the registered animation specified by name. Parameters

the name of the active animation

Returns the active animation within the system, 0 if this animation does not exist

5.61.4.8 void GetRegisteredAnimations (std::vector< const Animatable * > & toFill) const

Fills the given vector with all of the animatables registered with the mixer. Parameters

toFill the vector of pointers to fill.

5.61.4.9 bool IsAnimationPlaying (const std::string & pAnim) const

Returns whether or not the specified animation is playing

5.61.4.10 void PlayAnimation (Animatable * pAnim)

PlayAnimation adds the given animation to active play list. Parameters

The name of the animation registered with the mixer

5.61.4.11 void RegisterAnimation (const Animatable * pAnimation)

This function registers an animation within the system. Registered animation can be configured and played by name. This animation is registered with the name specified by GetName().

Parameters

the animation to register, the name this animation is registered with comes from calling GetName() on the animatable.

5.61.4.12 void RemoveRegisteredAnimation (const std::string & pAnim)

This function will remove a registered animation by name from the mixer.

5.61.4.13 void Update (float dt)

The [Update\(\)](#) must be called every frame, this is done automatically using an [AnimationHelper](#). Parameters

delta time

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

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

5.62 SkeletalDrawable Class Reference

Renders only the skeleton.

```
#include <inc/dtAnim/skeletaldrawable.h>
```

Classes

- struct **CPrimitiveRenderObject**
- struct **IPrimitiveRenderObject**

Public Member Functions

- [SkeletalDrawable](#) (const [Cal3DModelWrapper](#) *model)
- [osg::Object](#) * [clone](#) (const [osg::CopyOp](#) ©op) const
- [osg::Object](#) * [cloneType](#) () const
- void [drawImplementation](#) ([osg::RenderInfo](#) &[RenderInfo](#)) const

Protected Member Functions

- [~SkeletalDrawable](#) ()

5.62.1 Detailed Description

Renders only the skeleton.

5.62.2 Constructor & Destructor Documentation

5.62.2.1 [SkeletalDrawable](#) (const [Cal3DModelWrapper](#) * *model*)

5.62.2.2 [~SkeletalDrawable](#) () [protected]

5.62.3 Member Function Documentation

5.62.3.1 [osg::Object](#) * [clone](#) (const [osg::CopyOp](#) & *copyop*) const

5.62.3.2 [osg::Object](#) * [cloneType](#) () const

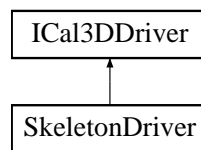
5.62.3.3 void [drawImplementation](#) ([osg::RenderInfo](#) & *RenderInfo*) const

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

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

5.63 SkeletonDriver Class Reference

#include <inc/dtAnim/skeletondriver.h> Inheritance diagram for SkeletonDriver::



Public Member Functions

- [SkeletonDriver](#) ([Cal3DModelWrapper](#) *pWrapper)
- void [SetWrapper](#) ([Cal3DModelWrapper](#) *)
- void [Update](#) (double dt)

Protected Member Functions

- virtual [~SkeletonDriver](#) ()

5.63.1 Constructor & Destructor Documentation

5.63.1.1 [SkeletonDriver](#) ([Cal3DModelWrapper](#) * *pWrapper*)

5.63.1.2 [~SkeletonDriver](#) () [[protected](#), [virtual](#)]

5.63.2 Member Function Documentation

5.63.2.1 void [SetWrapper](#) ([Cal3DModelWrapper](#) * *pWrapper*) [[virtual](#)]

Implements [ICal3DDriver](#).

5.63.2.2 void [Update](#) (double *dt*) [[virtual](#)]

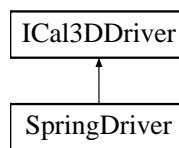
Implements [ICal3DDriver](#).

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

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

5.64 SpringDriver Class Reference

#include <inc/dtAnim/springdriver.h> Inheritance diagram for SpringDriver::



Public Member Functions

- [SpringDriver](#) ([Cal3DModelWrapper](#) *pWrapper)
- void [SetWrapper](#) ([Cal3DModelWrapper](#) *)
- void [Update](#) (double dt)

Protected Member Functions

- virtual [~SpringDriver](#) ()

5.64.1 Constructor & Destructor Documentation

5.64.1.1 [SpringDriver](#) ([Cal3DModelWrapper](#) * *pWrapper*)

5.64.1.2 [~SpringDriver](#) () [protected, virtual]

5.64.2 Member Function Documentation

5.64.2.1 void [SetWrapper](#) ([Cal3DModelWrapper](#) * *pWrapper*) [virtual]

Implements [ICal3DDriver](#).

5.64.2.2 void [Update](#) (double *dt*) [virtual]

Implements [ICal3DDriver](#).

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

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

5.65 SubmeshComputeBound Class Reference

Public Member Functions

- [SubmeshComputeBound](#) (const osg::BoundingBox &boundingBox)
- osg::BoundingBox [computeBound](#) (const osg::Drawable &) const

Public Attributes

- const osg::BoundingBox & [mBoundingBox](#)

5.65.1 Constructor & Destructor Documentation

5.65.1.1 [SubmeshComputeBound](#) (const osg::BoundingBox & *boundingBox*) [inline]

5.65.2 Member Function Documentation

5.65.2.1 [osg::BoundingBox computeBound](#) (const osg::Drawable &) const [inline]

5.65.3 Member Data Documentation

5.65.3.1 [const osg::BoundingBox& mBoundingBox](#)

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

- [submesh.cpp](#)

5.66 SubmeshCullCallback Class Reference

```
#include <inc/dtAnim/submesh.h>
```

Public Member Functions

- [SubmeshCullCallback](#) ([CaI3DModelWrapper](#) &wrapper, int meshID)
- virtual bool [cull](#) (osg::NodeVisitor *nv, osg::Drawable *drawable, osg::RenderInfo *renderInfo) const

5.66.1 Constructor & Destructor Documentation

5.66.1.1 [SubmeshCullCallback](#) ([CaI3DModelWrapper](#) & *wrapper*, int *meshID*) [inline]

5.66.2 Member Function Documentation

5.66.2.1 bool [cull](#) (osg::NodeVisitor * *nv*, osg::Drawable * *drawable*, osg::RenderInfo * *renderInfo*) const [virtual]

copy user data to the state.

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

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

5.67 SubmeshDirtyCallback Class Reference

```
#include <inc/dtAnim/submesh.h>
```

Public Member Functions

- virtual void [update](#) (osg::NodeVisitor *, osg::Drawable *d)

5.67.1 Member Function Documentation

5.67.1.1 void update (osg::NodeVisitor *, osg::Drawable * d) [virtual]

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

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

5.68 SubmeshDrawable Class Reference

Adapter that converts cal3d submeshes into osg::Drawables.

```
#include <inc/dtAnim/submesh.h>
```

Public Member Functions

- [SubmeshDrawable](#) ([Cal3DModelWrapper](#) *wrapper, unsigned mesh, unsigned submesh)
Creates a submesh for one model given the mesh and submesh of this mesh.
- virtual void [accept](#) (osg::PrimitiveFunctor &pf) const
Accept a PrimitiveFunctor and call its methods to tell it about the interal primitives that this Drawable has.
- virtual osg::Object * [clone](#) (const osg::CopyOp &) const
- virtual osg::Object * [cloneType](#) () const
- virtual void [drawImplementation](#) (osg::RenderInfo &renderInfo) const
Draws the geometry.
- float [GetCurrentLOD](#) () const
- const [LODOptions](#) & [GetLODOptions](#) () const
- [LODOptions](#) & [GetLODOptions](#) ()
- unsigned int [GetMeshID](#) () const
- const [Cal3DModelWrapper](#) & [GetModelWrapper](#) () const
- [Cal3DModelWrapper](#) & [GetModelWrapper](#) ()
- unsigned int [GetSubmeshID](#) () const
- void [SetBoundingBox](#) (const osg::BoundingBox &boundingBox)
- void [SetCurrentLOD](#) (float lod)
- virtual bool [supports](#) (osg::PrimitiveFunctor &) const
Accept PrimitiveVisitor, in this case a TriangleVisitor.

Static Public Attributes

- static const unsigned [LOD_COUNT](#) = 4

Protected Member Functions

- [~SubmeshDrawable](#) ()
- void [InitVertexBuffers](#) (osg::State &state) const

5.68.1 Detailed Description

Adapter that converts cal3d submeshes into osg::Drawables. h The easy way would be to draw all the character (CalModel) in a single Drawable, but this approach lacks from state sorting. Each submesh of each mesh of a model can have different state attributes. With the current approach, if you have 1000 soldiers with two different textures each one, there will be only two state changes per frame, and not 2000.

Users of the osgCal library doesn't need to know about this class, it is internal.

5.68.2 Constructor & Destructor Documentation

5.68.2.1 SubmeshDrawable (Cal3DModelWrapper * wrapper, unsigned mesh, unsigned submesh)

Creates a submesh for one model given the mesh and submesh of this mesh.

5.68.2.2 ~SubmeshDrawable (void) [protected]

5.68.3 Member Function Documentation

5.68.3.1 void accept (osg::PrimitiveFunctor & pf) const [virtual]

Accept a PrimitiveFunctor and call its methods to tell it about the interal primitives that this Drawable has. this processes the lowest LOD at the moment, because that's what's loaded at the front of the VBO.

5.68.3.2 `osg::Object * clone (const osg::CopyOp &) const [virtual]`

5.68.3.3 `osg::Object * cloneType () const [virtual]`

5.68.3.4 `void drawImplementation (osg::RenderInfo & renderInfo) const [virtual]`

Draws the geometry.

5.68.3.5 `float GetCurrentLOD () const [inline]`

5.68.3.6 `const LODOptions& GetLODOptions () const [inline]`

5.68.3.7 `LODOptions& GetLODOptions () [inline]`

5.68.3.8 `unsigned int GetMeshID () const [inline]`

5.68.3.9 `const Cal3DModelWrapper& GetModelWrapper () const [inline]`

5.68.3.10 `Cal3DModelWrapper& GetModelWrapper () [inline]`

5.68.3.11 `unsigned int GetSubmeshID () const [inline]`

5.68.3.12 `void InitVertexBuffers (osg::State & state) const [protected]`

Fill the index and vertex VBOs once for each level of detail

offset into the vbo to fill the correct lod.

5.68.3.13 `void SetBoundingBox (const osg::BoundingBox & boundingBox) [inline]`

5.68.3.14 `void SetCurrentLOD (float lod) [inline]`

5.68.3.15 `virtual bool supports (osg::PrimitiveFunctor &) const [inline, virtual]`

Accept PrimitiveVisitor, in this case a TriangleVisitor. Return true, SubMesh does support accept(PrimitiveFunctor&).

5.68.4 Member Data Documentation

5.68.4.1 `const unsigned LOD_COUNT = 4 [static]`

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

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

5.69 SubmeshUserData Class Reference

```
#include <inc/dtAnim/submesh.h>
```

Public Types

- typedef osg::Object [BaseClass](#)

Public Member Functions

- virtual const char * [className](#) () const
return the name of the object's class type.
- virtual osg::Object * [clone](#) (const osg::CopyOp &) const
Clone an object, with Object return type.*
- virtual osg::Object * [cloneType](#) () const
Clone the type of an object, with Object return type.*
- virtual bool [isSameKindAs](#) (const osg::Object *) const
- virtual const char * [libraryName](#) () const
return the name of the object's library.

Public Attributes

- float [mLOD](#)

5.69.1 Member Typedef Documentation

5.69.1.1 typedef osg::Object BaseClass

5.69.2 Member Function Documentation

5.69.2.1 const char * className () const [virtual]

return the name of the object's class type. Must be defined by derived classes.

5.69.2.2 osg::Object * clone (const osg::CopyOp & op) const [virtual]

Clone an object, with Object* return type. Must be defined by derived classes.

5.69.2.3 osg::Object * cloneType () const [virtual]

Clone the type of an object, with Object* return type. Must be defined by derived classes.

5.69.2.4 virtual bool isSameKindAs (const osg::Object *) const [virtual]

5.69.2.5 const char * libraryName () const [virtual]

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

5.69.3 Member Data Documentation

5.69.3.1 float mLOD

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

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

5.70 TargetTriangle Struct Reference

```
#include <inc/dtAnim/posemesh.h>
```

Public Attributes

- float [mAzimuth](#)
- float [mElevation](#)
- bool [mIsInside](#)
- int [mTriangleID](#)

5.70.1 Member Data Documentation

5.70.1.1 float [mAzimuth](#)

5.70.1.2 float [mElevation](#)

5.70.1.3 bool [mIsInside](#)

5.70.1.4 int [mTriangleID](#)

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

- [posemesh.h](#)

5.71 Triangle Struct Reference

```
#include <inc/dtAnim/posemesh.h>
```

Public Member Functions

- [Triangle](#) (const [Vertex](#) *a, const [Vertex](#) *b, const [Vertex](#) *c, unsigned short aIndex, unsigned short bIndex, unsigned short cIndex)

Public Attributes

- unsigned short [mIndices](#) [3]
- const [Vertex](#) * [mVertices](#) [3]

5.71.1 Constructor & Destructor Documentation

5.71.1.1 [Triangle](#) (const [Vertex](#) * a, const [Vertex](#) * b, const [Vertex](#) * c, unsigned short *aIndex*, unsigned short *bIndex*, unsigned short *cIndex*)

5.71.2 Member Data Documentation

5.71.2.1 unsigned short [mIndices](#)[3]

5.71.2.2 const [Vertex](#)* [mVertices](#)[3]

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

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

5.72 TriangleEdge Struct Reference

```
#include <inc/dtAnim/posemesh.h>
```

Public Member Functions

- [TriangleEdge](#) (const [MeshIndexPair](#) edge, const int triangleIndex)
- [TriangleEdge](#) ()

Public Attributes

- [MeshIndexPair](#) mEdge
- int mTriangleID

5.72.1 Constructor & Destructor Documentation

5.72.1.1 [TriangleEdge](#) () [inline]

5.72.1.2 [TriangleEdge](#) (const [MeshIndexPair](#) edge, const int *triangleIndex*) [inline]

5.72.2 Member Data Documentation

5.72.2.1 [MeshIndexPair](#) mEdge

5.72.2.2 int mTriangleID

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

- [posemesh.h](#)

5.73 UpdateCallback Class Reference

Used to grab the created geometry from the [CreateGeometryDrawCallback](#) and add it as a child to the supplied Group.

Public Member Functions

- [UpdateCallback](#) ([CreateGeometryDrawCallback](#) *callback, osg::Group &group)
- [~UpdateCallback](#) ()
- virtual void [operator\(\)](#) (osg::Node *node, osg::NodeVisitor *nv)

5.73.1 Detailed Description

Used to grab the created geometry from the [CreateGeometryDrawCallback](#) and add it as a child to the supplied Group.

