



Delta3D Version 2.4.0

dtTerrain::

Reference Manual

Contents

1	Main Page	1
2	Directory Documentation	3
2.1	inc/dtTerrain/ Directory Reference	3
2.2	src/dtTerrain/ Directory Reference	4
2.3	inc/ Directory Reference	5
2.4	src/ Directory Reference	6
3	Namespace Documentation	7
3.1	dtTerrain Namespace Reference	7
3.1.1	Detailed Description	9
3.1.2	Function Documentation	10
3.1.2.1	IMPLEMENT_ENUM	10
3.1.2.2	IMPLEMENT_ENUM	10
3.1.2.3	IMPLEMENT_ENUM	10
3.1.2.4	IMPLEMENT_ENUM	10
3.1.2.5	IMPLEMENT_ENUM	10
3.1.2.6	IMPLEMENT_ENUM	10
3.1.2.7	IMPLEMENT_ENUM	10
3.1.2.8	IMPLEMENT_ENUM	10
3.1.2.9	IMPLEMENT_ENUM	10
3.1.2.10	IMPLEMENT_ENUM	10
3.1.2.11	IMPLEMENT_ENUM	10
3.1.2.12	IMPLEMENT_ENUM	10
3.1.2.13	IMPLEMENT_ENUM	10
3.1.2.14	IMPLEMENT_ENUM	10
3.1.2.15	IMPLEMENT_ENUM	10
3.1.2.16	IMPLEMENT_MANAGEMENT_LAYER	10
3.1.2.17	SimpleLineOfSight	10
3.1.2.18	vec3dToString	10
3.1.2.19	vec3ToString	10
3.1.3	Variable Documentation	10
3.1.3.1	BOTTOM	10
3.1.3.2	COLOR_MAP_IMAGE_NAME	10
3.1.3.3	GEOTIFF_IMAGE_NAME	10
3.1.3.4	I	10
3.1.3.5	SIDE	10

4	Class Documentation	11
4.1	ColorMapDecorator Class Reference	11
4.1.1	Detailed Description	11
4.1.2	Constructor & Destructor Documentation	11
4.1.2.1	ColorMapDecorator	11
4.1.3	Member Function Documentation	11
4.1.3.1	GetOSGNode	11
4.1.3.2	OnLoadTerrainTile	12
4.2	DrawableEntry Struct Reference	13
4.2.1	Detailed Description	13
4.2.2	Member Data Documentation	13
4.2.2.1	baseGradientTexture	13
4.2.2.2	drawable	13
4.2.2.3	sceneNode	13
4.3	DTEDLevelEnum Class Reference	14
4.3.1	Detailed Description	14
4.3.2	Member Function Documentation	14
4.3.2.1	GetNumeral	14
4.3.3	Member Data Documentation	14
4.3.3.1	FIVE	14
4.3.3.2	FOUR	14
4.3.3.3	ONE	14
4.3.3.4	THREE	14
4.3.3.5	TWO	14
4.3.3.6	ZERO	15
4.4	DTEDTerrainReader Class Reference	16
4.4.1	Detailed Description	16
4.4.2	Constructor & Destructor Documentation	17
4.4.2.1	DTEDTerrainReader	17
4.4.2.2	~DTEDTerrainReader	17
4.4.3	Member Function Documentation	17
4.4.3.1	GenerateTerrainTileCachePath	17
4.4.3.2	GetDataType	17
4.4.3.3	GetDTEDFilePath	17
4.4.3.4	GetDTEDLevel	17
4.4.3.5	OnLoadTerrainTile	17
4.4.3.6	SetDTEDLevel	18
4.5	FixedPointNoise Class Reference	19
4.5.1	Detailed Description	19
4.5.2	Member Function Documentation	19
4.5.2.1	operator()	19

4.5.2.2	operator()	19
4.5.2.3	operator()	19
4.6	GeoCoordinates Class Reference	20
4.6.1	Detailed Description	21
4.6.2	Constructor & Destructor Documentation	21
4.6.2.1	GeoCoordinates	21
4.6.3	Member Function Documentation	21
4.6.3.1	GetAltitude	21
4.6.3.2	GetCartesianPoint	22
4.6.3.3	GetCartesianPoint	22
4.6.3.4	GetLatitude	22
4.6.3.5	GetLatitude	22
4.6.3.6	GetLongitude	22
4.6.3.7	GetLongitude	22
4.6.3.8	GetOrigin	22
4.6.3.9	GetRawCartesianPoint	22
4.6.3.10	operator<	22
4.6.3.11	operator==	22
4.6.3.12	SetAltitude	22
4.6.3.13	SetCartesianPoint	22
4.6.3.14	SetLatitude	23
4.6.3.15	SetLatitude	23
4.6.3.16	SetLongitude	23
4.6.3.17	SetLongitude	23
4.6.3.18	SetOrigin	23
4.6.3.19	ToString	23
4.6.3.20	ToStringAll	23
4.6.4	Member Data Documentation	23
4.6.4.1	EQUATORIAL_RADIUS	23
4.6.4.2	FLATTENING	23
4.6.4.3	geoOrigin	23
4.6.4.4	gOriginOffset	23
4.6.4.5	mAltitude	23
4.6.4.6	mCartesianPoint	23
4.6.4.7	mLatitude	23
4.6.4.8	mLongitude	23
4.6.4.9	mUpdateCartesianPoint	24
4.6.4.10	POLAR_RADIUS	24
4.7	GeoCoordinatesException Class Reference	25
4.7.1	Detailed Description	25
4.7.2	Constructor & Destructor Documentation	25

4.7.2.1	GeoCoordinatesException	25
4.7.3	Member Data Documentation	25
4.7.3.1	OUT_OF_BOUNDS	25
4.8	GeospecificImage Struct Reference	26
4.8.1	Member Data Documentation	26
4.8.1.1	mFileName	26
4.8.1.2	mGeoTransform	26
4.8.1.3	mImage	26
4.8.1.4	mInverseGeoTransform	26
4.8.1.5	mMaxLatitude	26
4.8.1.6	mMaxLongitude	26
4.8.1.7	mMinLatitude	26
4.8.1.8	mMinLongitude	26
4.9	GeoTiffDecorator Class Reference	27
4.9.1	Detailed Description	27
4.9.2	Constructor & Destructor Documentation	27
4.9.2.1	GeoTiffDecorator	27
4.9.2.2	~GeoTiffDecorator	27
4.9.3	Member Function Documentation	27
4.9.3.1	AddGeoSpecificImage	27
4.9.3.2	CalculateBaseImage	28
4.9.3.3	GetOSGNode	28
4.9.3.4	LoadAllGeoSpecificImages	28
4.9.3.5	OnLoadTerrainTile	28
4.9.3.6	SetResultingImageDimensions	28
4.10	HeightColorMap Struct Reference	29
4.10.1	Detailed Description	29
4.10.2	Member Function Documentation	29
4.10.2.1	GetColor	29
4.11	HeightField Class Reference	30
4.11.1	Detailed Description	31
4.11.2	Constructor & Destructor Documentation	31
4.11.2.1	HeightField	31
4.11.2.2	HeightField	31
4.11.2.3	~HeightField	31
4.11.3	Member Function Documentation	31
4.11.3.1	Allocate	31
4.11.3.2	ConvertFromImage	31
4.11.3.3	ConvertFromRaw	31
4.11.3.4	ConvertToImage	32
4.11.3.5	GetHeight	32

4.11.3.6	GetHeightFieldData	32
4.11.3.7	GetInterpolatedHeight	32
4.11.3.8	GetNumColumns	32
4.11.3.9	GetNumRows	32
4.11.3.10	GetXInterval	32
4.11.3.11	GetYInterval	32
4.11.3.12	SetHeight	32
4.11.3.13	SetXInterval	32
4.11.3.14	SetYInterval	32
4.12	HeightFieldException Class Reference	33
4.12.1	Detailed Description	33
4.12.2	Member Data Documentation	33
4.12.2.1	INVALID_HEIGHTFIELD	33
4.12.2.2	INVALID_IMAGE_FORMAT	33
4.12.2.3	OUT_OF_BOUNDS	33
4.13	HeightFieldResizePolicy Class Reference	34
4.13.1	Detailed Description	34
4.13.2	Constructor & Destructor Documentation	34
4.13.2.1	HeightFieldResizePolicy	34
4.13.3	Member Data Documentation	34
4.13.3.1	NEAREST_POWER_OF_TWO	34
4.13.3.2	NEAREST_POWER_OF_TWO_PLUS_ONE	34
4.13.3.3	NONE	34
4.14	ImageUtilException Class Reference	35
4.14.1	Detailed Description	35
4.14.2	Constructor & Destructor Documentation	35
4.14.2.1	ImageUtilException	35
4.14.3	Member Data Documentation	35
4.14.3.1	INVALID_IMAGE_DIMENSIONS	35
4.14.3.2	INVALID_RASTER_FORMAT	35
4.14.3.3	LOAD_FAILED	35
4.15	ImageUtils Class Reference	36
4.15.1	Detailed Description	37
4.15.2	Member Function Documentation	37
4.15.2.1	ApplyMask	37
4.15.2.2	CalculateDetailNoise	37
4.15.2.3	CalculateScaleMapNoise	37
4.15.2.4	CreateBaseGradientMap	37
4.15.2.5	CreateDetailGradientMap	37
4.15.2.6	CreateDetailMap	37
4.15.2.7	CreateDetailScaleMap	38

4.15.2.8	EnsurePow2Image	38
4.15.2.9	ImageStats	38
4.15.2.10	LoadGeospecificLCCImage	38
4.15.2.11	MakeBaseColor	38
4.15.2.12	MakeFilteredImage	38
4.15.2.13	MakeRelativeElevationImage	38
4.15.2.14	MakeSlopeAspectImage	39
4.16	Index Struct Reference	40
4.16.1	Detailed Description	40
4.16.2	Constructor & Destructor Documentation	40
4.16.2.1	Index	40
4.16.2.2	Index	40
4.16.2.3	Index	41
4.16.2.4	Index	41
4.16.3	Member Function Documentation	41
4.16.3.1	clamp	41
4.16.3.2	operator&=	41
4.16.3.3	operator+=	41
4.16.3.4	operator-=	41
4.16.3.5	operator<	41
4.16.3.6	operator<<=	41
4.16.3.7	operator>	41
4.16.3.8	operator>>=	41
4.16.4	Member Data Documentation	41
4.16.4.1	q	41
4.16.4.2	x	41
4.16.4.3	y	41
4.17	LCCAnalyzer Class Reference	42
4.17.1	Detailed Description	43
4.17.2	Constructor & Destructor Documentation	43
4.17.2.1	LCCAnalyzer	43
4.17.2.2	~LCCAnalyzer	43
4.17.3	Member Function Documentation	43
4.17.3.1	AddGeospecificImage	43
4.17.3.2	CheckBaseLCCImages	43
4.17.3.3	CheckSlopeAndElevationMaps	43
4.17.3.4	Clear	43
4.17.3.5	ComputeProbabilityMap	43
4.17.3.6	GenerateBaseFilterImage	43
4.17.3.7	GetImageSize	43
4.17.3.8	IsGeoSpecificLCCImageLoaded	43