5.73.2 Constructor & Destructor Documentation

5.73.2.1 [UpdateCallback](#) ([CreateGeometryDrawCallback](#) * *callback*, osg::Group & *group*) [[inline](#)]

5.73.2.2 [~UpdateCallback](#) () [[inline](#)]

5.73.3 Member Function Documentation

5.73.3.1 virtual void [operator\(\)](#) (osg::Node * *node*, osg::NodeVisitor * *nv*) [[inline](#), [virtual](#)]

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

- [animnodebuilder.cpp](#)

5.74 Vertex Struct Reference

```
#include <inc/dtAnim/posemesh.h>
```

Public Member Functions

- [Vertex](#) (const osg::Vec3 &data, unsigned int animID)

Public Attributes

- unsigned int [mAnimID](#)
- osg::Vec3 [mData](#)
- osg::Vec3 [mDebugData](#)
- float [mDebugPrecision](#)
- osg::Quat [mDebugRotation](#)

5.74.1 Constructor & Destructor Documentation

5.74.1.1 [Vertex](#) (const osg::Vec3 & *data*, unsigned int *animID*)

5.74.2 Member Data Documentation

5.74.2.1 unsigned int [mAnimID](#)

5.74.2.2 osg::Vec3 [mData](#)

5.74.2.3 osg::Vec3 [mDebugData](#)

5.74.2.4 float [mDebugPrecision](#)

5.74.2.5 osg::Quat [mDebugRotation](#)

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

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

File Documentation

6.1 animactorregistry.cpp File Reference

```
#include <dtAnim/animactorregistry.h>
#include <dtAnim/animationgameactor.h>
#include <dtAnim/cal3dgameactor.h>
```

Functions

- DT_ANIM_EXPORT dtDAL::ActorPluginRegistry * [CreatePluginRegistry](#) ()
- DT_ANIM_EXPORT void [DestroyPluginRegistry](#) (dtDAL::ActorPluginRegistry *registry)

6.1.1 Function Documentation

6.1.1.1 DT_ANIM_EXPORT dtDAL::ActorPluginRegistry* [CreatePluginRegistry](#) ()

6.1.1.2 DT_ANIM_EXPORT void [DestroyPluginRegistry](#) (dtDAL::ActorPluginRegistry * *registry*)

6.2 animactorregistry.h File Reference

```
#include <dtDAL/actorpluginregistry.h>
```

```
#include <dtAnim/export.h>
```

Classes

- class [AnimActorRegistry](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.3 animatable.cpp File Reference

```
#include <dtAnim/animatable.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.4 animatable.h File Reference

```
#include <dtAnim/export.h>
#include <osg/Referenced>
#include <dtCore/refptr.h>
#include <dtUtil/functor.h>
#include <string>
```

Classes

- class [Animatable](#)

This class is used to specify the base class of an object which has semantics for animating.

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

Typedefs

- typedef dtUtil::Functor< void, TYPELIST_1(const [dtAnim::Animatable](#) &) > [AnimationCallback](#)

6.5 animationchannel.cpp File Reference

```
#include <dtAnim/animationchannel.h>
#include <dtAnim/animationwrapper.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <osg/Math>
#include <dtUtil/log.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.6 animationchannel.h File Reference

```
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
#include <dtAnim/animatable.h>
```

Classes

- class [AnimationChannel](#)
AnimationChannel derives from [Animatable](#) and holds an [AnimationWrapper](#), and contains semantics for playing an animation using the [Cal3DModelWrapper](#) API.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.7 animationcomponent.cpp File Reference

```
#include <dtAnim/animationcomponent.h>
#include <dtGame/basemessages.h>
#include <dtGame/defaultgroundclamper.h>
#include <dtUtil/log.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.8 animationcomponent.h File Reference

```
#include <string>
#include <map>
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
#include <dtGame/gmcomponent.h>
#include <dtAnim/animationhelper.h>
```

Classes

- class [AnimationComponent](#)

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.
- namespace [dtGame](#)

6.9 animationgameactor.cpp File Reference

```
#include <dtAnim/animationgameactor.h>
#include <dtGame/gamemanager.h>
#include <dtGame/actorupdatemessage.h>
#include <dtDAL/enginepropertytypes.h>
#include <dtDAL/actorproxyicon.h>
#include <dtGame/basemessages.h>
#include <dtGame/invokable.h>
#include <dtDAL/functor.h>
#include <dtAnim/animnodebuilder.h>
#include <osg/MatrixTransform>
#include <osg/Geode>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.10 animationgameactor.h File Reference

```
#include <dtCore/refptr.h>
#include <dtGame/gameactor.h>
#include <dtAnim/export.h>
#include <dtDAL/namedparameter.h>
#include <dtAnim/animationhelper.h>
#include <string>
```

Classes

- class [AnimationGameActor](#)
This class is the game actor for an animated model.
- class [AnimationGameActorProxy](#)
This class is the proxy for an animated model game object.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.11 animationhelper.cpp File Reference

```
#include <dtAnim/animationhelper.h>
#include <dtAnim/animnodebuilder.h>
#include <dtAnim/cal3ddatabase.h>
#include <dtAnim/cal3dmodeldata.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtAnim/ical3ddriver.h>
#include <dtAnim/animationsequence.h>
#include <dtAnim/animationchannel.h>
#include <dtAnim/animationwrapper.h>
#include <dtDAL/actorproperty.h>
#include <dtDAL/enginepropertytypes.h>
#include <dtDAL/actorproxy.h>
#include <dtCore/hotspotattachment.h>
#include <dtUtil/log.h>
#include <osg/Node>
#include <osg/Texture2D>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.12 animationhelper.h File Reference

```
#include <dtAnim/export.h>
#include <dtAnim/cal3danimator.h>
#include <dtAnim/sequencemixer.h>
#include <dtAnim/attachmentcontroller.h>
#include <dtCore/refptr.h>
#include <osg/Referenced>
#include <string>
#include <vector>
```

Classes

- class [AnimationHelper](#)

The [AnimationHelper](#) class is a utility class to simplify adding animation to an articulated entity, it provides support for loading, rendering and animating.

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

- namespace [dtDAL](#)

6.13 animationsequence.cpp File Reference

```
#include <dtAnim/animationsequence.h>
#include <dtAnim/animatable.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtCore/refptr.h>
#include <dtUtil/log.h>
#include <dtUtil/mathdefines.h>
#include <algorithm>
```

Classes

- class [AnimSeqForceFade](#)
- struct [AnimSequenceUpdater](#)
- class [CloneFunctor](#)
- class [RecalcFunctor](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.14 animationsequence.h File Reference

```
#include <dtAnim/export.h>
#include <dtAnim/animatable.h>
#include <dtCore/refptr.h>
#include <dtCore/observerptr.h>
#include <list>
#include <string>
```

Classes

- class [AnimationController](#)
AnimationController is responsible for updating the sequences child Animatables.
- class [AnimationSequence](#)
AnimationSequence derives from [Animatable](#) and contains a child list of animations to play.

Namespaces

- namespace [dtAnim](#)
The *dtAnim* Library contains functionality used to support skeletal mesh animations.

6.15 animationwrapper.cpp File Reference

```
#include <dtAnim/animationwrapper.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.16 animationwrapper.h File Reference

```
#include <dtAnim/export.h>
#include <osg/Referenced>
#include <string>
```

Classes

- class [AnimationWrapper](#)
The [AnimationWrapper](#) is meant to be a wrapper around a Cal3D animation.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.17 animdriver.cpp File Reference

```
#include <dtAnim/animdriver.h>
#include <cal3d/model.h>
#include <cal3d/mixer.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.18 animdriver.h File Reference

```
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
#include <dtAnim/ical3ddriver.h>
#include <dtAnim/cal3dmodelwrapper.h>
```

Classes

- class [AnimDriver](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.19 animnodebuilder.cpp File Reference

```
#include <dtAnim/animnodebuilder.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtAnim/submesh.h>
#include <dtAnim/hardwaresubmesh.h>
#include <dtAnim/cal3ddatabase.h>
#include <dtAnim/cal3dmodeldata.h>
#include <dtCore/globals.h>
#include <dtCore/shaderprogram.h>
#include <dtCore/shadermanager.h>
#include <dtCore/shadergroup.h>
#include <dtUtil/log.h>
#include <dtUtil/macros.h>
#include <osg/Geode>
#include <osg/State>
#include <osg/BoundingSphere>
#include <osg/BoundingBox>
#include <osg/Texture2D>
#include <osg/GLExtensions>
#include <osg/ShapeDrawable>
#include <cal3d/hardwaremodel.h>
#include <cal3d/vector.h>
```

Classes

- class [CreateGeometryDrawCallback](#)
Used to delay the building of the animated characters geometry until it is first rendered, at which point a valid OpenGL context should be valid.
- class [UpdateCallback](#)
Used to grab the created geometry from the [CreateGeometryDrawCallback](#) and add it as a child to the supplied Group.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.20 animnodebuilder.h File Reference

```
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
#include <dtUtil/functor.h>
#include <osg/Referenced>
#include <osg/Node>
#include <cal3d/global.h>
```

Classes

- class [AnimNodeBuilder](#)
Class used to generate the renderable geometry for an animated character.
- class **Array**< **T** >
- class [Cal3DBoundingSphereCalculator](#)

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.
- namespace [dtCore](#)

6.21 attachmentcontroller.cpp File Reference

```
#include <dtAnim/attachmentcontroller.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtCore/transformable.h>
#include <dtCore/transform.h>
#include <dtCore/hotspotattachment.h>
#include <dtUtil/log.h>
#include <osg/Quat>
#include <osg/Vec3>
#include <osg/Matrix>
#include <algorithm>
```

Classes

- class [IsActor](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.22 attachmentcontroller.h File Reference

```
#include <osg/Referenced>
#include <dtCore/refptr.h>
#include <dtAnim/export.h>
#include <dtUtil/hotspotdefinition.h>
#include <vector>
```

Classes

- class [AttachmentController](#)
Stores a list of attachments for a cal model and can update their positions based each frame based on the position of the bones.
- class [AttachmentMover](#)
This is a helper functor for moving attachments.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.
- namespace [dtCore](#)

Typedefs

- typedef std::pair< dtCore::RefPtr< dtCore::Transformable >, dtUtil::HotSpotDefinition > [AttachmentPair](#)

6.23 cal3d animator.cpp File Reference

```
#include <dtAnim/cal3d animator.h>
#include <dtAnim/animdriver.h>
#include <dtAnim/skeletondriver.h>
#include <dtAnim/morphdriver.h>
#include <dtAnim/physiquedriver.h>
#include <dtAnim/springdriver.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.24 cal3d animator.h File Reference

```
#include <dtAnim/export.h>
#include <dtAnim/ical3ddriver.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <osg/Referenced>
#include <dtCore/refptr.h>
```

Classes

- class [Cal3DAnimator](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.25 cal3ddatabase.cpp File Reference

```
#include <dtAnim/cal3ddatabase.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtAnim/animnodebuilder.h>
#include <dtUtil/log.h>
#include <osgDB/FileNameUtils>
#include <osg/Texture2D>
#include <cal3d/model.h>
```

Classes

- struct [findWithCoreModel](#)
- struct [findWithFilename](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

Functions

- `template<class T , class Array >`
`const Array::value_type::element_type * FindWithFuncor (Array a, T functor)`

6.26 cal3ddatabase.h File Reference

```
#include <dtAnim/export.h>
#include <dtAnim/cal3dloader.h>
#include <dtAnim/animnodebuilder.h>
#include <dtAnim/cal3dmodeldata.h>
#include <dtCore/refptr.h>
#include <osg/Referenced>
#include <string>
#include <vector>
```

Classes

- class [Cal3DDatabase](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.27 cal3dgameactor.cpp File Reference

```
#include <dtAnim/cal3dgameactor.h>
#include <dtDAL/groupactorproperty.h>
#include <dtGame/gamemanager.h>
#include <dtGame/actorupdatemessage.h>
#include <dtDAL/enginepropertytypes.h>
#include <dtDAL/actorproxyicon.h>
#include <dtAnim/submesh.h>
#include <dtAnim/skeletaldrawable.h>
#include <dtAnim/cal3ddatabase.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtAnim/cal3d animator.h>
#include <dtCore/scene.h>
#include <dtGame/basemessages.h>
#include <dtCore/system.h>
#include <dtGame/invokable.h>
#include <dtDAL/functor.h>
#include <osg/MatrixTransform>
#include <osg/Geode>
#include <osg/Material>
#include <osg/PolygonMode>
#include <dtUtil/bits.h>
#include <cstdint>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.28 cal3dgameactor.h File Reference

```
#include <dtGame/gameactor.h>
#include <dtAnim/export.h>
#include <dtAnim/cal3d animator.h>
#include <dtAnim/cal3ddatabase.h>
#include <dtDAL/namedparameter.h>
#include <string>
```

Classes

- class [Cal3DGameActor](#)
This class is the game actor for an animated model.
- class [Cal3DGameActorProxy](#)
This class is the proxy for an animated model game object.
- struct [PropertyNames](#)
string constants for this actor

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.
- namespace [dtCore](#)

6.29 cal3dloader.cpp File Reference

```
#include <dtAnim/cal3dloader.h>
#include <osgDB/ReadFile>
#include <osg/Texture2D>
#include <cal3d/model.h>
#include <cal3d/coremodel.h>
#include <cal3d/coreanimation.h>
#include <dtAnim/characterfilehandler.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtAnim/animationwrapper.h>
#include <dtAnim/animationchannel.h>
#include <dtAnim/animationsequence.h>
#include <dtAnim/cal3dmodeldata.h>
#include <dtUtil/xercesparser.h>
#include <dtUtil/log.h>
#include <dtCore/globals.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.30 cal3dloader.h File Reference

```
#include <dtAnim/export.h>
#include <string>
#include <map>
#include <osg/ref_ptr>
#include <osg/Texture2D>
#include <dtCore/refptr.h>
#include <osg/Referenced>
```

Classes

- class [Cal3DLoader](#)
Loads a animation definition file and returns a valid CalModel.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.31 cal3dmodeldata.cpp File Reference

```
#include <dtAnim/cal3dmodeldata.h>
#include <dtAnim/animatable.h>
#include <dtAnim/animationwrapper.h>
#include <cal3d/coremodel.h>
#include <algorithm>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.32 cal3dmodeldata.h File Reference

```
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
#include <osg/Referenced>
#include <vector>
```

Classes

- class [Cal3DModelData](#)
- class [LODOptions](#)

A simple data class that stores the configuration options for level of detail.

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.33 cal3dmodelwrapper.cpp File Reference

```
#include <dtAnim/cal3dmodelwrapper.h>
#include <cal3d/cal3d.h>
#include <cal3d/coretrack.h>
#include <cal3d/corekeyframe.h>
#include <cassert>
#include <algorithm>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.34 cal3dmodelwrapper.h File Reference

```
#include <dtAnim/export.h>
#include <cal3d/model.h>
#include <cal3d/hardwaremodel.h>
#include <cal3d/coremodel.h>
#include <cal3d/renderer.h>
#include <cal3d/mixer.h>
#include <cal3d/morphtargetmixer.h>
#include <cal3d/physique.h>
#include <cal3d/springsystem.h>
#include <osg/Quat>
#include <osg/Referenced>
#include <osg/Vec4>
#include <osg/Vec3>
#include <osg/BoundingBox>
#include <vector>
```

Classes

- class [Cal3DModelWrapper](#)
Wraps the Cal3D CalModel class.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.35 characterfilehandler.cpp File Reference

```
#include <dtAnim/characterfilehandler.h>
#include <dtUtil/xercesutils.h>
#include <dtUtil/stringutils.h>
#include <dtUtil/log.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

Variables

- [XERCES_CPP_NAMESPACE_USE](#)

6.35.1 Variable Documentation

6.35.1.1 XERCES_CPP_NAMESPACE_USE

6.36 characterfilehandler.h File Reference

```
#include <dtAnim/export.h>
#include <dtUtil/macros.h>
#include <dtCore/refptr.h>
#include <xercesc/sax2/ContentHandler.hpp>
#include <vector>
#include <string>
#include <stack>
```

Classes

- struct [AnimatableStruct](#)
- struct [AnimationChannelStruct](#)
- struct [AnimationSequenceStruct](#)
- struct [AnimationStruct](#)
structure to contain all info related to an animation
- class [CharacterFileHandler](#)
Simple Xerces XML handler that will store the data read from a character definition .xml file.
- struct [MaterialStruct](#)
- struct [MeshStruct](#)
structure to contain all info related to a mesh
- struct [MorphAnimationStruct](#)
structure to contain all info related to an animation

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.
- namespace [dtUtil](#)

6.37 characterwrapper.cpp File Reference

```
#include <dtAnim/characterwrapper.h>
#include <osg/Math>
#include <dtCore/isector.h>
#include <dtCore/transform.h>
#include <dtUtil/matrixutil.h>
#include <dtAnim/sequencemixer.h>
#include <osg/MatrixTransform>
#include <dtAnim/cal3d animator.h>
#include <dtAnim/attachmentcontroller.h>
#include <dtAnim/animationsequence.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtAnim/ical3ddriver.h>
#include <dtAnim/animationhelper.h>
#include <dtUtil/hotspotdefinition.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.38 characterwrapper.h File Reference

```
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
#include <dtCore/observerptr.h>
#include <dtCore/transformable.h>
#include <string>
```

Classes

- class [CharacterWrapper](#)
A Wrapper around an animated character that will perform basic steering.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.
- namespace [dtCore](#)

6.39 chardrawable.cpp File Reference

```
#include <osg/MatrixTransform>
#include <osg/Node>
#include <osg/Timer>
#include <osg/Texture2D>
#include <dtAnim/submesh.h>
#include <dtAnim/chardrawable.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtAnim/cal3d animator.h>
#include <dtAnim/cal3dmodeldata.h>
#include <dtAnim/cal3ddatabase.h>
#include <dtAnim/animnodebuilder.h>
#include <dtCore/system.h>
#include <dtUtil/log.h>
#include <cassert>
#include <cal3d/corematerial.h>
#include <dtAnim/ical3ddriver.h>
```

6.40 chardrawable.h File Reference

```
#include <dtCore/transformable.h>
```

```
#include <dtAnim/export.h>
```

Classes

- class [CharDrawable](#)
A "view" of the cal3d animation state.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.41 dtanim.h File Reference

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.42 export.h File Reference

Defines

- #define [DT_ANIM_EXPORT](#)

6.42.1 Define Documentation

6.42.1.1 #define DT_ANIM_EXPORT

6.43 hardwaresubmesh.cpp File Reference

```
#include <dtAnim/hardwaresubmesh.h>
#include <dtAnim/submesh.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <osg/Material>
#include <osg/Texture2D>
#include <osg/PolygonMode>
#include <osg/Uniform>
#include <osg/PrimitiveSet>
#include <osg/Vec3>
#include <osg/BoundingBox>
#include <dtUtil/matrixutil.h>
#include <dtUtil/log.h>
#include <cal3d/hardwaremodel.h>
#include <osg/CullFace>
#include <osg/BlendFunc>
```

Classes

- class [HardwareSubmeshCallback](#)
- class [HardwareSubmeshComputeBound](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

Defines

- #define [BUFFER_OFFSET\(x\)](#) ((GLvoid*) (0 + ((x) * sizeof(float))))

6.43.1 Define Documentation

6.43.1.1 #define [BUFFER_OFFSET\(x\)](#) ((GLvoid*) (0 + ((x) * sizeof(float))))

6.44 hardwaresubmesh.h File Reference

```
#include <osg/Drawable>
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
```

Classes

- class [HardwareSubmeshDrawable](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.45 hotspotdriver.cpp File Reference

```
#include <dtAnim/hotspotdriver.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtAnim/attachmentcontroller.h>
#include <dtCore/hotspotattachment.h>
#include <dtCore/transformable.h>
#include <algorithm>
#include <cstdint>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.46 hotspotdriver.h File Reference

```
#include <dtAnim/ical3ddriver.h>
#include <vector>
#include <dtCore/refptr.h>
#include <dtAnim/export.h>
```

Classes

- class [HotSpotDriver](#)
updates the body offset value for HotSpot instances.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.
- namespace [dtCore](#)

6.47 ical3ddriver.h File Reference

```
#include <osg/Referenced>
```

```
#include <dtAnim/export.h>
```

Classes

- class [ICal3DDriver](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.48 mainpage.h File Reference

6.48.1 Detailed Description

This file contains Doxygen special commands and text for the [Main Page](#) and some other minor aspects of this documentation. It is not part of Delta3D.

6.49 morphdriver.cpp File Reference

```
#include <dtAnim/morphdriver.h>
#include <cal3d/model.h>
#include <cal3d/morphtargetmixer.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.50 morphdriver.h File Reference

```
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
#include <dtAnim/ical3ddriver.h>
#include <dtAnim/cal3dmodelwrapper.h>
```

Classes

- class [MorphDriver](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.51 physiquedriver.cpp File Reference

```
#include <dtAnim/physiquedriver.h>
#include <cal3d/model.h>
#include <cal3d/physique.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.52 physiquedriver.h File Reference

```
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
#include <dtAnim/ical3ddriver.h>
#include <dtAnim/cal3dmodelwrapper.h>
```

Classes

- class [PhysiqueDriver](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.53 posemath.cpp File Reference

```
#include <dtAnim/posemesh.h>  
#include <dtAnim/posemath.h>  
#include <dtUtil/mathdefines.h>  
#include <osg/Quat>
```

6.54 posemath.h File Reference

```
#include "export.h"
#include <osg/Vec3>
#include <osg/Vec2>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

Functions

- void DT_ANIM_EXPORT [GetCelestialCoordinates](#) (osg::Vec3 targetDirection, const osg::Vec3 &lookForward, float &azimuth, float &elevation)
GetCelestialCoordinates - calculates the azimuth and elevation w.r.t.
- void DT_ANIM_EXPORT [GetCelestialDirection](#) (const float azimuth, const float elevation, const osg::Vec3 &forwardDirection, const osg::Vec3 &upDirection, osg::Vec3 &outDirection)
GetCelestialDirection - calculates the direction that a given azimuth and elevation points.
- void DT_ANIM_EXPORT [GetClosestPointOnSegment](#) (const osg::Vec3 &startPoint, const osg::Vec3 &endPoint, const osg::Vec3 &refPoint, osg::Vec3 &closestPoint)
GetClosestPointOnSegment - calculates the point on a segment that is closest to a reference point.
- bool DT_ANIM_EXPORT [IsPointBetweenVectors](#) (const osg::Vec3f &point, const osg::Vec3f &origin, const osg::Vec3f &A, const osg::Vec3f &B)
IsPointBetweenVectors - determines whether a points is between the vectors origin to A and origin to B.
- void DT_ANIM_EXPORT [MapCelestialToScreen](#) (float azimuth, float elevation, float maxDistance, float windowHeight, float windowWidth, float windowHeight, const osg::Vec2 &screenOrigin, osg::Vec2 &outScreenPos)
MapCelestialToScreen - maps azimuth and elevation to a screen position.

6.55 posemesh.cpp File Reference

```
#include <dtAnim/posemesh.h>
#include <dtAnim/posemath.h>
#include <dtAnim/posemeshxml.h>
#include <dtAnim/posemeshutility.h>
#include <dtUtil/log.h>
#include <dtUtil/mathdefines.h>
#include <dtUtil/exception.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <iostream>
#include <sstream>
```

6.56 posemesh.h File Reference

```
#include "export.h"
#include <vector>
#include <osg/Vec3>
#include <osg/Quat>
#include <osg/Geometry>
#include <osg/ref_ptr>
#include <map>
#include <string>
#include <dtUtil/export.h>
#include <dtUtil/barycentric.h>
```

Classes

- class [PoseMesh](#)
- struct [TargetTriangle](#)
- struct [Triangle](#)
- struct [TriangleEdge](#)
- struct [Vertex](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

Defines

- #define [TRIANGLE_NOT_FOUND](#) -1

6.56.1 Define Documentation

6.56.1.1 #define TRIANGLE_NOT_FOUND -1

6.57 posemeshdatabase.cpp File Reference

```
#include <dtAnim/posemeshdatabase.h>
#include <dtAnim/posemeshloader.h>
#include <dtAnim/posemesh.h>
#include <dtAnim/posemath.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtUtil/log.h>
#include <dtUtil/exception.h>
#include <algorithm>
#include <cstdint>
#include <istream>
```

Classes

- struct [DeletePointer< PtrT >](#)
- struct [PoseBuilderFunctor< ContainerT >](#)
- struct [PredicatePoseMeshName](#)

6.58 posemeshdatabase.h File Reference

```
#include "export.h"
#include <string>
#include <vector>
#include <osg/Vec3>
#include <osg/Quat>
#include <osg/Referenced>
#include <osg/ref_ptr>
```

Classes

- class [PoseMeshDatabase](#)
manager of the HotSpotData resources

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.59 posemeshloader.cpp File Reference

```
#include <dtAnim/posemeshloader.h>
#include <dtAnim/posemeshxml.h>
#include <dtUtil/xercesparser.h>
#include <dtUtil/log.h>
#include <dtUtil/exception.h>
```

6.60 posemeshloader.h File Reference

```
#include <vector>
#include <string>
#include "posemeshxml.h"
```

Classes

- class [PoseMeshLoader](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.61 posemeshutility.cpp File Reference

```
#include <dtAnim/posemeshutility.h>
#include <dtAnim/posemesh.h>
#include <dtAnim/posemath.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtUtil/mathdefines.h>
#include <dtUtil/log.h>
#include <algorithm>
```

Functions

- bool [BaseReferencePredicate](#) (const std::pair< int, float > &lhs, const std::pair< int, float > &rhs)

6.61.1 Function Documentation

6.61.1.1 bool [BaseReferencePredicate](#) (const std::pair< int, float > &*lhs*, const std::pair< int, float > &*rhs*)

6.62 posemeshutility.h File Reference

```
#include "posemesh.h"  
#include <osg/Referenced>
```

Classes

- struct [BaseReferenceBlend](#)
- class [PoseMeshUtility](#)
Convenience functionality for manipulating pose meshes.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.63 posemeshxml.cpp File Reference

```
#include <dtAnim/posemeshxml.h>
#include <dtAnim/posemesh.h>
#include <dtCore/exceptionenum.h>
#include <dtUtil/xercesutils.h>
#include <dtUtil/exception.h>
#include <cstddef>
#include <cassert>
#include <sstream>
```

6.64 posemeshxml.h File Reference

```
#include <xercesc/sax2/ContentHandler.hpp>
#include <vector>
#include <stack>
#include <string>
#include <osg/Vec3>
```

Classes

- struct [PoseMeshData](#)
- class [PoseMeshFileHandler](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

Typedefs

- typedef std::vector< std::string > [StringVector](#)

6.65 sequencemixer.cpp File Reference

```
#include <dtAnim/animatable.h>
#include <dtAnim/sequencemixer.h>
#include <dtUtil/log.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.66 sequencemixer.h File Reference

```
#include <dtAnim/export.h>
#include <dtAnim/animationsequence.h>
#include <osg/Referenced>
#include <dtCore/refptr.h>
#include <string>
#include <map>
#include <vector>
```

Classes

- class [SequenceMixer](#)
The SequenceMixer's job is to manage animations and animation sequences.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.67 skeletaldrawable.cpp File Reference

```
#include <dtAnim/skeletaldrawable.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <osg/Vec3>
#include <cstddef>
#include <algorithm>
#include <dtUtil/bits.h>
#include <dtUtil/log.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.68 skeletaldrawable.h File Reference

```
#include <dtAnim/export.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <osg/Drawable>
#include <dtCore/refptr.h>
```

Classes

- struct **CPrimitiveRenderObject**
- struct **IPrimitiveRenderObject**
- struct [RenderPrimitive](#)
a functor to be used in a loop algorithm
- class [SkeletalDrawable](#)
Renders only the skeleton.

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.69 skeletondriver.cpp File Reference

```
#include <dtAnim/skeletondriver.h>
#include <cal3d/model.h>
#include <cal3d/mixer.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.70 skeletondriver.h File Reference

```
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
#include <dtAnim/ical3ddriver.h>
#include <dtAnim/cal3dmodelwrapper.h>
```

Classes

- class [SkeletonDriver](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.71 springdriver.cpp File Reference

```
#include <dtAnim/springdriver.h>
#include <cal3d/model.h>
#include <cal3d/springsystem.h>
```

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.72 springdriver.h File Reference

```
#include <dtAnim/export.h>
#include <dtCore/refptr.h>
#include <dtAnim/ical3ddriver.h>
#include <dtAnim/cal3dmodelwrapper.h>
```

Classes

- class [SpringDriver](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

6.73 submesh.cpp File Reference

```
#include <dtAnim/submesh.h>
#include <dtAnim/cal3dmodelwrapper.h>
#include <dtAnim/cal3dmodeldata.h>
#include <dtAnim/cal3ddatabase.h>
#include <dtUtil/log.h>
#include <dtUtil/mathdefines.h>
#include <osg/Material>
#include <osg/Texture2D>
#include <osg/PolygonMode>
#include <osg/Geometry>
#include <osg/CullFace>
#include <osg/Math>
#include <osg/BlendFunc>
#include <osg/GLExtensions>
#include <cassert>
```

Classes

- class [SubmeshComputeBound](#)

Namespaces

- namespace [dtAnim](#)

The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

Defines

- #define [BUFFER_OFFSET\(x\)](#) ((GLvoid*) (0 + ((x) * sizeof(float))))
- #define [INDEX_OFFSET\(x\)](#) ((GLvoid*) (0 + ((x) * sizeof(CallIndex))))

6.73.1 Define Documentation

6.73.1.1 #define [BUFFER_OFFSET\(x\)](#) ((GLvoid*) (0 + ((x) * sizeof(float))))

6.73.1.2 #define [INDEX_OFFSET\(x\)](#) ((GLvoid*) (0 + ((x) * sizeof(CallIndex))))

6.74 submesh.h File Reference