- 4.17.3.9 LCCHistogram 43
- 4.17.3.10 LoadAllGeoSpecificImages 44
- 4.17.3.11 MakeBaseLCCColor 44
- 4.17.3.12 MakeCombinedImage 44
- 4.17.3.13 MakeLCCMask 44
- 4.17.3.14 OutputDebugImages 44
- 4.17.3.15 ProcessLCCData 44
- 4.17.3.16 SetMaxImageSize 44
- 4.17.3.17 SetOutputDebugImages 45
- 4.18 LCCAnalyzerException Class Reference 46
 - 4.18.1 Detailed Description 46
 - 4.18.2 Constructor & Destructor Documentation 46
 - 4.18.2.1 LCCAnalyzerException 46
 - 4.18.3 Member Data Documentation 46
 - 4.18.3.1 INVALID_CACHE 46
 - 4.18.3.2 NO_VALID_GEO_IMAGES 46
- 4.19 LCCAnalyzerResourceName Class Reference 47
 - 4.19.1 Detailed Description 47
 - 4.19.2 Constructor & Destructor Documentation 47
 - 4.19.2.1 LCCAnalyzerResourceName 47
 - 4.19.3 Member Data Documentation 47
 - 4.19.3.1 BASE_COLOR 47
 - 4.19.3.2 BASE_FILTER_NAME 47
 - 4.19.3.3 BASE_LCC_COLOR 47
 - 4.19.3.4 COMPOSITE_LCC_IMAGE 47
 - 4.19.3.5 IMAGE_EXT 47
 - 4.19.3.6 LCC_IMAGE_NAME 47
 - 4.19.3.7 REL_ELEV_IMAGE 47
 - 4.19.3.8 SCENE_GRAPH 47
 - 4.19.3.9 SLOPE_IMAGE 47
 - 4.19.3.10 WATER_MASK 47
- 4.20 LCCModel Struct Reference 48
 - 4.20.1 Detailed Description 48
 - 4.20.2 Constructor & Destructor Documentation 48
 - 4.20.2.1 LCCModel 48
 - 4.20.3 Member Data Documentation 48
 - 4.20.3.1 name 48
 - 4.20.3.2 scale 48
 - 4.20.3.3 sceneNode 48
- 4.21 LCCType Class Reference 49
 - 4.21.1 Constructor & Destructor Documentation 50

4.21.1.1	LCCType	50
4.21.1.2	~LCCType	50
4.21.2	Member Function Documentation	50
4.21.2.1	AddModel	50
4.21.2.2	ClearModels	50
4.21.2.3	GetAspect	50
4.21.2.4	GetIndex	50
4.21.2.5	GetLCCName	50
4.21.2.6	GetMaxElevation	50
4.21.2.7	GetMaxSlope	50
4.21.2.8	GetMinElevation	50
4.21.2.9	GetMinSlope	50
4.21.2.10	GetModel	50
4.21.2.11	GetModel	50
4.21.2.12	GetNumberOfModels	51
4.21.2.13	GetRGB	51
4.21.2.14	RemoveModel	51
4.21.2.15	SetAspect	51
4.21.2.16	SetElevation	51
4.21.2.17	SetRelativeElevation	51
4.21.2.18	SetRGB	51
4.21.2.19	SetSlope	51
4.22	MathUtils Class Reference	52
4.22.1	Member Function Documentation	52
4.22.1.1	F2F	52
4.22.1.2	F2I	52
4.22.1.3	Fade	52
4.22.1.4	I2F	52
4.22.1.5	ILerp	52
4.22.1.6	IMul	52
4.22.1.7	Lerp	52
4.23	PagedTerrainTile Class Reference	53
4.23.1	Detailed Description	54
4.23.2	Constructor & Destructor Documentation	54
4.23.2.1	PagedTerrainTile	54
4.23.2.2	~PagedTerrainTile	54
4.23.3	Member Function Documentation	54
4.23.3.1	GetBaseTextureImage	54
4.23.3.2	GetCachePath	54
4.23.3.3	GetGeoCoordinates	54
4.23.3.4	GetHeightField	54

4.23.3.5	GetParentTerrain	54
4.23.3.6	GetParentTerrain	54
4.23.3.7	GetUpdateCache	54
4.23.3.8	IsCachingEnabled	55
4.23.3.9	ReadFromCache	55
4.23.3.10	SetBaseTextureImage	55
4.23.3.11	SetCachePath	55
4.23.3.12	SetGeoCoordinates	55
4.23.3.13	SetHeightField	55
4.23.3.14	SetUpdateCache	55
4.23.3.15	WriteToCache	55
4.23.4	Friends And Related Function Documentation	55
4.23.4.1	PagedTerrainTileFactory	55
4.23.4.2	Terrain	56
4.24	PagedTerrainTileFactory Class Reference	57
4.24.1	Detailed Description	57
4.24.2	Constructor & Destructor Documentation	57
4.24.2.1	PagedTerrainTileFactory	57
4.24.2.2	~PagedTerrainTileFactory	57
4.24.3	Member Function Documentation	57
4.24.3.1	CreateNewTile	57
4.25	PagedTerrainTileLoadState Class Reference	58
4.25.1	Detailed Description	58
4.25.2	Constructor & Destructor Documentation	58
4.25.2.1	PagedTerrainTileLoadState	58
4.25.3	Member Data Documentation	58
4.25.3.1	LOADED	58
4.25.3.2	NOT_LOADED	58
4.25.3.3	PAGING	58
4.26	PagedTerrainTileResourceName Class Reference	59
4.26.1	Detailed Description	59
4.26.2	Constructor & Destructor Documentation	59
4.26.2.1	PagedTerrainTileResourceName	59
4.26.3	Member Data Documentation	59
4.26.3.1	HEIGHTFIELD_FILENAME	59
4.27	RawData Struct Reference	60
4.27.1	Detailed Description	60
4.27.2	Member Data Documentation	60
4.27.2.1	error	60
4.27.2.2	height	60
4.27.2.3	radius	60