```
#include <osg/Drawable>
#include <cal3d/cal3d.h>
#include <dtAnim/export.h>
#include <dtAnim/cal3dmodeldata.h>
#include <dtCore/refptr.h>
#include <osg/Geometry>
```

Classes

- class [SubmeshCullCallback](#)
- class [SubmeshDirtyCallback](#)
- class [SubmeshDrawable](#)
Adapter that converts cal3d submeshes into osg::Drawables.
- class [SubmeshUserData](#)

Namespaces

- namespace [dtAnim](#)
The [dtAnim](#) Library contains functionality used to support skeletal mesh animations.

Index

- Symbols -

- ~AnimDriver
 - dtAnim::AnimDriver, [47](#)
- ~AnimNodeBuilder
 - dtAnim::AnimNodeBuilder, [49](#)
- ~Animatable
 - dtAnim::Animatable, [21](#)
- ~AnimationChannel
 - dtAnim::AnimationChannel, [26](#)
- ~AnimationComponent
 - dtAnim::AnimationComponent, [30](#)
- ~AnimationController
 - dtAnim::AnimationSequence::AnimationController, [32](#)
- ~AnimationGameActor
 - dtAnim::AnimationGameActor, [34](#)
- ~AnimationGameActorProxy
 - dtAnim::AnimationGameActorProxy, [35](#)
- ~AnimationHelper
 - dtAnim::AnimationHelper, [38](#)
- ~AnimationSequence
 - dtAnim::AnimationSequence, [41](#)
- ~AnimationWrapper
 - dtAnim::AnimationWrapper, [45](#)
- ~AttachmentController
 - dtAnim::AttachmentController, [53](#)
- ~Cal3DAnimator
 - dtAnim::Cal3DAnimator, [58](#)
- ~Cal3DDatabase
 - dtAnim::Cal3DDatabase, [61](#)
- ~Cal3DGameActor
 - dtAnim::Cal3DGameActor, [63](#)
- ~Cal3DGameActorProxy
 - dtAnim::Cal3DGameActorProxy, [65](#)
- ~Cal3DLoader
 - dtAnim::Cal3DLoader, [67](#)
- ~Cal3DModelData
 - dtAnim::Cal3DModelData, [69](#)
- ~Cal3DModelWrapper
 - dtAnim::Cal3DModelWrapper, [74](#)
- ~CharDrawable
 - dtAnim::CharDrawable, [89](#)
- ~CharacterFileHandler
 - dtAnim::CharacterFileHandler, [82](#)
- ~CharacterWrapper
 - dtAnim::CharacterWrapper, [86](#)
- ~CreateGeometryDrawCallback
 - dtAnim::CreateGeometryDrawCallback, [92](#)
- ~HardwareSubmeshDrawable
 - dtAnim::HardwareSubmeshDrawable, [98](#)
- ~HotSpotDriver
 - dtAnim::HotSpotDriver, [99](#)
- ~MorphDriver
 - dtAnim::MorphDriver, [106](#)
- ~PhysiqueDriver
 - dtAnim::PhysiqueDriver, [107](#)
- ~PoseMesh
 - dtAnim::PoseMesh, [110](#)
- ~PoseMeshDatabase
 - dtAnim::PoseMeshDatabase, [113](#)
- ~PoseMeshFileHandler

- dtAnim::PoseMeshFileHandler, [115](#)
- ~PoseMeshLoader
 - dtAnim::PoseMeshLoader, [116](#)
- ~PoseMeshUtility
 - dtAnim::PoseMeshUtility, [117](#)
- ~SequenceMixer
 - dtAnim::SequenceMixer, [124](#)
- ~SkeletalDrawable
 - dtAnim::SkeletalDrawable, [126](#)
- ~SkeletonDriver
 - dtAnim::SkeletonDriver, [127](#)
- ~SpringDriver
 - dtAnim::SpringDriver, [128](#)
- ~SubmeshDrawable
 - dtAnim::SubmeshDrawable, [132](#)
- ~UpdateCallback
 - dtAnim::UpdateCallback, [138](#)

- A -

- accept
 - dtAnim::SubmeshDrawable, [132](#)
- Add
 - dtAnim::Cal3DModelData, [69](#)
- AddAnimation
 - dtAnim::AnimationSequence, [41](#)
- AddAttachment
 - dtAnim::AttachmentController, [53](#)
- AddedToScene
 - dtAnim::Cal3DGameActor, [63](#)
- AddHotSpot
 - dtAnim::HotSpotDriver, [99](#)
- AnimActorRegistry
 - dtAnim::AnimActorRegistry, [19](#)
- animactorregistry.cpp, [141](#)
 - CreatePluginRegistry, [141](#)
 - DestroyPluginRegistry, [141](#)
- animactorregistry.h, [142](#)
- Animatable
 - dtAnim::Animatable, [21](#)
- animatable.cpp, [143](#)
- animatable.h, [144](#)
- AnimatableArray
 - dtAnim::Cal3DModelData, [69](#)
- AnimatableStruct
 - dtAnim::CharacterFileHandler::AnimatableStruct, [24](#)
- ANIMATION_ACTOR_TYPE
 - dtAnim::AnimActorRegistry, [19](#)
- ANIMATION_BLEND_DELAY
 - dtAnim::Cal3DGameActor::PropertyNames, [120](#)
- ANIMATION_BLEND_GROUP
 - dtAnim::Cal3DGameActor::PropertyNames, [120](#)
- ANIMATION_BLEND_ID
 - dtAnim::Cal3DGameActor::PropertyNames, [120](#)
- ANIMATION_BLEND_WEIGHT
 - dtAnim::Cal3DGameActor::PropertyNames, [120](#)
- ANIMATION_ELEMENT
 - dtAnim::CharacterFileHandler, [82](#)
- ANIMATION_GROUP
 - dtAnim::Cal3DGameActor::PropertyNames, [120](#)
- ANIMATION_GROUP_LABEL

- dtAnim::Cal3DGameActor::PropertyNames, 120
- ANIMATION_NAME_ELEMENT
 - dtAnim::CharacterFileHandler, 82
- ANIMATION_NODE
 - dtAnim::PoseMeshFileHandler, 115
- AnimationCallback
 - dtAnim, 12
- AnimationChannel
 - dtAnim::AnimationChannel, 26
- animationchannel.cpp, 145
- animationchannel.h, 146
- AnimationChannelStruct
 - dtAnim::CharacterFileHandler::AnimationChannelStruct, 28
- AnimationComponent
 - dtAnim::AnimationComponent, 30
- animationcomponent.cpp, 147
- animationcomponent.h, 148
- AnimationContainer
 - dtAnim::AnimationSequence, 41
- AnimationController
 - dtAnim::AnimationSequence::AnimationController, 32
- AnimationGameActor
 - dtAnim::AnimationGameActor, 34
- animationgameactor.cpp, 149
- animationgameactor.h, 150
- AnimationGameActorProxy
 - dtAnim::AnimationGameActorProxy, 35
- AnimationHelper
 - dtAnim::AnimationHelper, 38
- animationhelper.cpp, 151
- animationhelper.h, 152
- AnimationSequence
 - dtAnim::AnimationSequence, 41
- animationsequence.cpp, 153
- animationsequence.h, 154
- AnimationSequenceStruct
 - dtAnim::CharacterFileHandler::AnimationSequenceStruct, 43
- AnimationTable
 - dtAnim::SequenceMixer, 124
- AnimationWrapper
 - dtAnim::AnimationWrapper, 45
- animationwrapper.cpp, 155
- animationwrapper.h, 156
- AnimationWrapperArray
 - dtAnim::Cal3DModelData, 69
- AnimComplter
 - dtAnim::AnimationComponent, 30
- AnimCompMap
 - dtAnim::AnimationComponent, 30
- AnimCompMapping
 - dtAnim::AnimationComponent, 30
- AnimDriver
 - dtAnim::AnimDriver, 47
- animdriver.cpp, 157
- animdriver.h, 158
- AnimNodeBuilder
 - dtAnim::AnimNodeBuilder, 49
- animnodebuilder.cpp, 159
- animnodebuilder.h, 160
- AnimSeqForceFade
 - dtAnim::AnimSeqForceFade, 51
- AnimSequenceUpdater
 - dtAnim::AnimSequenceUpdater, 52
- ApplyAnimationGroup
 - dtAnim::Cal3DGameActor, 63
- ApplyCoreModelScaleFactor
 - dtAnim::Cal3DModelWrapper, 74
- AttachmentContainer
 - dtAnim::AttachmentController, 53
- AttachmentController
 - dtAnim::AttachmentController, 53
- attachmentcontroller.cpp, 161
- attachmentcontroller.h, 162
- AttachmentMover
 - dtAnim::AttachmentMover, 55
- AttachmentPair
 - dtAnim, 12
- AttachMesh
 - dtAnim::Cal3DModelWrapper, 74
- B -
- Barycentric2D
 - dtAnim::PoseMesh, 110
- Barycentric2DVector
 - dtAnim::PoseMesh, 110
- BASE_WEIGHT_ELEMENT
 - dtAnim::CharacterFileHandler, 82
- BaseClass
 - dtAnim::AnimationComponent, 30
 - dtAnim::CharacterWrapper, 86
 - dtAnim::SubmeshUserData, 134
- BaseReferencePose
 - dtAnim::PoseMeshUtility, 117
- BaseReferencePredicate
 - posemeshutility.cpp, 201
- BeginRenderingQuery
 - dtAnim::Cal3DModelWrapper, 74
- blendAlpha
 - dtAnim::PoseMeshUtility::BaseReferenceBlend, 56
- BlendCycle
 - dtAnim::Cal3DModelWrapper, 74
- BlendPoses
 - dtAnim::PoseMeshUtility, 117
- BUFFER_OFFSET
 - hardwaresubmesh.cpp, 183
 - submesh.cpp, 213
- BuildInvokables
 - dtAnim::Cal3DGameActorProxy, 65
- BuildPropertyMap
 - dtAnim::AnimationGameActorProxy, 35
 - dtAnim::Cal3DGameActorProxy, 65
- C -
- CAL3D_ACTOR_TYPE
 - dtAnim::AnimActorRegistry, 19
- Cal3DAnimator
 - dtAnim::Cal3DAnimator, 58
- cal3danimator.cpp, 163
- cal3danimator.h, 164
- Cal3DBoundingSphereCalculator
 - dtAnim::AnimNodeBuilder::Cal3DBoundingSphereCalculator, 59
- Cal3DDatabase
 - dtAnim::Cal3DDatabase, 61
- cal3ddatabase.cpp, 165
- cal3ddatabase.h, 166
- Cal3DGameActor
 - dtAnim::Cal3DGameActor, 63
- cal3dgameactor.cpp, 167

- cal3dgameactor.h, 168
- Cal3DGameActorProxy
 - dtAnim::Cal3DGameActorProxy, 65
- Cal3DLoader
 - dtAnim::Cal3DLoader, 67
- cal3dloader.cpp, 169
- cal3dloader.h, 170
- Cal3DModelData
 - dtAnim::Cal3DModelData, 69
- cal3dmodeldata.cpp, 171
- cal3dmodeldata.h, 172
- Cal3DModelWrapper
 - dtAnim::Cal3DModelWrapper, 74
- cal3dmodelwrapper.cpp, 173
- cal3dmodelwrapper.h, 174
- CHANNEL_ELEMENT
 - dtAnim::CharacterFileHandler, 82
- CHARACTER_ELEMENT
 - dtAnim::CharacterFileHandler, 82
- CHARACTER_XML_LOGGER
 - dtAnim::CharacterFileHandler, 82
- CharacterFileHandler
 - dtAnim::CharacterFileHandler, 82
- characterfilehandler.cpp, 175
 - XERCES_CPP_NAMESPACE_USE, 175
- characterfilehandler.h, 176
- characters
 - dtAnim::CharacterFileHandler, 82
 - dtAnim::PoseMeshFileHandler, 115
- CharacterWrapper
 - dtAnim::CharacterWrapper, 86
- characterwrapper.cpp, 177
- characterwrapper.h, 178
- CharDrawable
 - dtAnim::CharDrawable, 89
- chardrawable.cpp, 179
- chardrawable.h, 180
- CHILD_ELEMENT
 - dtAnim::CharacterFileHandler, 82
- className
 - dtAnim::SubmeshUserData, 134
- ClearActiveAnimations
 - dtAnim::SequenceMixer, 124
- ClearAll
 - dtAnim::AnimationHelper, 38
 - dtAnim::Cal3DModelWrapper, 74
- ClearAllAnimations
 - dtAnim::CharacterWrapper, 86
- ClearAnimation
 - dtAnim::AnimationHelper, 38
 - dtAnim::AnimationSequence, 41
 - dtAnim::CharacterWrapper, 86
 - dtAnim::SequenceMixer, 124
- ClearCycle
 - dtAnim::Cal3DModelWrapper, 74
- ClearPoses
 - dtAnim::PoseMeshUtility, 117
- ClearRegisteredAnimations
 - dtAnim::SequenceMixer, 124
- Clone
 - dtAnim::Animatable, 21
 - dtAnim::AnimationChannel, 26
 - dtAnim::AnimationSequence, 41
 - dtAnim::AnimationSequence::AnimationController, 33
- clone
 - dtAnim::HardwareSubmeshDrawable, 98
- dtAnim::SkeletalDrawable, 126
- dtAnim::SubmeshDrawable, 132
- dtAnim::SubmeshUserData, 134
- CloneFuncor
 - dtAnim::CloneFuncor, 91
- cloneType
 - dtAnim::HardwareSubmeshDrawable, 98
 - dtAnim::SkeletalDrawable, 126
 - dtAnim::SubmeshDrawable, 133
 - dtAnim::SubmeshUserData, 134
- computeBound
 - dtAnim::AnimNodeBuilder::Cal3DBoundingBoxSphereCalculator, 59
 - dtAnim::HardwareSubmeshComputeBound, 97
 - dtAnim::SubmeshComputeBound, 129
- ContainerType
 - dtAnim::AnimationSequence, 41
- CreateActor
 - dtAnim::AnimationGameActorProxy, 35
 - dtAnim::Cal3DGameActorProxy, 65
- CreateFunc
 - dtAnim::AnimNodeBuilder, 49
- CreateGeometryDrawCallback
 - dtAnim::CreateGeometryDrawCallback, 92
- CreateHardware
 - dtAnim::AnimNodeBuilder, 49
- CreateNode
 - dtAnim::AnimNodeBuilder, 49
- CreateNULL
 - dtAnim::AnimNodeBuilder, 49
- CreatePluginRegistry
 - animactorregistry.cpp, 141
- CreateSoftware
 - dtAnim::AnimNodeBuilder, 49
- CreateSoftwareInternal
 - dtAnim::AnimNodeBuilder, 49
- CreateSoftwareNoVBO
 - dtAnim::AnimNodeBuilder, 49
- cull
 - dtAnim::SubmeshCullCallback, 130
- D -**
- DEFAULT_NAME
 - dtAnim::AnimationComponent, 31
- DEFAULT_VALUE
 - dtAnim::PoseMeshFileHandler, 115
- DeletePointer, 93
 - operator(), 93
- DestroyPluginRegistry
 - animactorregistry.cpp, 141
- DetachMesh
 - dtAnim::Cal3DModelWrapper, 74
- drawImplementation
 - dtAnim::CreateGeometryDrawCallback, 92
 - dtAnim::HardwareSubmeshDrawable, 98
 - dtAnim::SkeletalDrawable, 126
 - dtAnim::SubmeshDrawable, 133
- DT_ANIM_EXPORT
 - export.h, 182
- dtAnim, 9
 - AnimationCallback, 12
 - AttachmentPair, 12
 - FindWithFuncor, 12
 - GetCelestialCoordinates, 12
 - GetCelestialDirection, 12
 - GetClosestPointOnSegment, 12

- IsPointBetweenVectors, 12
- MapCelestialToScreen, 12
- StringVector, 12
- dtanim.h, 181
- dtAnim::AnimActorRegistry, 19
 - AnimActorRegistry, 19