4.27.2.4	scale	60
4.28	SoarXCacheResourceName Class Reference	61
4.28.1	Constructor & Destructor Documentation	61
4.28.1.1	SoarXCacheResourceName	61
4.28.2	Member Data Documentation	61
4.28.2.1	BASE_GRADIENT_TEXTURE	61
4.28.2.2	DETAIL_GRADIENT_TEXTURE	61
4.28.2.3	DETAIL_SCALE_MAP	61
4.28.2.4	DETAIL_VERTEX_NOISE	61
4.28.2.5	VERTEX_DATA	61
4.29	SoarXCullCallback Struct Reference	62
4.29.1	Detailed Description	62
4.29.2	Member Function Documentation	62
4.29.2.1	cull	62
4.30	SoarXDrawable Class Reference	63
4.30.1	Detailed Description	65
4.30.2	Constructor & Destructor Documentation	65
4.30.2.1	SoarXDrawable	65
4.30.2.2	SoarXDrawable	65
4.30.2.3	~SoarXDrawable	65
4.30.3	Member Function Documentation	66
4.30.3.1	Active	66
4.30.3.2	Append	66
4.30.3.3	Build	66
4.30.3.4	CalculateError	66
4.30.3.5	CalculateRadii	66
4.30.3.6	CalculateVertexErrors	66
4.30.3.7	CalculateVertexErrorsHelper	66
4.30.3.8	CheckChildren1	66
4.30.3.9	CheckChildren2	66
4.30.3.10	Clear	66
4.30.3.11	clone	66
4.30.3.12	cloneType	66
4.30.3.13	computeBound	66
4.30.3.14	drawImplementation	66
4.30.3.15	GetBaseHorizontalResolution	66
4.30.3.16	GetBaseVerticalResolution	67
4.30.3.17	GetDetailHorizontalResolution	67
4.30.3.18	GetDetailMultiplier	67
4.30.3.19	GetDetailSize	67
4.30.3.20	GetDetailVerticalResolution	67

4.30.3.21	GetHeight	67
4.30.3.22	GetRawData	67
4.30.3.23	GetThreshold	67
4.30.3.24	GetVertex	67
4.30.3.25	GetVertex	67
4.30.3.26	LeftRefine	67
4.30.3.27	Refine	67
4.30.3.28	Render	67
4.30.3.29	Repair	67
4.30.3.30	RepairBoundingSphereHierarchy	67
4.30.3.31	RestoreDataFromCache	67
4.30.3.32	RightRefine	67
4.30.3.33	SetDetailMultiplier	68
4.30.3.34	SetDetailNoise	68
4.30.3.35	SetEyePoint	68
4.30.3.36	SetThreshold	68
4.30.3.37	supports	68
4.30.3.38	TurnCorner	68
4.30.3.39	WriteDataToCache	68
4.30.4	Friends And Related Function Documentation	68
4.30.4.1	SoarXCullCallback	68
4.31	SoarXTerrainRenderer Class Reference	69
4.31.1	Detailed Description	70
4.31.2	Member Typedef Documentation	71
4.31.2.1	DrawableMap	71
4.31.3	Constructor & Destructor Documentation	71
4.31.3.1	SoarXTerrainRenderer	71
4.31.3.2	~SoarXTerrainRenderer	71
4.31.4	Member Function Documentation	71
4.31.4.1	CalculateDetailNoise	71
4.31.4.2	CheckBaseGradientCache	71
4.31.4.3	CheckDetailGradientCache	71
4.31.4.4	CheckDetailNoiseCache	71
4.31.4.5	CheckDetailScaleCache	71
4.31.4.6	CreateFragmentShader	71
4.31.4.7	GetDetailMultiplier	71
4.31.4.8	GetHeight	71
4.31.4.9	GetNormal	71
4.31.4.10	GetRootDrawable	72
4.31.4.11	GetTerrainFragmentShader	72
4.31.4.12	GetThreshold	72