 - ANIMATION_ACTOR_TYPE, 19
 - CAL3D_ACTOR_TYPE, 19
 - RegisterActorTypes, 19
- dtAnim::Animatable, 20
 - ~Animatable, 21
 - Animatable, 21
 - Clone, 21
 - ForceFadeOut, 21
 - GetBaseWeight, 21
 - GetCurrentWeight, 22
 - GetElapsedTime, 22
 - GetEndTime, 22
 - GetFadeIn, 22
 - GetFadeOut, 22
 - GetName, 22
 - GetSpeed, 22
 - GetStartDelay, 22
 - GetStartTime, 22
 - IsActive, 22
 - operator=, 22
 - Prune, 22
 - Recalculate, 22
 - SetActive, 22
 - SetBaseWeight, 22
 - SetCurrentWeight, 22
 - SetElapsedTime, 22
 - SetEndCallback, 22
 - SetEndTime, 22
 - SetFadeIn, 23
 - SetFadeOut, 23
 - SetName, 23
 - SetPrune, 23
 - SetSpeed, 23
 - SetStartDelay, 23
 - SetStartTime, 23
 - ShouldPrune, 23
 - Update, 23
- dtAnim::AnimationChannel, 25
 - ~AnimationChannel, 26
 - AnimationChannel, 26
 - Clone, 26
 - ForceFadeOut, 26
 - GetAnimation, 26
 - GetMaxDuration, 26
 - GetModel, 26
 - IsAction, 27
 - IsLooping, 27
 - operator=, 27
 - Prune, 27
 - Recalculate, 27
 - SetAction, 27
 - SetAnimation, 27
 - SetLooping, 27
 - SetMaxDuration, 27
 - SetModel, 27
 - Update, 27
- dtAnim::AnimationComponent, 29
 - ~AnimationComponent, 30
 - AnimationComponent, 30
 - AnimComplter, 30
 - AnimCompMap, 30
 - AnimCompMapping, 30
 - BaseClass, 30
 - DEFAULT_NAME, 31
 - GetEyePointActor, 30
 - GetGroundClamper, 30
 - GetHelperForProxy, 30
 - GetTerrainActor, 30
 - GroundClamp, 30
 - IsRegisteredActor, 30
 - ProcessMessage, 30
 - RegisterActor, 30
 - SetEyePointActor, 31
 - SetGroundClamper, 31
 - SetTerrainActor, 31
 - TickLocal, 31
 - UnregisterActor, 31
- dtAnim::AnimationGameActor, 34
 - ~AnimationGameActor, 34
 - AnimationGameActor, 34
 - GetHelper, 34
 - mHelper, 34
 - SetModel, 34
- dtAnim::AnimationGameActorProxy, 35
 - ~AnimationGameActorProxy, 35
 - AnimationGameActorProxy, 35
 - BuildPropertyMap, 35
 - CreateActor, 35
 - GetBillBoardIcon, 35
 - GetRenderMode, 35
- dtAnim::AnimationHelper, 37
 - ~AnimationHelper, 38
 - AnimationHelper, 38
 - ClearAll, 38
 - ClearAnimation, 38
 - GetActorProperties, 38
 - GetAnimator, 38
 - GetAttachmentController, 38
 - GetGroundClamp, 38
 - GetModelWrapper, 38
 - GetNode, 38, 39
 - GetSequenceMixer, 39
 - LoadModel, 39
 - PlayAnimation, 39
 - PROPERTY_SKELETAL_MESH, 39
 - SetAttachmentController, 39
 - SetGroundClamp, 39
 - Update, 39
- dtAnim::AnimationSequence, 40
 - ~AnimationSequence, 41
 - AddAnimation, 41
 - AnimationContainer, 41
 - AnimationSequence, 41
 - ClearAnimation, 41
 - Clone, 41
 - ContainerType, 41
 - ForceFadeOut, 41
 - GetAnimation, 42
 - GetChildAnimations, 42
 - GetController, 42
 - IsAnimationPlaying, 42
 - operator=, 42
 - Prune, 42
 - Recalculate, 42
 - SetController, 42
 - Update, 42

- dtAnim::AnimationSequence::AnimationController, 32
 - ~AnimationController, 32
 - AnimationController, 32
 - Clone, 33
 - GetParent, 33
 - operator=, 33
 - Recalculate, 33
 - SetComputeSpeed, 33
 - SetComputeWeight, 33
 - SetParent, 33
 - Update, 33
- dtAnim::AnimationWrapper, 45
 - ~AnimationWrapper, 45
 - AnimationWrapper, 45
 - GetDuration, 45
 - GetID, 45
 - GetName, 45
 - GetSpeed, 45
 - SetDuration, 45
 - SetName, 46
 - SetSpeed, 46
- dtAnim::AnimDriver, 47
 - ~AnimDriver, 47
 - AnimDriver, 47
 - SetWrapper, 47
 - Update, 47
- dtAnim::AnimNodeBuilder, 48
 - ~AnimNodeBuilder, 49
 - AnimNodeBuilder, 49
 - CreateFunc, 49
 - CreateHardware, 49
 - CreateNode, 49
 - CreateNULL, 49
 - CreateSoftware, 49
 - CreateSoftwareInternal, 49
 - CreateSoftwareNoVBO, 49
 - GetCreate, 49
 - LoadShaders, 49
 - operator=, 49
 - SetCreate, 49
 - SupportsHardware, 49
 - SupportsSoftware, 49
- dtAnim::AnimNodeBuilder::Cal3DBoundingBoxSphereCalculator, 59
 - Cal3DBoundingBoxSphereCalculator, 59
 - computeBound, 59
- dtAnim::AnimSeqForceFade, 51
 - AnimSeqForceFade, 51
 - operator(), 51
- dtAnim::AnimSequenceUpdater, 52
 - AnimSequenceUpdater, 52
 - operator(), 52
- dtAnim::AttachmentController, 53
 - ~AttachmentController, 53
 - AddAttachment, 53
 - AttachmentContainer, 53
 - AttachmentController, 53
 - GetAttachments, 53
 - RemoveAttachment, 53
 - Update, 53
- dtAnim::AttachmentMover, 55
 - AttachmentMover, 55
 - operator(), 55
 - operator=, 55
- dtAnim::Cal3DAnimator, 57
 - ~Cal3DAnimator, 58
 - Cal3DAnimator, 58
 - GetAnimationDriver, 58
 - GetMorphTargetDriver, 58
 - GetPhysiqueDriver, 58
 - GetPostDriver, 58
 - GetPreDriver, 58
 - GetSkeletonDriver, 58
 - GetSpringDriver, 58
 - GetWrapper, 58
 - SetAnimationDriver, 58
 - SetMorphTargetDriver, 58
 - SetPhysiqueDriver, 58
 - SetPostDriver, 58
 - SetPreDriver, 58
 - SetSkeletonDriver, 58
 - SetSpringDriver, 58
 - SetWrapper, 58
 - Update, 58
- dtAnim::Cal3DDatabase, 60
 - ~Cal3DDatabase, 61
 - Cal3DDatabase, 61
 - Find, 61
 - GetInstance, 61
 - GetModelData, 61
 - GetNodeBuilder, 61
 - Load, 61
 - mFileLoader, 61
 - mInstance, 61
 - mModelData, 61
 - mNodeBuilder, 61
 - ModelDataArray, 61
 - PurgeLoaderCaches, 61
 - TruncateDatabase, 61
- dtAnim::Cal3DGameActor, 62
 - ~Cal3DGameActor, 63
 - AddedToScene, 63
 - ApplyAnimationGroup, 63
 - Cal3DGameActor, 63
 - GetAnimator, 63
 - GetRenderMode, 63
 - MakeAnimationGroup, 63
 - mAnimator, 64
 - mModelGeode, 64
 - mModelLoader, 64
 - mRenderModeBits, 64
 - mSkeletalGeode, 64
 - OnEnteredWorld, 63
 - OnTickLocal, 63
 - RENDER_MODE_BONES, 63
 - RENDER_MODE_NONE, 63
 - RENDER_MODE_SKIN, 63
 - RenderModeBitContainer, 63
 - RenderModeBits, 63
 - SetModel, 63
 - SetRenderMode, 63
- dtAnim::Cal3DGameActor::PropertyNames, 120
 - ANIMATION_BLEND_DELAY, 120
 - ANIMATION_BLEND_GROUP, 120
 - ANIMATION_BLEND_ID, 120
 - ANIMATION_BLEND_WEIGHT, 120
 - ANIMATION_GROUP, 120
 - ANIMATION_GROUP_LABEL, 120
 - RENDER_MODE, 120
 - RENDER_MODE_LABEL, 120
- dtAnim::Cal3DGameActorProxy, 65
 - ~Cal3DGameActorProxy, 65

- BuildInvokables, 65
- BuildPropertyMap, 65
- Cal3DGameActorProxy, 65
- CreateActor, 65
- GetBillBoardIcon, 66
- GetRenderMode, 66
- dtAnim::Cal3DLoader, 67
 - ~Cal3DLoader, 67
 - Cal3DLoader, 67
 - Load, 67
 - PurgeAllCaches, 67
- dtAnim::Cal3DModelData, 68
 - ~Cal3DModelData, 69
 - Add, 69
 - AnimatableArray, 69
 - AnimationWrapperArray, 69
 - Cal3DModelData, 69
 - GetAnimatables, 69
 - GetAnimationWrappers, 69
 - GetCoreModel, 69
 - GetFilename, 69
 - GetIndexVBO, 69
 - GetLODOptions, 69
 - GetPoseMeshFilename, 69
 - GetShaderGroupName, 69
 - GetShaderMaxBones, 69
 - GetShaderName, 69
 - GetVertexVBO, 69
 - operator=, 69
 - Remove, 69
 - SetIndexVBO, 69
 - SetPoseMeshFilename, 69
 - SetShaderGroupName, 70
 - SetShaderMaxBones, 70
 - SetShaderName, 70
 - SetVertexVBO, 70
- dtAnim::Cal3DModelWrapper, 71
 - ~Cal3DModelWrapper, 74
 - ApplyCoreModelScaleFactor, 74
 - AttachMesh, 74
 - BeginRenderingQuery, 74
 - BlendCycle, 74
 - Cal3DModelWrapper, 74
 - ClearAll, 74
 - ClearCycle, 74
 - DetachMesh, 74
 - EndRenderingQuery, 74
 - ExecuteAction, 74
 - GetAmbientColor, 74
 - GetAnimationTime, 74
 - GetBoneAbsoluteRotation, 74
 - GetBoneAbsoluteRotationForKeyFrame, 75
 - GetBoneAbsoluteTranslation, 75
 - GetBoneRelativeRotation, 75
 - GetBoundingBox, 75
 - GetCalModel, 75
 - GetCoreAnimationCount, 75
 - GetCoreAnimationDuration, 75
 - GetCoreAnimationIDByName, 75
 - GetCoreAnimationKeyframeCount, 75
 - GetCoreAnimationKeyframeCountForTrack, 75
 - GetCoreAnimationName, 75
 - GetCoreAnimationTrackCount, 76
 - GetCoreBoneChildrenIDs, 76
 - GetCoreBoneID, 76
 - GetCoreBoneNames, 76
 - GetCoreMaterial, 76
 - GetCoreMaterialAmbient, 76
 - GetCoreMaterialCount, 76
 - GetCoreMaterialDiffuse, 76
 - GetCoreMaterialName, 76
 - GetCoreMaterialShininess, 76
 - GetCoreMaterialSpecular, 76
 - GetCoreMeshCount, 76
 - GetCoreMeshName, 76
 - GetCoreTrackKeyFrameQuat, 76
 - GetDiffuseColor, 76
 - GetFaceCount, 77
 - GetFaces, 77
 - GetMapCount, 77
 - GetMapUserData, 77
 - GetMeshCount, 77
 - GetNormals, 77
 - GetOrCreateCalHardwareModel, 77
 - GetParentBoneID, 77
 - GetRootBoneIDs, 77
 - GetShininess, 77
 - GetSpecularColor, 77
 - GetSubmeshCount, 77
 - GetTextureCoords, 77
 - GetVertexCount, 77
 - GetVertices, 77
 - HasAnimation, 77
 - HasBone, 77
 - HasTrackForBone, 77
 - HideMesh, 77
 - IsMeshVisible, 77
 - NULL_BONE, 78
 - RemoveAction, 77
 - SelectMeshSubmesh, 77
 - SetAnimationTime, 77
 - SetCalModel, 77
 - SetLODLevel, 78
 - SetMaterialSet, 78
 - ShowMesh, 78
 - Update, 78
 - UpdateAnimation, 78
 - UpdateMorphTargetMixer, 78
 - UpdatePhysique, 78
 - UpdateSkeleton, 78
 - UpdateSpringSystem, 78
- dtAnim::CharacterFileHandler, 79
 - ~CharacterFileHandler, 82
 - ANIMATION_ELEMENT, 82
 - ANIMATION_NAME_ELEMENT, 82
 - BASE_WEIGHT_ELEMENT, 82
 - CHANNEL_ELEMENT, 82
 - CHARACTER_ELEMENT, 82
 - CHARACTER_XML_LOGGER, 82
 - CharacterFileHandler, 82
 - characters, 82
 - CHILD_ELEMENT, 82
 - endDocument, 82
 - endElement, 82
 - endPrefixMapping, 82
 - FADE_IN_ELEMENT, 82
 - FADE_OUT_ELEMENT, 82
 - FILENAME_ELEMENT, 82
 - ignorableWhitespace, 82
 - IS_ACTION_ELEMENT, 82
 - IS_LOOPING_ELEMENT, 82
 - LOD_ELEMENT, 82

- LOD_END_DISTANCE_ELEMENT, 82
- LOD_START_DISTANCE_ELEMENT, 82
- mAnimationChannels, 82
- mAnimations, 82
- mAnimationSequences, 82
- MATERIAL_ELEMENT, 83
- MAX_DURATION_ELEMENT, 83
- MAX_VISIBLE_DISTANCE_ELEMENT, 83
- MESH_ELEMENT, 83
- mFoundLODOptions, 83
- mFoundScale, 83
- mLODEndDistance, 83
- mLODMaxVisibleDistance, 83
- mLODStartDistance, 83
- mMaterials, 83
- mMeshes, 83
- mMorphAnimations, 83
- mName, 83
- MORPH_ANIMATION_ELEMENT, 83
- mPoseMeshFilename, 83
- mScale, 83
- mShaderGroup, 83
- mShaderMaxBones, 83
- mShaderName, 83
- mSkeletonFilename, 83
- NAME_ELEMENT, 83
- POSEMESH_ELEMENT, 84
- processingInstruction, 82
- SCALE_ELEMENT, 84
- SCALE_FACTOR_ELEMENT, 84
- SEQUENCE_ELEMENT, 84
- setDocumentLocator, 82
- SHADER_GROUP_ELEMENT, 84
- SHADER_MAX_BONES_ELEMENT, 84
- SHADER_NAME_ELEMENT, 84
- SKELETON_ELEMENT, 84
- SKINNING_SHADER_ELEMENT, 84
- skippedEntity, 82
- SPEED_ELEMENT, 84
- START_DELAY_ELEMENT, 84
- startDocument, 82
- startElement, 82
- startPrefixMapping, 82
- dtAnim::CharacterFileHandler::AnimatableStruct, 24
 - AnimatableStruct, 24
 - mBaseWeight, 24
 - mFadeln, 24
 - mFadeOut, 24
 - mName, 24
 - mSpeed, 24
 - mStartDelay, 24
- dtAnim::CharacterFileHandler::AnimationChannelStruct, 28
 - AnimationChannelStruct, 28
 - mAnimationName, 28
 - mIsAction, 28
 - mIsLooping, 28
 - mMaxDuration, 28
- dtAnim::CharacterFileHandler::AnimationSequenceStruct, 43
 - AnimationSequenceStruct, 43
 - mChildNames, 43
- dtAnim::CharacterFileHandler::AnimationStruct, 44
 - mFileName, 44
 - mName, 44
- dtAnim::CharacterFileHandler::MaterialStruct, 103
 - mFileName, 103
 - mName, 103
- dtAnim::CharacterFileHandler::MeshStruct, 104
 - mFileName, 104
 - mName, 104
- dtAnim::CharacterFileHandler::MorphAnimationStruct, 105
 - mFileName, 105
 - mName, 105
- dtAnim::CharacterWrapper, 85
 - ~CharacterWrapper, 86
 - BaseClass, 86
 - CharacterWrapper, 86
 - ClearAllAnimations, 86
 - ClearAnimation, 86
 - GetAnimationHelper, 86
 - GetHeading, 87
 - GetHeightAboveGround, 87