4.31.4.13	InitializeRenderer	72
4.31.4.14	OnLoadTerrainTile	72
4.31.4.15	OnUnloadTerrainTile	72
4.31.4.16	SetDetailMultiplier	72
4.31.4.17	SetEnableFog	72
4.31.4.18	SetTerrainFragmentShader	72
4.31.4.19	SetThreshold	72
4.31.4.20	SetupRenderState	72
4.31.5	Member Data Documentation	73
4.31.5.1	GRADIENT_SCALE	73
4.32	Terrain Class Reference	74
4.32.1	Detailed Description	76
4.32.2	Constructor & Destructor Documentation	76
4.32.2.1	Terrain	76
4.32.2.2	~Terrain	76
4.32.3	Member Function Documentation	76
4.32.3.1	AddDecorationLayer	76
4.32.3.2	AddResourcePath	76
4.32.3.3	ClearDecorationLayers	76
4.32.3.4	CreateTerrainTile	76
4.32.3.5	EnsureTileVisibility	76
4.32.3.6	FindAllResources	76
4.32.3.7	FindResource	77
4.32.3.8	GetCachePath	77
4.32.3.9	GetDataReader	77
4.32.3.10	GetDataReader	77
4.32.3.11	GetDataRenderer	77
4.32.3.12	GetDataRenderer	77
4.32.3.13	GetDecorationLayer	77
4.32.3.14	GetDecorationLayer	77
4.32.3.15	GetDecorationLayers	77
4.32.3.16	GetHeight	77
4.32.3.17	GetLineOfSightSpacing	77
4.32.3.18	GetLoadDistance	77
4.32.3.19	GetNumDecorationLayers	77
4.32.3.20	GetResourcePathList	78
4.32.3.21	GetTerrainTileFactory	78
4.32.3.22	HideDecorationLayer	78
4.32.3.23	HideDecorationLayer	78
4.32.3.24	IsClearLineOfSight	78
4.32.3.25	IsTerrainTileResident	78

4.32.3.26	LoadTerrainTile	78
4.32.3.27	OnMessage	78
4.32.3.28	PostFrame	78
4.32.3.29	PreFrame	78
4.32.3.30	RemoveDecorationLayer	78
4.32.3.31	RemoveDecorationLayer	78
4.32.3.32	RemoveResourcePath	78
4.32.3.33	SetCachePath	79
4.32.3.34	SetDataReader	79
4.32.3.35	SetDataRenderer	79
4.32.3.36	SetLineOfSightSpacing	79
4.32.3.37	SetLoadDistance	79
4.32.3.38	SetTerrainTileFactory	79
4.32.3.39	ShowDecorationLayer	79
4.32.3.40	ShowDecorationLayer	79
4.32.3.41	UnloadAllTerrainTiles	79
4.32.3.42	UnloadTerrainTile	79
4.32.4	Member Data Documentation	79
4.32.4.1	mResidentTiles	79
4.32.4.2	mTilesToLoadQ	79
4.32.4.3	mTilesToUnloadQ	79
4.33	TerrainCullCallback Class Reference	80
4.33.1	Constructor & Destructor Documentation	80
4.33.1.1	TerrainCullCallback	80
4.33.2	Member Function Documentation	80
4.33.2.1	operator()	80
4.34	TerrainDataReader Class Reference	81
4.34.1	Detailed Description	82
4.34.2	Constructor & Destructor Documentation	82
4.34.2.1	TerrainDataReader	82
4.34.2.2	~TerrainDataReader	82
4.34.3	Member Function Documentation	82
4.34.3.1	ConvertHeightField	82
4.34.3.2	GenerateTerrainTileCachePath	82
4.34.3.3	GetDataType	82
4.34.3.4	GetInterpolatedHeight	82
4.34.3.5	GetParentTerrain	83
4.34.3.6	GetParentTerrain	83
4.34.3.7	GetResizePolicy	83
4.34.3.8	OnLoadTerrainTile	83
4.34.3.9	OnUnloadTerrainTile	83

4.34.3.10	ScaleOSGHeightField	83
4.34.3.11	SetResizePolicy	83
4.34.4	Friends And Related Function Documentation	84
4.34.4.1	Terrain	84
4.35	TerrainDataReaderException Class Reference	85
4.35.1	Detailed Description	85
4.35.2	Constructor & Destructor Documentation	85
4.35.2.1	TerrainDataReaderException	85
4.35.3	Member Data Documentation	85
4.35.3.1	COULD_NOT_READ_DATA	85
4.35.3.2	DATA_RESOURCE_NOT_FOUND	85
4.35.3.3	READER_PLUGIN_NOT_FOUND	85
4.36	TerrainDataRenderer Class Reference	86
4.36.1	Detailed Description	86
4.36.2	Constructor & Destructor Documentation	87
4.36.2.1	TerrainDataRenderer	87
4.36.2.2	~TerrainDataRenderer	87
4.36.3	Member Function Documentation	87
4.36.3.1	GetHeight	87
4.36.3.2	GetNormal	87
4.36.3.3	GetParentTerrain	87
4.36.3.4	GetParentTerrain	87
4.36.3.5	GetRootDrawable	87
4.36.3.6	OnLoadTerrainTile	87
4.36.3.7	OnUnloadTerrainTile	87
4.36.4	Friends And Related Function Documentation	88
4.36.4.1	Terrain	88
4.37	TerrainDataType Class Reference	89
4.37.1	Detailed Description	89
4.37.2	Constructor & Destructor Documentation	89
4.37.2.1	TerrainDataType	89
4.37.3	Member Data Documentation	89
4.37.3.1	DTED	89
4.38	TerrainDecorationLayer Class Reference	90
4.38.1	Detailed Description	91
4.38.2	Constructor & Destructor Documentation	91
4.38.2.1	TerrainDecorationLayer	91
4.38.2.2	~TerrainDecorationLayer	91
4.38.3	Member Function Documentation	91
4.38.3.1	GetOSGNode	91
4.38.3.2	GetParentTerrain	91