 - GetLocalOffset, 87
 - GetRotationSpeed, 87
 - GetSpeed, 87
 - IsAnimationPlaying, 87
 - PlayAnimation, 87
 - RotateToHeading, 87
 - RotateToPoint, 87
 - SetGroundClamp, 87, 88
 - SetHeading, 88
 - SetHeightAboveGround, 88
 - SetLocalOffset, 88
 - SetRotationSpeed, 88
 - SetSpeed, 88
 - Update, 88
- dtAnim::CharDrawable, 89
 - ~CharDrawable, 89
 - CharDrawable, 89
 - GetCal3DWrapper, 89
 - GetNode, 89
 - mAnimator, 90
 - mLastMeshCount, 90
 - mNode, 90
 - OnMessage, 89
 - RebuildSubmeshes, 89
 - SetCal3DWrapper, 89
- dtAnim::CloneFunctor, 91
 - CloneFunctor, 91
 - operator(), 91
- dtAnim::CreateGeometryDrawCallback, 92
 - ~CreateGeometryDrawCallback, 92
 - CreateGeometryDrawCallback, 92
 - drawImplementation, 92
 - mCreatedNode, 92
- dtAnim::findWithCoreModel, 94
 - findWithCoreModel, 94
 - mModel, 94
 - operator(), 94
- dtAnim::findWithFilename, 95
 - findWithFilename, 95
 - mFilename, 95
 - operator(), 95
- dtAnim::HardwareSubmeshCallback, 96
 - HardwareSubmeshCallback, 96
 - update, 96
- dtAnim::HardwareSubmeshComputeBound, 97
 - computeBound, 97
 - HardwareSubmeshComputeBound, 97
 - mDefaultBox, 97
- dtAnim::HardwareSubmeshDrawable, 98
 - ~HardwareSubmeshDrawable, 98
 - clone, 98

- cloneType, 98
- drawImplementation, 98
- HardwareSubmeshDrawable, 98
- SetBoundingBox, 98
- dtAnim::HotSpotDriver, 99
 - ~HotSpotDriver, 99
 - AddHotSpot, 99
 - GetHotSpots, 99
 - HotSpotContainer, 99
 - HotSpotDriver, 99
 - RemoveHotSpot, 99
 - SetWrapper, 99
 - Update, 99
- dtAnim::ICal3DDriver, 100
 - SetWrapper, 100
 - Update, 100
- dtAnim::IsActor, 101
 - IsActor, 101
 - operator(), 101
- dtAnim::LODOptions, 102
 - GetEndDistance, 102
 - GetMaxVisibleDistance, 102
 - GetStartDistance, 102
 - LODOptions, 102
 - SetEndDistance, 102
 - SetMaxVisibleDistance, 102
 - SetStartDistance, 102
- dtAnim::MorphDriver, 106
 - ~MorphDriver, 106
 - MorphDriver, 106
 - SetWrapper, 106
 - Update, 106
- dtAnim::PhysiqueDriver, 107
 - ~PhysiqueDriver, 107
 - PhysiqueDriver, 107
 - SetWrapper, 107
 - Update, 107
- dtAnim::PoseMesh, 109
 - ~PoseMesh, 110
 - Barycentric2D, 110
 - Barycentric2DVector, 110
 - EdgeLineMap, 110
 - FindPoseTriangleID, 110
 - GetBarySpaces, 110
 - GetEffectorForwardAxis, 110
 - GetEffectorID, 110
 - GetEffectorName, 110
 - GetIndexPairsForTriangle, 110
 - GetName, 110
 - GetRootForwardAxis, 110
 - GetSilhouette, 110
 - GetTargetTriangleData, 110
 - GetTriangles, 111
 - GetVertices, 111
 - MeshIndexPair, 110
 - PoseMesh, 110
 - StringVector, 110
 - TriangleEdgeVector, 110
 - TriangleVector, 110
 - VertexVector, 110
- dtAnim::PoseMesh::TargetTriangle, 135
 - mAzimuth, 135
 - mElevation, 135
 - mIsInside, 135
 - mTriangleID, 135
- dtAnim::PoseMesh::Triangle, 136
 - mIndices, 136
 - mVertices, 136
 - Triangle, 136
- dtAnim::PoseMesh::TriangleEdge, 137
 - mEdge, 137
 - mTriangleID, 137
 - TriangleEdge, 137
- dtAnim::PoseMesh::Vertex, 139
 - mAnimID, 139
 - mData, 139
 - mDebugData, 139
 - mDebugPrecision, 139
 - mDebugRotation, 139
 - Vertex, 139
- dtAnim::PoseMeshData, 112
 - mAnimations, 112
 - mEffectorForward, 112
 - mEffectorName, 112
 - mName, 112
 - mRootForward, 112
 - mRootName, 112
- dtAnim::PoseMeshDatabase, 113
 - ~PoseMeshDatabase, 113
 - GetMeshes, 113
 - GetPoseMeshByName, 113
 - LoadFromFile, 113
 - PoseMeshDatabase, 113
 - PoseMeshList, 113
- dtAnim::PoseMeshFileHandler, 114
 - ~PoseMeshFileHandler, 115
 - ANIMATION_NODE, 115
 - characters, 115
 - DEFAULT_VALUE, 115
 - EFFECTOR_ATTRIBUTE, 115
 - EFFECTOR_FORWARD_ATTRIBUTE, 115
 - endDocument, 115
 - endElement, 115
 - endPrefixMapping, 115
 - GetData, 115
 - ignorableWhitespace, 115
 - NAME_ATTRIBUTE, 115
 - NODE_ANIMATION, 114
 - NODE_POSEMESH, 114
 - NODE_TRIANGLE, 114
 - NODE_UNKNOWN, 114
 - NodeStack, 114
 - POSE_NODE, 115
 - PoseMeshDataVector, 114
 - PoseMeshFileHandler, 115
 - PoseNode, 114
 - processingInstruction, 115
 - ROOT_ATTRIBUTE, 115
 - ROOT_FORWARD_ATTRIBUTE, 115
 - setDocumentLocator, 115
 - skippedEntity, 115
 - startDocument, 115
 - startElement, 115
 - startPrefixMapping, 115
 - TRIANGLE_NODE, 115
- dtAnim::PoseMeshLoader, 116
 - ~PoseMeshLoader, 116
 - Load, 116
 - MeshDataContainer, 116
 - PoseMeshLoader, 116
- dtAnim::PoseMeshUtility, 117
 - ~PoseMeshUtility, 117

- BaseReferencePose, 117
 - BlendPoses, 117
 - ClearPoses, 117
 - GetBaseReferenceBlend, 117
 - PoseMeshUtility, 117
 - SetBaseReferencePoses, 118
 - dtAnim::PoseMeshUtility::BaseReferenceBlend, 56
 - blendAlpha, 56
 - endAnimID, 56
 - endDirection, 56
 - startAnimID, 56
 - startDirection, 56
 - dtAnim::RecalcFunctor, 121
 - GetEnd, 121
 - operator(), 121
 - RecalcFunctor, 121
 - dtAnim::SequenceMixer, 123
 - ~SequenceMixer, 124
 - AnimationTable, 124
 - ClearActiveAnimations, 124
 - ClearAnimation, 124
 - ClearRegisteredAnimations, 124
 - ForceRecalculate, 124
 - GetActiveAnimation, 124
 - GetRegisteredAnimation, 124
 - GetRegisteredAnimations, 124
 - IsAnimationPlaying, 125
 - PlayAnimation, 125
 - RegisterAnimation, 125
 - RemoveRegisteredAnimation, 125
 - SequenceMixer, 124
 - TableKey, 124
 - Update, 125
 - dtAnim::SkeletalDrawable, 126
 - ~SkeletalDrawable, 126
 - clone, 126
 - cloneType, 126
 - drawImplementation, 126
 - SkeletalDrawable, 126
 - dtAnim::SkeletalDrawable::IPrimitiveRenderObject::RenderPrimitive, 122
 - mModelWrapper, 122
 - operator(), 122
 - RenderPrimitive, 122
 - dtAnim::SkeletonDriver, 127
 - ~SkeletonDriver, 127
 - SetWrapper, 127
 - SkeletonDriver, 127
 - Update, 127
 - dtAnim::SpringDriver, 128
 - ~SpringDriver, 128
 - SetWrapper, 128
 - SpringDriver, 128
 - Update, 128
 - dtAnim::SubmeshComputeBound, 129
 - computeBound, 129
 - mBoundingBox, 129
 - SubmeshComputeBound, 129
 - dtAnim::SubmeshCullCallback, 130
 - cull, 130
 - SubmeshCullCallback, 130
 - dtAnim::SubmeshDirtyCallback, 131
 - update, 131
 - dtAnim::SubmeshDrawable, 132
 - ~SubmeshDrawable, 132
 - accept, 132
 - clone, 132
 - cloneType, 133
 - drawImplementation, 133
 - GetCurrentLOD, 133
 - GetLODOptions, 133
 - GetMeshID, 133
 - GetModelWrapper, 133
 - GetSubmeshID, 133
 - InitVertexBuffers, 133
 - LOD_COUNT, 133
 - SetBoundingBox, 133
 - SetCurrentLOD, 133
 - SubmeshDrawable, 132
 - supports, 133
 - dtAnim::SubmeshUserData, 134
 - BaseClass, 134
 - className, 134
 - clone, 134
 - cloneType, 134
 - isSameKindAs, 134
 - libraryName, 134
 - mLOD, 134
 - dtAnim::UpdateCallback, 138
 - ~UpdateCallback, 138
 - operator(), 138
 - UpdateCallback, 138
 - dtCore, 14
 - dtDAL, 15
 - dtGame, 16
 - dtUtil, 17
- E -**
- EdgeLineMap
 - dtAnim::PoseMesh, 110
 - EFFECTOR_ATTRIBUTE
 - dtAnim::PoseMeshFileHandler, 115
 - EFFECTOR_FORWARD_ATTRIBUTE
 - dtAnim::PoseMeshFileHandler, 115
 - endAnimID
 - dtAnim::PoseMeshUtility::BaseReferenceBlend, 56
 - endDirection
 - dtAnim::PoseMeshUtility::BaseReferenceBlend, 56
 - endDocument
 - dtAnim::CharacterFileHandler, 82
 - dtAnim::PoseMeshFileHandler, 115
 - endElement
 - dtAnim::CharacterFileHandler, 82
 - dtAnim::PoseMeshFileHandler, 115
 - endPrefixMapping
 - dtAnim::CharacterFileHandler, 82
 - dtAnim::PoseMeshFileHandler, 115
 - EndRenderingQuery
 - dtAnim::Cal3DModelWrapper, 74
 - ExecuteAction
 - dtAnim::Cal3DModelWrapper, 74
 - export.h, 182
 - DT_ANIM_EXPORT, 182
- F -**
- FADE_IN_ELEMENT
 - dtAnim::CharacterFileHandler, 82
 - FADE_OUT_ELEMENT
 - dtAnim::CharacterFileHandler, 82
 - FILENAME_ELEMENT
 - dtAnim::CharacterFileHandler, 82

Find
 dtAnim::Cal3DDatabase, 61
 FindPoseTriangleID
 dtAnim::PoseMesh, 110
 findWithCoreModel
 dtAnim::findWithCoreModel, 94
 findWithFilename
 dtAnim::findWithFilename, 95
 FindWithFunctor
 dtAnim, 12
 ForceFadeOut
 dtAnim::Animatable, 21
 dtAnim::AnimationChannel, 26
 dtAnim::AnimationSequence, 41
 ForceRecalculate
 dtAnim::SequenceMixer, 124
- G -
 GetActiveAnimation
 dtAnim::SequenceMixer, 124
 GetActorProperties
 dtAnim::AnimationHelper, 38
 GetAmbientColor
 dtAnim::Cal3DModelWrapper, 74
 GetAnimatables
 dtAnim::Cal3DModelData, 69
 GetAnimation
 dtAnim::AnimationChannel, 26
 dtAnim::AnimationSequence, 42
 GetAnimationDriver
 dtAnim::Cal3DAnimator, 58
 GetAnimationHelper
 dtAnim::CharacterWrapper, 86
 GetAnimationTime
 dtAnim::Cal3DModelWrapper, 74
 GetAnimationWrappers
 dtAnim::Cal3DModelData, 69
 GetAnimator
 dtAnim::AnimationHelper, 38
 dtAnim::Cal3DGameActor, 63
 GetAttachmentController
 dtAnim::AnimationHelper, 38
 GetAttachments
 dtAnim::AttachmentController, 53
 GetBarySpaces
 dtAnim::PoseMesh, 110
 GetBaseReferenceBlend
 dtAnim::PoseMeshUtility, 117
 GetBaseWeight
 dtAnim::Animatable, 21
 GetBillBoardIcon
 dtAnim::AnimationGameActorProxy, 35
 dtAnim::Cal3DGameActorProxy, 66
 GetBoneAbsoluteRotation
 dtAnim::Cal3DModelWrapper, 74
 GetBoneAbsoluteRotationForKeyFrame
 dtAnim::Cal3DModelWrapper, 75
 GetBoneAbsoluteTranslation
 dtAnim::Cal3DModelWrapper, 75
 GetBoneRelativeRotation
 dtAnim::Cal3DModelWrapper, 75
 GetBoundingBox
 dtAnim::Cal3DModelWrapper, 75
 GetCal3DWrapper
 dtAnim::CharDrawable, 89
 GetCalModel
 dtAnim::Cal3DModelWrapper, 75
 GetCelestialCoordinates
 dtAnim, 12
 GetCelestialDirection
 dtAnim, 12
 GetChildAnimations
 dtAnim::AnimationSequence, 42
 GetClosestPointOnSegment
 dtAnim, 12
 GetController
 dtAnim::AnimationSequence, 42
 GetCoreAnimationCount
 dtAnim::Cal3DModelWrapper, 75
 GetCoreAnimationDuration
 dtAnim::Cal3DModelWrapper, 75
 GetCoreAnimationIDByName
 dtAnim::Cal3DModelWrapper, 75
 GetCoreAnimationKeyframeCount
 dtAnim::Cal3DModelWrapper, 75
 GetCoreAnimationKeyframeCountForTrack
 dtAnim::Cal3DModelWrapper, 75
 GetCoreAnimationName
 dtAnim::Cal3DModelWrapper, 75
 GetCoreAnimationTrackCount
 dtAnim::Cal3DModelWrapper, 76
 GetCoreBoneChildrenIDs
 dtAnim::Cal3DModelWrapper, 76
 GetCoreBoneID
 dtAnim::Cal3DModelWrapper, 76
 GetCoreBoneNames
 dtAnim::Cal3DModelWrapper, 76
 GetCoreMaterial
 dtAnim::Cal3DModelWrapper, 76
 GetCoreMaterialAmbient
 dtAnim::Cal3DModelWrapper, 76
 GetCoreMaterialCount
 dtAnim::Cal3DModelWrapper, 76
 GetCoreMaterialDiffuse
 dtAnim::Cal3DModelWrapper, 76
 GetCoreMaterialName
 dtAnim::Cal3DModelWrapper, 76
 GetCoreMaterialShininess
 dtAnim::Cal3DModelWrapper, 76
 GetCoreMaterialSpecular
 dtAnim::Cal3DModelWrapper, 76
 GetCoreMeshCount
 dtAnim::Cal3DModelWrapper, 76
 GetCoreMeshName
 dtAnim::Cal3DModelWrapper, 76
 GetCoreModel
 dtAnim::Cal3DModelData, 69
 GetCoreTrackKeyFrameQuat
 dtAnim::Cal3DModelWrapper, 76
 GetCreate
 dtAnim::AnimNodeBuilder, 49
 GetCurrentLOD
 dtAnim::SubmeshDrawable, 133
 GetCurrentWeight
 dtAnim::Animatable, 22
 GetData
 dtAnim::PoseMeshFileHandler, 115
 GetDiffuseColor
 dtAnim::Cal3DModelWrapper, 76
 GetDuration
 dtAnim::AnimationWrapper, 45
 GetEffectorForwardAxis

- dtAnim::PoseMesh, 110
- GetEffectorID
 - dtAnim::PoseMesh, 110
- GetEffectorName
 - dtAnim::PoseMesh, 110
- GetElapsedTime
 - dtAnim::Animatable, 22
- GetEnd
 - dtAnim::RecalcFuncor, 121
- GetEndDistance
 - dtAnim::LODOptions, 102
- GetEndTime
 - dtAnim::Animatable, 22
- GetEyePointActor
 - dtAnim::AnimationComponent, 30