4.38.3.3	GetParentTerrain	91
4.38.3.4	IsVisible	91
4.38.3.5	OnLoadTerrainTile	91
4.38.3.6	OnTerrainTileResident	91
4.38.3.7	OnUnloadTerrainTile	92
4.38.3.8	SetVisible	92
4.38.4	Friends And Related Function Documentation	92
4.38.4.1	Terrain	92
4.39	TerrainException Class Reference	93
4.39.1	Detailed Description	93
4.39.2	Constructor & Destructor Documentation	93
4.39.2.1	TerrainException	93
4.39.3	Member Data Documentation	93
4.39.3.1	INVALID_DATA_READER	93
4.39.3.2	INVALID_DATA_RENDERER	93
4.39.3.3	INVALID_DECORATION_LAYER	93
4.39.3.4	INVALID_RESOURCE_PATH	93
4.39.3.5	NULL_POINTER	93
4.39.3.6	UNSUPPORTED_DATA_FORMAT	94
4.40	TerrainRendererException Class Reference	95
4.40.1	Detailed Description	95
4.40.2	Constructor & Destructor Documentation	95
4.40.2.1	TerrainRendererException	95
4.40.3	Member Data Documentation	95
4.40.3.1	INVALID_HEIGHTFIELD_DATA	95
4.41	VegetationDecorator Class Reference	96
4.41.1	Detailed Description	97
4.41.2	Member Typedef Documentation	97
4.41.2.1	VegetationMap	97
4.41.3	Constructor & Destructor Documentation	97
4.41.3.1	VegetationDecorator	97
4.41.3.2	~VegetationDecorator	97
4.41.4	Member Function Documentation	97
4.41.4.1	AddVegetation	97
4.41.4.2	BuildVegetationForType	97
4.41.4.3	GetLCCAnalyzer	97
4.41.4.4	GetNumLooks	97
4.41.4.5	GetOSGNode	97
4.41.4.6	GetVegetation	98
4.41.4.7	GetVegType	98
4.41.4.8	OnLoadTerrainTile	98

4.41.4.9	OnTerrainTileResident	98
4.41.4.10	OnUnloadTerrainTile	98
4.41.4.11	SetGeospecificImage	98
4.41.4.12	SetLCCTypes	98
4.41.4.13	SetMaxVegetationPerCell	98
4.41.4.14	SetRandomSeed	98
4.41.4.15	SetVegetationDistance	98
4.42	VegetationException Class Reference	99
4.42.1	Detailed Description	99
4.42.2	Constructor & Destructor Documentation	99
4.42.2.1	VegetationException	99
4.42.3	Member Data Documentation	99
4.42.3.1	INVALID_LCC_TYPES	99
4.42.3.2	INVALID_SLOPE_ASPECT_IMAGE	99
4.43	Vertex Struct Reference	100
4.43.1	Detailed Description	100
4.43.2	Constructor & Destructor Documentation	100
4.43.2.1	Vertex	100
4.43.2.2	Vertex	100
4.43.3	Member Data Documentation	100
4.43.3.1	error	100
4.43.3.2	index	100
4.43.3.3	position	100
4.43.3.4	radius	100
5	File Documentation	101
5.1	colormapdecorator.cpp File Reference	101
5.2	colormapdecorator.h File Reference	102
5.3	dtedterrainreader.cpp File Reference	103
5.4	dtedterrainreader.h File Reference	104
5.5	fixedpointnoise.cpp File Reference	105
5.6	fixedpointnoise.h File Reference	106
5.7	geocoordinates.cpp File Reference	107
5.8	geocoordinates.h File Reference	108
5.9	geotiffdecorator.cpp File Reference	109
5.10	geotiffdecorator.h File Reference	110
5.11	heightfield.cpp File Reference	111
5.12	heightfield.h File Reference	112
5.13	imageutils.cpp File Reference	113
5.14	imageutils.h File Reference	114
5.15	lccanalyzer.cpp File Reference	115

5.16	lccanalyzer.h File Reference	116
5.17	lcctype.cpp File Reference	117
5.18	lcctype.h File Reference	118
5.19	mainpage.h File Reference	119
	5.19.1 Detailed Description	119
5.20	mathutils.cpp File Reference	120
5.21	mathutils.h File Reference	121
5.22	pagedterraintile.cpp File Reference	122
5.23	pagedterraintile.h File Reference	123
5.24	pagedterraintilefactory.h File Reference	124
5.25	soarxdrawable.cpp File Reference	125
5.26	soarxdrawable.h File Reference	126
5.27	soarxterrainrenderer.cpp File Reference	127
5.28	soarxterrainrenderer.h File Reference	128
5.29	terrain.cpp File Reference	129
5.30	terrain.h File Reference	130
5.31	terrain_export.h File Reference	131
	5.31.1 Define Documentation	131
	5.31.1.1 DT_TERRAIN_EXPORT	131
5.32	terraindatareader.cpp File Reference	132
5.33	terraindatareader.h File Reference	133
5.34	terraindatarenderer.cpp File Reference	134
5.35	terraindatarenderer.h File Reference	135
5.36	terraindatatype.cpp File Reference	136
5.37	terraindatatype.h File Reference	137
5.38	terraindecorationlayer.cpp File Reference	138
5.39	terraindecorationlayer.h File Reference	139
5.40	vegetationdecorator.cpp File Reference	140
5.41	vegetationdecorator.h File Reference	141

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** (p. 7) (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!