- GetFaceCount
 - dtAnim::Cal3DModelWrapper, 77
- GetFaces
 - dtAnim::Cal3DModelWrapper, 77
- GetFadeIn
 - dtAnim::Animatable, 22
- GetFadeOut
 - dtAnim::Animatable, 22
- GetFilename
 - dtAnim::Cal3DModelData, 69
- GetGroundClamp
 - dtAnim::AnimationHelper, 38
- GetGroundClamper
 - dtAnim::AnimationComponent, 30
- GetHeading
 - dtAnim::CharacterWrapper, 87
- GetHeightAboveGround
 - dtAnim::CharacterWrapper, 87
- GetHelper
 - dtAnim::AnimationGameActor, 34
- GetHelperForProxy
 - dtAnim::AnimationComponent, 30
- GetHotSpots
 - dtAnim::HotSpotDriver, 99
- GetID
 - dtAnim::AnimationWrapper, 45
- GetIndexPairsForTriangle
 - dtAnim::PoseMesh, 110
- GetIndexVBO
 - dtAnim::Cal3DModelData, 69
- GetInstance
 - dtAnim::Cal3DDatabase, 61
- GetLocalOffset
 - dtAnim::CharacterWrapper, 87
- GetLODOptions
 - dtAnim::Cal3DModelData, 69
 - dtAnim::SubmeshDrawable, 133
- GetMapCount
 - dtAnim::Cal3DModelWrapper, 77
- GetMapUserData
 - dtAnim::Cal3DModelWrapper, 77
- GetMaxDuration
 - dtAnim::AnimationChannel, 26
- GetMaxVisibleDistance
 - dtAnim::LODOptions, 102
- GetMeshCount
 - dtAnim::Cal3DModelWrapper, 77
- GetMeshes
 - dtAnim::PoseMeshDatabase, 113
- GetMeshID
 - dtAnim::SubmeshDrawable, 133
- GetModel
 - dtAnim::AnimationChannel, 26
- GetModelData
 - dtAnim::Cal3DDatabase, 61
- GetModelWrapper
 - dtAnim::AnimationHelper, 38
 - dtAnim::SubmeshDrawable, 133
- GetMorphTargetDriver
 - dtAnim::Cal3DAnimator, 58
- GetName
 - dtAnim::Animatable, 22
 - dtAnim::AnimationWrapper, 45
 - dtAnim::PoseMesh, 110
- GetNode
 - dtAnim::AnimationHelper, 38, 39
 - dtAnim::CharDrawable, 89
- GetNodeBuilder
 - dtAnim::Cal3DDatabase, 61
- GetNormals
 - dtAnim::Cal3DModelWrapper, 77
- GetOrCreateCalHardwareModel
 - dtAnim::Cal3DModelWrapper, 77
- GetParent
 - dtAnim::AnimationSequence::AnimationController, 33
- GetParentBoneID
 - dtAnim::Cal3DModelWrapper, 77
- GetPhysiqueDriver
 - dtAnim::Cal3DAnimator, 58
- GetPoseMeshByName
 - dtAnim::PoseMeshDatabase, 113
- GetPoseMeshFilename
 - dtAnim::Cal3DModelData, 69
- GetPostDriver
 - dtAnim::Cal3DAnimator, 58
- GetPreDriver
 - dtAnim::Cal3DAnimator, 58
- GetRegisteredAnimation
 - dtAnim::SequenceMixer, 124
- GetRegisteredAnimations
 - dtAnim::SequenceMixer, 124
- GetRenderMode
 - dtAnim::AnimationGameActorProxy, 35
 - dtAnim::Cal3DGameActor, 63
 - dtAnim::Cal3DGameActorProxy, 66
- GetRootBoneIDs
 - dtAnim::Cal3DModelWrapper, 77
- GetRootForwardAxis
 - dtAnim::PoseMesh, 110
- GetRotationSpeed
 - dtAnim::CharacterWrapper, 87
- GetSequenceMixer
 - dtAnim::AnimationHelper, 39
- GetShaderGroupName
 - dtAnim::Cal3DModelData, 69
- GetShaderMaxBones
 - dtAnim::Cal3DModelData, 69
- GetShaderName
 - dtAnim::Cal3DModelData, 69
- GetShininess
 - dtAnim::Cal3DModelWrapper, 77
- GetSilhouette
 - dtAnim::PoseMesh, 110
- GetSkeletonDriver
 - dtAnim::Cal3DAnimator, 58
- GetSpecularColor
 - dtAnim::Cal3DModelWrapper, 77

- GetSpeed
 - dtAnim::Animatable, 22
 - dtAnim::AnimationWrapper, 45
 - dtAnim::CharacterWrapper, 87
- GetSpringDriver
 - dtAnim::Cal3DAnimator, 58
- GetStartDelay
 - dtAnim::Animatable, 22
- GetStartDistance
 - dtAnim::LODOptions, 102
- GetStartTime
 - dtAnim::Animatable, 22
- GetSubmeshCount
 - dtAnim::Cal3DModelWrapper, 77
- GetSubmeshID
 - dtAnim::SubmeshDrawable, 133
- GetTargetTriangleData
 - dtAnim::PoseMesh, 110
- GetTerrainActor
 - dtAnim::AnimationComponent, 30
- GetTextureCoords
 - dtAnim::Cal3DModelWrapper, 77
- GetTriangles
 - dtAnim::PoseMesh, 111
- GetVertexCount
 - dtAnim::Cal3DModelWrapper, 77
- GetVertexVBO
 - dtAnim::Cal3DModelData, 69
- GetVertices
 - dtAnim::Cal3DModelWrapper, 77
 - dtAnim::PoseMesh, 111
- GetWrapper
 - dtAnim::Cal3DAnimator, 58
- GroundClamp
 - dtAnim::AnimationComponent, 30
- H -**
- hardwaresubmesh.cpp, 183
 - BUFFER_OFFSET, 183
- hardwaresubmesh.h, 184
- HardwareSubmeshCallback
 - dtAnim::HardwareSubmeshCallback, 96
- HardwareSubmeshComputeBound
 - dtAnim::HardwareSubmeshComputeBound, 97
- HardwareSubmeshDrawable
 - dtAnim::HardwareSubmeshDrawable, 98
- HasAnimation
 - dtAnim::Cal3DModelWrapper, 77
- HasBone
 - dtAnim::Cal3DModelWrapper, 77
- HasTrackForBone
 - dtAnim::Cal3DModelWrapper, 77
- HideMesh
 - dtAnim::Cal3DModelWrapper, 77
- HotSpotContainer
 - dtAnim::HotSpotDriver, 99
- HotSpotDriver
 - dtAnim::HotSpotDriver, 99
- hotspotdriver.cpp, 185
- hotspotdriver.h, 186
- I -**
- ical3ddriver.h, 187
- ignorableWhitespace
 - dtAnim::CharacterFileHandler, 82
- dtAnim::PoseMeshFileHandler, 115
- inc/ Directory Reference, 7
- inc/dtAnim/ Directory Reference, 5
- INDEX_OFFSET
 - submesh.cpp, 213
- InitVertexBuffers
 - dtAnim::SubmeshDrawable, 133
- IS_ACTION_ELEMENT
 - dtAnim::CharacterFileHandler, 82
- IS_LOOPING_ELEMENT
 - dtAnim::CharacterFileHandler, 82
- IsAction
 - dtAnim::AnimationChannel, 27
- IsActive
 - dtAnim::Animatable, 22
- IsActor
 - dtAnim::IsActor, 101
- IsAnimationPlaying
 - dtAnim::AnimationSequence, 42
 - dtAnim::CharacterWrapper, 87
 - dtAnim::SequenceMixer, 125
- IsLooping
 - dtAnim::AnimationChannel, 27
- IsMeshVisible
 - dtAnim::Cal3DModelWrapper, 77
- IsPointBetweenVectors
 - dtAnim, 12
- IsRegisteredActor
 - dtAnim::AnimationComponent, 30
- isSameKindAs
 - dtAnim::SubmeshUserData, 134
- L -**
- libraryName
 - dtAnim::SubmeshUserData, 134
- Load
 - dtAnim::Cal3DDatabase, 61
 - dtAnim::Cal3DLoader, 67
 - dtAnim::PoseMeshLoader, 116
- LoadFromFile
 - dtAnim::PoseMeshDatabase, 113
- LoadModel
 - dtAnim::AnimationHelper, 39
- LoadShaders
 - dtAnim::AnimNodeBuilder, 49
- LOD_COUNT
 - dtAnim::SubmeshDrawable, 133
- LOD_ELEMENT
 - dtAnim::CharacterFileHandler, 82
- LOD_END_DISTANCE_ELEMENT
 - dtAnim::CharacterFileHandler, 82
- LOD_START_DISTANCE_ELEMENT
 - dtAnim::CharacterFileHandler, 82
- LODOptions
 - dtAnim::LODOptions, 102
- M -**
- mainpage.h, 188
- MakeAnimationGroup
 - dtAnim::Cal3DGameActor, 63
- mAnimationChannels
 - dtAnim::CharacterFileHandler, 82
- mAnimationName
 - dtAnim::CharacterFileHandler::AnimationChannelStruct, 28

- mAnimations
 - dtAnim::CharacterFileHandler, 82
 - dtAnim::PoseMeshData, 112
- mAnimationSequences
 - dtAnim::CharacterFileHandler, 82
- mAnimator
 - dtAnim::Cal3DGameActor, 64
 - dtAnim::CharDrawable, 90
- mAnimID
 - dtAnim::PoseMesh::Vertex, 139
- MapCelestialToScreen
 - dtAnim, 12
- MATERIAL_ELEMENT
 - dtAnim::CharacterFileHandler, 83
- MAX_DURATION_ELEMENT
 - dtAnim::CharacterFileHandler, 83
- MAX_VISIBLE_DISTANCE_ELEMENT
 - dtAnim::CharacterFileHandler, 83
- mAzimuth
 - dtAnim::PoseMesh::TargetTriangle, 135
- mBaseWeight
 - dtAnim::CharacterFileHandler::AnimatableStruct, 24
- mBoundingBox
 - dtAnim::SubmeshComputeBound, 129
- mChildNames
 - dtAnim::CharacterFileHandler::AnimationSequenceStruct, 43
- mCreatedNode
 - dtAnim::CreateGeometryDrawCallback, 92
- mData
 - dtAnim::PoseMesh::Vertex, 139
- mDebugData
 - dtAnim::PoseMesh::Vertex, 139
- mDebugPrecision
 - dtAnim::PoseMesh::Vertex, 139
- mDebugRotation
 - dtAnim::PoseMesh::Vertex, 139
- mDefaultBox
 - dtAnim::HardwareSubmeshComputeBound, 97
- mEdge
 - dtAnim::PoseMesh::TriangleEdge, 137
- mEffectorForward
 - dtAnim::PoseMeshData, 112
- mEffectorName
 - dtAnim::PoseMeshData, 112
- mElevation
 - dtAnim::PoseMesh::TargetTriangle, 135
- MESH_ELEMENT
 - dtAnim::CharacterFileHandler, 83
- MeshDataContainer
 - dtAnim::PoseMeshLoader, 116
- MeshIndexPair
 - dtAnim::PoseMesh, 110
- mFadeln
 - dtAnim::CharacterFileHandler::AnimatableStruct, 24
- mFadeOut
 - dtAnim::CharacterFileHandler::AnimatableStruct, 24
- mFileLoader
 - dtAnim::Cal3DDatabase, 61
- mFileName
 - dtAnim::CharacterFileHandler::AnimationStruct, 44
 - dtAnim::CharacterFileHandler::MaterialStruct, 103
 - dtAnim::CharacterFileHandler::MeshStruct, 104
 - dtAnim::CharacterFileHandler::MorphAnimationStruct, 105
- mFilename
 - dtAnim::findWithFilename, 95
- mFoundLOOptions
 - dtAnim::CharacterFileHandler, 83
- mFoundScale
 - dtAnim::CharacterFileHandler, 83
- mHelper
 - dtAnim::AnimationGameActor, 34
- mIndices
 - dtAnim::PoseMesh::Triangle, 136
- mInstance
 - dtAnim::Cal3DDatabase, 61
- mIsAction
 - dtAnim::CharacterFileHandler::AnimationChannelStruct, 28
- mIsInside
 - dtAnim::PoseMesh::TargetTriangle, 135
- mIsLooping
 - dtAnim::CharacterFileHandler::AnimationChannelStruct, 28
- mLastMeshCount
 - dtAnim::CharDrawable, 90
- mLOD
 - dtAnim::SubmeshUserData, 134
- mLODEndDistance
 - dtAnim::CharacterFileHandler, 83
- mLODMaxVisibleDistance
 - dtAnim::CharacterFileHandler, 83
- mLODStartDistance
 - dtAnim::CharacterFileHandler, 83
- mMaterials
 - dtAnim::CharacterFileHandler, 83
- mMaxDuration
 - dtAnim::CharacterFileHandler::AnimationChannelStruct, 28
- mMeshes
 - dtAnim::CharacterFileHandler, 83
- mModel
 - dtAnim::findWithCoreModel, 94
- mModelData
 - dtAnim::Cal3DDatabase, 61
- mModelGeode
 - dtAnim::Cal3DGameActor, 64
- mModelLoader
 - dtAnim::Cal3DGameActor, 64
- mModelWrapper
 - dtAnim::SkeletalDrawable::IPrimitiveRenderObject::RenderPrimitive, 122
- mMorphAnimations
 - dtAnim::CharacterFileHandler, 83
- mName
 - dtAnim::CharacterFileHandler, 83
 - dtAnim::CharacterFileHandler::AnimatableStruct, 24
 - dtAnim::CharacterFileHandler::AnimationStruct, 44
 - dtAnim::CharacterFileHandler::MaterialStruct, 103
 - dtAnim::CharacterFileHandler::MeshStruct, 104
 - dtAnim::CharacterFileHandler::MorphAnimationStruct, 105
 - dtAnim::PoseMeshData, 112
 - PredicatePoseMeshName, 119
- mNode
 - dtAnim::CharDrawable, 90
- mNodeBuilder
 - dtAnim::Cal3DDatabase, 61
- ModelDataArray
 - dtAnim::Cal3DDatabase, 61
- MORPH_ANIMATION_ELEMENT

- dtAnim::CharacterFileHandler, 83
- MorphDriver
 - dtAnim::MorphDriver, 106
- morphdriver.cpp, 189
- morphdriver.h, 190
- mPoseMeshFilename
 - dtAnim::CharacterFileHandler, 83
- mRenderModeBits
 - dtAnim::Cal3DGameActor, 64
- mRootForward
 - dtAnim::PoseMeshData, 112
- mRootName
 - dtAnim::PoseMeshData, 112
- mScale
 - dtAnim::CharacterFileHandler, 83
- mShaderGroup
 - dtAnim::CharacterFileHandler, 83
- mShaderMaxBones
 - dtAnim::CharacterFileHandler, 83
- mShaderName
 - dtAnim::CharacterFileHandler, 83
- mSkeletalGeode
 - dtAnim::Cal3DGameActor, 64
- mSkeletonFilename
 - dtAnim::CharacterFileHandler, 83
- mSpeed
 - dtAnim::CharacterFileHandler::AnimatableStruct, 24
- mStartDelay
 - dtAnim::CharacterFileHandler::AnimatableStruct, 24
- mTriangleID
 - dtAnim::PoseMesh::TargetTriangle, 135
 - dtAnim::PoseMesh::TriangleEdge, 137
- mVertices
 - dtAnim::PoseMesh::Triangle, 136

- N -

- NAME_ATTRIBUTE
 - dtAnim::PoseMeshFileHandler, 115
- NAME_ELEMENT
 - dtAnim::CharacterFileHandler, 83
- NODE_ANIMATION
 - dtAnim::PoseMeshFileHandler, 114
- NODE_POSEMESH
 - dtAnim::PoseMeshFileHandler, 114
- NODE_TRIANGLE
 - dtAnim::PoseMeshFileHandler, 114
- NODE_UNKNOWN
 - dtAnim::PoseMeshFileHandler, 114
- NodeStack
 - dtAnim::PoseMeshFileHandler, 114
- NULL_BONE
 - dtAnim::Cal3DModelWrapper, 78

- O -

- OnEnteredWorld
 - dtAnim::Cal3DGameActor, 63
- OnMessage
 - dtAnim::CharDrawable, 89
- OnTickLocal
 - dtAnim::Cal3DGameActor, 63
- operator()
 - DeletePointer, 93
 - dtAnim::AnimSeqForceFade, 51
 - dtAnim::AnimSequenceUpdater, 52
 - dtAnim::AttachmentMover, 55

- dtAnim::CloneFunctor, 91
- dtAnim::findWithCoreModel, 94
- dtAnim::findWithFilename, 95
- dtAnim::IsActor, 101
- dtAnim::RecalcFunctor, 121
- dtAnim::SkeletalDrawable::IPrimitiveRenderObject::RenderPrimitive
 - 122
- dtAnim::UpdateCallback, 138
- PoseBuilderFunctor, 108
- PredicatePoseMeshName, 119
- operator=
 - dtAnim::Animatable, 22
 - dtAnim::AnimationChannel, 27
 - dtAnim::AnimationSequence, 42
 - dtAnim::AnimationSequence::AnimationController, 33
 - dtAnim::AnimNodeBuilder, 49
 - dtAnim::AttachmentMover, 55
 - dtAnim::Cal3DModelData, 69

- P -

- PhysiqueDriver
 - dtAnim::PhysiqueDriver, 107
- physiquedriver.cpp, 191
- physiquedriver.h, 192
- PlayAnimation
 - dtAnim::AnimationHelper, 39
 - dtAnim::CharacterWrapper, 87
 - dtAnim::SequenceMixer, 125
- POSE_NODE
 - dtAnim::PoseMeshFileHandler, 115
- PoseBuilderFunctor, 108
 - operator(), 108
 - PoseBuilderFunctor, 108
- posemath.cpp, 193
- posemath.h, 194
- PoseMesh
 - dtAnim::PoseMesh, 110
- posemesh.cpp, 195
- posemesh.h, 196
 - TRIANGLE_NOT_FOUND, 196
- POSEMESH_ELEMENT
 - dtAnim::CharacterFileHandler, 84
- PoseMeshDatabase
 - dtAnim::PoseMeshDatabase, 113
- posemeshdatabase.cpp, 197
- posemeshdatabase.h, 198
- PoseMeshDataVector
 - dtAnim::PoseMeshFileHandler, 114
- PoseMeshFileHandler
 - dtAnim::PoseMeshFileHandler, 115
- PoseMeshList
 - dtAnim::PoseMeshDatabase, 113
- PoseMeshLoader
 - dtAnim::PoseMeshLoader, 116
- posemeshloader.cpp, 199
- posemeshloader.h, 200
- PoseMeshUtility
 - dtAnim::PoseMeshUtility, 117
- posemeshutility.cpp, 201
 - BaseReferencePredicate, 201
- posemeshutility.h, 202
- posemeshxml.cpp, 203
- posemeshxml.h, 204
- PoseNode
 - dtAnim::PoseMeshFileHandler, 114
- PredicatePoseMeshName, 119

- mName, [119](#)
 - operator(), [119](#)
 - PredicatePoseMeshName, [119](#)
- processingInstruction
 - dtAnim::CharacterFileHandler, [82](#)
 - dtAnim::PoseMeshFileHandler, [115](#)
- ProcessMessage
 - dtAnim::AnimationComponent, [30](#)
- PROPERTY_SKELETAL_MESH
 - dtAnim::AnimationHelper, [39](#)
- Prune
 - dtAnim::Animatable, [22](#)
 - dtAnim::AnimationChannel, [27](#)
 - dtAnim::AnimationSequence, [42](#)
- PurgeAllCaches
 - dtAnim::Cal3DLoader, [67](#)
- PurgeLoaderCaches
 - dtAnim::Cal3DDatabase, [61](#)
- R -**
- RebuildSubmeshes
 - dtAnim::CharDrawable, [89](#)
- RecalcFunctor
 - dtAnim::RecalcFunctor, [121](#)
- Recalculate
 - dtAnim::Animatable, [22](#)
 - dtAnim::AnimationChannel, [27](#)
 - dtAnim::AnimationSequence, [42](#)
 - dtAnim::AnimationSequence::AnimationController, [33](#)
- RegisterActor
 - dtAnim::AnimationComponent, [30](#)
- RegisterActorTypes
 - dtAnim::AnimActorRegistry, [19](#)
- RegisterAnimation
 - dtAnim::SequenceMixer, [125](#)
- Remove
 - dtAnim::Cal3DModelData, [69](#)
- RemoveAction
 - dtAnim::Cal3DModelWrapper, [77](#)
- RemoveAttachment
 - dtAnim::AttachmentController, [53](#)
- RemoveHotSpot
 - dtAnim::HotSpotDriver, [99](#)
- RemoveRegisteredAnimation
 - dtAnim::SequenceMixer, [125](#)
- RENDER_MODE_BONES
 - dtAnim::Cal3DGameActor, [63](#)
- RENDER_MODE_NONE
 - dtAnim::Cal3DGameActor, [63](#)
- RENDER_MODE_SKIN
 - dtAnim::Cal3DGameActor, [63](#)
- RENDER_MODE
 - dtAnim::Cal3DGameActor::PropertyNames, [120](#)
- RENDER_MODE_LABEL
 - dtAnim::Cal3DGameActor::PropertyNames, [120](#)
- RenderModeBitContainer
 - dtAnim::Cal3DGameActor, [63](#)
- RenderModeBits
 - dtAnim::Cal3DGameActor, [63](#)
- RenderPrimitive
 - dtAnim::SkeletalDrawable::IPrimitiveRenderObject::RenderPrimitive, [122](#)
- ROOT_ATTRIBUTE
 - dtAnim::PoseMeshFileHandler, [115](#)
- ROOT_FORWARD_ATTRIBUTE
 - dtAnim::PoseMeshFileHandler, [115](#)
- RotateToHeading
 - dtAnim::CharacterWrapper, [87](#)
- RotateToPoint
 - dtAnim::CharacterWrapper, [87](#)
- S -**
- SCALE_ELEMENT
 - dtAnim::CharacterFileHandler, [84](#)
- SCALE_FACTOR_ELEMENT
 - dtAnim::CharacterFileHandler, [84](#)
- SelectMeshSubmesh
 - dtAnim::Cal3DModelWrapper, [77](#)
- SEQUENCE_ELEMENT
 - dtAnim::CharacterFileHandler, [84](#)
- SequenceMixer
 - dtAnim::SequenceMixer, [124](#)
- sequencemixer.cpp, [205](#)
- sequencemixer.h, [206](#)
- SetAction
 - dtAnim::AnimationChannel, [27](#)
- SetActive
 - dtAnim::Animatable, [22](#)
- SetAnimation
 - dtAnim::AnimationChannel, [27](#)
- SetAnimationDriver
 - dtAnim::Cal3DAnimator, [58](#)
- SetAnimationTime
 - dtAnim::Cal3DModelWrapper, [77](#)
- SetAttachmentController
 - dtAnim::AnimationHelper, [39](#)
- SetBaseReferencePoses
 - dtAnim::PoseMeshUtility, [118](#)
- SetBaseWeight
 - dtAnim::Animatable, [22](#)
- SetBoundingBox
 - dtAnim::HardwareSubmeshDrawable, [98](#)
 - dtAnim::SubmeshDrawable, [133](#)
- SetCal3DWrapper
 - dtAnim::CharDrawable, [89](#)
- SetCalModel
 - dtAnim::Cal3DModelWrapper, [77](#)
- SetComputeSpeed
 - dtAnim::AnimationSequence::AnimationController, [33](#)
- SetComputeWeight
 - dtAnim::AnimationSequence::AnimationController, [33](#)
- SetController
 - dtAnim::AnimationSequence, [42](#)
- SetCreate
 - dtAnim::AnimNodeBuilder, [49](#)
- SetCurrentLOD
 - dtAnim::SubmeshDrawable, [133](#)
- SetCurrentWeight
 - dtAnim::Animatable, [22](#)
- setDocumentLocator
 - dtAnim::CharacterFileHandler, [82](#)
 - dtAnim::PoseMeshFileHandler, [115](#)
- SetDuration
 - dtAnim::AnimationWrapper, [45](#)
- SetElapsedTime
 - dtAnim::Animatable, [22](#)
- SetEndCallback
 - dtAnim::Animatable, [22](#)
- SetEndDistance
 - dtAnim::LODOptions, [102](#)
- SetEndTime
 - dtAnim::Animatable, [22](#)

- SetEyePointActor
 - dtAnim::AnimationComponent, 31
- SetFadeIn
 - dtAnim::Animatable, 23
- SetFadeOut
 - dtAnim::Animatable, 23
- SetGroundClamp
 - dtAnim::AnimationHelper, 39
 - dtAnim::CharacterWrapper, 87, 88
- SetGroundClamper
 - dtAnim::AnimationComponent, 31
- SetHeading
 - dtAnim::CharacterWrapper, 88
- SetHeightAboveGround
 - dtAnim::CharacterWrapper, 88
- SetIndexVBO
 - dtAnim::Cal3DModelData, 69
- SetLocalOffset
 - dtAnim::CharacterWrapper, 88
- SetLODLevel
 - dtAnim::Cal3DModelWrapper, 78
- SetLooping
 - dtAnim::AnimationChannel, 27
- SetMaterialSet
 - dtAnim::Cal3DModelWrapper, 78
- SetMaxDuration
 - dtAnim::AnimationChannel, 27
- SetMaxVisibleDistance
 - dtAnim::LODOptions, 102
- SetModel
 - dtAnim::AnimationChannel, 27
 - dtAnim::AnimationGameActor, 34
 - dtAnim::Cal3DGameActor, 63
- SetMorphTargetDriver
 - dtAnim::Cal3DAnimator, 58
- SetName
 - dtAnim::Animatable, 23
 - dtAnim::AnimationWrapper, 46
- SetParent
 - dtAnim::AnimationSequence::AnimationController, 33
- SetPhysiqueDriver
 - dtAnim::Cal3DAnimator, 58
- SetPoseMeshFilename
 - dtAnim::Cal3DModelData, 69
- SetPostDriver
 - dtAnim::Cal3DAnimator, 58
- SetPreDriver
 - dtAnim::Cal3DAnimator, 58
- SetPrune
 - dtAnim::Animatable, 23
- SetRenderMode
 - dtAnim::Cal3DGameActor, 63
- SetRotationSpeed
 - dtAnim::CharacterWrapper, 88
- SetShaderGroupName
 - dtAnim::Cal3DModelData, 70
- SetShaderMaxBones
 - dtAnim::Cal3DModelData, 70
- SetShaderName
 - dtAnim::Cal3DModelData, 70
- SetSkeletonDriver
 - dtAnim::Cal3DAnimator, 58
- SetSpeed
 - dtAnim::Animatable, 23
 - dtAnim::AnimationWrapper, 46
 - dtAnim::CharacterWrapper, 88
- SetSpringDriver
 - dtAnim::Cal3DAnimator, 58
- SetStartDelay
 - dtAnim::Animatable, 23
- SetStartDistance
 - dtAnim::LODOptions, 102
- SetStartTime
 - dtAnim::Animatable, 23
- SetTerrainActor
 - dtAnim::AnimationComponent, 31
- SetVertexVBO
 - dtAnim::Cal3DModelData, 70
- SetWrapper
 - dtAnim::AnimDriver, 47
 - dtAnim::Cal3DAnimator, 58
 - dtAnim::HotSpotDriver, 99
 - dtAnim::ICal3DDriver, 100
 - dtAnim::MorphDriver, 106
 - dtAnim::PhysiqueDriver, 107
 - dtAnim::SkeletonDriver, 127
 - dtAnim::SpringDriver, 128
- SHADER_GROUP_ELEMENT
 - dtAnim::CharacterFileHandler, 84
- SHADER_MAX_BONES_ELEMENT
 - dtAnim::CharacterFileHandler, 84
- SHADER_NAME_ELEMENT
 - dtAnim::CharacterFileHandler, 84
- ShouldPrune
 - dtAnim::Animatable, 23
- ShowMesh
 - dtAnim::Cal3DModelWrapper, 78
- SkeletalDrawable
 - dtAnim::SkeletalDrawable, 126
- skeletaldrawable.cpp, 207
- skeletaldrawable.h, 208
- SKELETON_ELEMENT
 - dtAnim::CharacterFileHandler, 84
- SkeletonDriver
 - dtAnim::SkeletonDriver, 127
- skeletondriver.cpp, 209
- skeletondriver.h, 210
- SKINNING_SHADER_ELEMENT
 - dtAnim::CharacterFileHandler, 84
- skippedEntity
 - dtAnim::CharacterFileHandler, 82
 - dtAnim::PoseMeshFileHandler, 115
- SPEED_ELEMENT
 - dtAnim::CharacterFileHandler, 84
- SpringDriver
 - dtAnim::SpringDriver, 128
- springdriver.cpp, 211
- springdriver.h, 212
- src/ Directory Reference, 8
- src/dtAnim/ Directory Reference, 6
- START_DELAY_ELEMENT
 - dtAnim::CharacterFileHandler, 84
- startAnimID
 - dtAnim::PoseMeshUtility::BaseReferenceBlend, 56
- startDirection
 - dtAnim::PoseMeshUtility::BaseReferenceBlend, 56
- startDocument
 - dtAnim::CharacterFileHandler, 82
 - dtAnim::PoseMeshFileHandler, 115
- startElement
 - dtAnim::CharacterFileHandler, 82
 - dtAnim::PoseMeshFileHandler, 115

startPrefixMapping
 dtAnim::CharacterFileHandler, 82
 dtAnim::PoseMeshFileHandler, 115

StringVector
 dtAnim, 12
 dtAnim::PoseMesh, 110

submesh.cpp, 213
 BUFFER_OFFSET, 213
 INDEX_OFFSET, 213

submesh.h, 214

SubmeshComputeBound
 dtAnim::SubmeshComputeBound, 129

SubmeshCullCallback
 dtAnim::SubmeshCullCallback, 130

SubmeshDrawable
 dtAnim::SubmeshDrawable, 132

supports
 dtAnim::SubmeshDrawable, 133

SupportsHardware
 dtAnim::AnimNodeBuilder, 49

SupportsSoftware
 dtAnim::AnimNodeBuilder, 49

- T -

TableKey
 dtAnim::SequenceMixer, 124

TickLocal
 dtAnim::AnimationComponent, 31

Triangle
 dtAnim::PoseMesh::Triangle, 136

TRIANGLE_NODE
 dtAnim::PoseMeshFileHandler, 115

TRIANGLE_NOT_FOUND
 posemesh.h, 196

TriangleEdge
 dtAnim::PoseMesh::TriangleEdge, 137

TriangleEdgeVector
 dtAnim::PoseMesh, 110

TriangleVector
 dtAnim::PoseMesh, 110

TruncateDatabase
 dtAnim::Cal3DDatabase, 61

- U -

UnregisterActor
 dtAnim::AnimationComponent, 31

Update
 dtAnim::Animatable, 23
 dtAnim::AnimationChannel, 27
 dtAnim::AnimationHelper, 39
 dtAnim::AnimationSequence, 42
 dtAnim::AnimationSequence::AnimationController, 33
 dtAnim::AnimDriver, 47
 dtAnim::AttachmentController, 53
 dtAnim::Cal3DAnimator, 58
 dtAnim::Cal3DModelWrapper, 78
 dtAnim::CharacterWrapper, 88
 dtAnim::HotSpotDriver, 99
 dtAnim::ICal3DDriver, 100
 dtAnim::MorphDriver, 106
 dtAnim::PhysiqueDriver, 107
 dtAnim::SequenceMixer, 125
 dtAnim::SkeletonDriver, 127
 dtAnim::SpringDriver, 128

update
 dtAnim::HardwareSubmeshCallback, 96
 dtAnim::SubmeshDirtyCallback, 131

UpdateAnimation
 dtAnim::Cal3DModelWrapper, 78

UpdateCallback
 dtAnim::UpdateCallback, 138

UpdateMorphTargetMixer
 dtAnim::Cal3DModelWrapper, 78

UpdatePhysique
 dtAnim::Cal3DModelWrapper, 78

UpdateSkeleton
 dtAnim::Cal3DModelWrapper, 78

UpdateSpringSystem
 dtAnim::Cal3DModelWrapper, 78

- V -

Vertex
 dtAnim::PoseMesh::Vertex, 139

VertexVector
 dtAnim::PoseMesh, 110

- X -

XERCES_CPP_NAMESPACE_USE
 characterfilehandler.cpp, 175