Directory Documentation

2.1 inc/dtTerrain/ Directory Reference

Files

- file **colormapdecorator.h**
- file **dtedterrainreader.h**
- file **fixedpointnoise.h**
- file **geocoordinates.h**
- file **geotiffdecorator.h**
- file **heightfield.h**
- file **imageutils.h**
- file **lccanalyzer.h**
- file **lcctype.h**
- file **mainpage.h**
- file **mathutils.h**
- file **pagedterrintile.h**
- file **pagedterrintilefactory.h**
- file **soarxdrawable.h**
- file **soarxterrainrenderer.h**
- file **terrain.h**
- file **terrain_export.h**
- file **terraindatareader.h**
- file **terraindatarenderer.h**
- file **terraindatatype.h**
- file **terraindecorationlayer.h**
- file **vegetationdecorator.h**

2.2 src/dtTerrain/ Directory Reference

Files

- file **colormapdecorator.cpp**
- file **dtedterrainreader.cpp**
- file **fixedpointnoise.cpp**
- file **geocoordinates.cpp**
- file **geotiffdecorator.cpp**
- file **heightfield.cpp**
- file **imageutils.cpp**
- file **lccanalyzer.cpp**
- file **lcctype.cpp**
- file **mathutils.cpp**
- file **pagedterrintile.cpp**
- file **soarxdrawable.cpp**
- file **soarxterrainrenderer.cpp**
- file **terrain.cpp**
- file **terraindatareader.cpp**
- file **terraindatarenderer.cpp**
- file **terraindatatype.cpp**
- file **terraindecorationlayer.cpp**
- file **vegetationdecorator.cpp**

2.3 inc/ Directory Reference

Directories

- directory dtTerrain

2.4 src/ Directory Reference

Directories

- directory dtTerrain

Namespace Documentation

3.1 dtTerrain Namespace Reference

An extensible library for rendering terrain databases.

Classes

- class **ColorMapDecorator**
This terrain decorator is a terrain tile base texture decorator.
- class **DTEDTerrainReader**
This data reader loads DTED data mapping to a specified latitude and longitude.
- class **FixedPointNoise**
This is a fixed point implementation of an improved Ken Perlin noise generator.
- class **GeoCoordinates**
This class is intended to represent cartesian and geographic coordinates.
- class **GeoCoordinatesException**
This class contains the various types of exceptions that the coordinate system class may throw during a given operation.
- class **GeoTiffDecorator**
This class is a terrain texturing decorator that supports the mapping of geospecific images onto terrain tiles.
- class **HeightField**
This class wraps an array of elevation posts.
- class **HeightFieldException**
This enumeration defines the different types of exceptions the height field may throw if an error occurs.
- class **HeightFieldResizePolicy**
This enumeration specifies the valid resize policies the data readers should follow when loading a new heightfield.
- class **ImageUtilException**
Defines exceptions that may be thrown from the image utility methods.
- class **ImageUtils**
This class is a static class containing many methods for creating images procedurally, concatenating images, and building images from source height data.
- class **LCCAnalyzer**
The LCC Analyzer calculates LCC data for use in the vegetation decorator.

- class **LCCAnalyzerException**
Defines the exceptions thrown by the lcc analyzer.
- class **LCCAnalyzerResourceName**
This enumeration identifies the resources that are cached and managed by the vegetation decorator layer.
- class **LCCType**
- class **MathUtils**
- class **PagedTerrainTile**
This class is the base class for a tile of data.
- class **PagedTerrainTileFactory**
This class is the base for a simple factory used to create new terrain tiles.
- class **PagedTerrainTileLoadState**
This enumeration defines the different states a tile could be in throughout its lifetime.
- class **PagedTerrainTileResourceName**
This enumeration is used to identity built in resources that the terrain tiles are aware of.
- class **SoarXCacheResourceName**
- struct **SoarXCullCallback**
This callback is used by the SoarX renderer to update the camera's position or eyepoint which is needed by the refinement process to calculate projected vertex errors.
- class **SoarXDrawable**
This class encapsulates the SoarX rendering algorithm itself.
- class **SoarXTerrainRenderer**
This renderer is an integration/implementation of the SoarX terrain rendering algorithm.
- class **Terrain**
This is the base terrain class that provides most of the functionality required for terrain rendering.
- class **TerrainCullCallback**
- class **TerrainDataReader**
This is mostly an interface class to the data reader functionality of Delta3D terrains.
- class **TerrainDataReaderException**
These exceptions are used by terrain data readers when an error occurs during the load process.
- class **TerrainDataRenderer**
This class is the interface for a terrain renderer.
- class **TerrainDataType**
This class defines the various types of terrain data supported by default in Delta3D.
- class **TerrainDecorationLayer**
This class is a terrain decoration layer.
- class **TerrainException**
This class enumerates the different exceptions that can be thrown by terrain instances.
- class **TerrainRendererException**
This enumeration contains generic errors that could occur in terrain renderers.

- class **VegetationDecorator**

This class is the vegetation decorator layer.

- class **VegetationException**

Defines the exception used by the vegetation decorator.

Functions

- **IMPLEMENT_ENUM** (**VegetationException**)
- **IMPLEMENT_ENUM** (**TerrainDataType**)
- **IMPLEMENT_ENUM** (**TerrainRendererException**)
- **IMPLEMENT_ENUM** (**HeightFieldResizePolicy**)
- **IMPLEMENT_ENUM** (**TerrainDataReaderException**)
- **IMPLEMENT_ENUM** (**TerrainException**)
- **IMPLEMENT_ENUM** (**SoarXCacheResourceName**)
- **IMPLEMENT_ENUM** (**PagedTerrainTileLoadState**)
- **IMPLEMENT_ENUM** (**PagedTerrainTileResourceName**)
- **IMPLEMENT_ENUM** (**LCCAnalyzerResourceName**)
- **IMPLEMENT_ENUM** (**LCCAnalyzerException**)
- **IMPLEMENT_ENUM** (**ImageUtilException**)
- **IMPLEMENT_ENUM** (**HeightFieldException**)
- **IMPLEMENT_ENUM** (**GeoCoordinatesException**)
- **IMPLEMENT_ENUM** (**DTEDTerrainReader::DTEDLevelEnum**)
- **IMPLEMENT_MANAGEMENT_LAYER** (**Terrain**)
- bool **SimpleLineOfSight** (**dtTerrain::Terrain** *terrain, const osg::Vec3 &pointOne, const osg::Vec3 &pointTwo)

Trivial Line Of Sight calculator.
- std::string **vec3dToString** (const osg::Vec3d &pt)
- std::string **vec3ToString** (const osg::Vec3 &pt)

Variables

- const bool **BOTTOM** = false

Constants used during the refinement process.
- const std::string **COLOR_MAP_IMAGE_NAME** = "colormap_decorator_image.rgb"
- const std::string **GEOTIFF_IMAGE_NAME** = "geo_tiff_decorator_image.rgb"
- const int **I** = (1 << 12)
- const bool **SIDE** = true

3.1.1 Detailed Description

An extensible library for rendering terrain databases.

3.1.2 Function Documentation

- 3.1.2.1 dtTerrain::IMPLEMENT_ENUM (VegetationException)
- 3.1.2.2 dtTerrain::IMPLEMENT_ENUM (TerrainDataType)
- 3.1.2.3 dtTerrain::IMPLEMENT_ENUM (TerrainRendererException)
- 3.1.2.4 dtTerrain::IMPLEMENT_ENUM (HeightFieldResizePolicy)
- 3.1.2.5 dtTerrain::IMPLEMENT_ENUM (TerrainDataReaderException)
- 3.1.2.6 dtTerrain::IMPLEMENT_ENUM (TerrainException)
- 3.1.2.7 dtTerrain::IMPLEMENT_ENUM (SoarXCacheResourceName)
- 3.1.2.8 dtTerrain::IMPLEMENT_ENUM (PagedTerrainTileLoadState)
- 3.1.2.9 dtTerrain::IMPLEMENT_ENUM (PagedTerrainTileResourceName)
- 3.1.2.10 dtTerrain::IMPLEMENT_ENUM (LCCAnalyzerResourceName)
- 3.1.2.11 dtTerrain::IMPLEMENT_ENUM (LCCAnalyzerException)
- 3.1.2.12 dtTerrain::IMPLEMENT_ENUM (ImageUtilException)
- 3.1.2.13 dtTerrain::IMPLEMENT_ENUM (HeightFieldException)
- 3.1.2.14 dtTerrain::IMPLEMENT_ENUM (GeoCoordinatesException)
- 3.1.2.15 dtTerrain::IMPLEMENT_ENUM (DTEDTerrainReader::DTEDLevelEnum)
- 3.1.2.16 dtTerrain::IMPLEMENT_MANAGEMENT_LAYER (Terrain)
- 3.1.2.17 **bool SimpleLineOfSight (dtTerrain::Terrain * *terrain*, const osg::Vec3 & *pointOne*, const osg::Vec3 & *pointTwo*)**

Trivial Line Of Sight calculator. We basically walk along the ray between the two points, doing point sampling. This is brute force and not the most efficient means of doing this test, but this is the first implementation.

This routine ONLY checks visibility against ground terrain! We do not check test against other objects, such as buildings or other vehicles.

Sampling rate: our DTED data is sampled at about 30m resolution. Therefore we adjust the postSpacing to be smaller than that, and use that to step along the viewing ray.

This uses dtTerrain::GetHeight which uses the current Renderer this may be cause lots extra work, especially when called repeatedly might be better to do it directly on heightField of tile?

Parameters

terrain Test the line of sight against this instance of **Terrain** (p. 74).

pointOne The start point.

pointTwo The end point.

Precondition `terrain != 0`

- 3.1.2.18 **std::string dtTerrain::vec3dToString (const osg::Vec3d & *pf*)**

- 3.1.2.19 **std::string dtTerrain::vec3ToString (const osg::Vec3 & *pf*)**

3.1.3 Variable Documentation

- 3.1.3.1 **const bool BOTTOM = false**

Constants used during the refinement process.

- 3.1.3.2 **const std::string COLOR_MAP_IMAGE_NAME = "colormap_decorator_image.rgb"**

- 3.1.3.3 **const std::string GEOTIFF_IMAGE_NAME = "geo_tiff_decorator_image.rgb"**

- 3.1.3.4 **const int I = (1 << 12)**

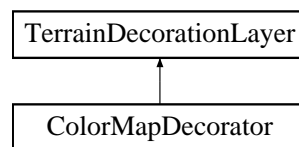
- 3.1.3.5 **const bool SIDE = true**

Class Documentation

4.1 ColorMapDecorator Class Reference

This terrain decorator is a terrain tile base texture decorator.

`#include <inc/dtTerrain/colormapdecorator.h>`Inheritance diagram for ColorMapDecorator::



Public Member Functions

- **ColorMapDecorator** (const std::string &name="ColorMapDecorator")
Constructs the decoration layer.
- virtual osg::Node * **GetOSGNode** ()
Since this decorator does not add any geometry to the terrain, just return NULL here.
- virtual void **OnLoadTerrainTile** (**PagedTerrainTile** &tile)
Generates a base texture map for this tile using the current height-color mapping assigned to this decorator.

4.1.1 Detailed Description

This terrain decorator is a terrain tile base texture decorator. It calculates a texture map for the tile based on its height field. Color values may be mapped to specific height values. All color values falling between these posts are interpolated forming a smooth gradient.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ColorMapDecorator (const std::string & name = "ColorMapDecorator")

Constructs the decoration layer.

4.1.3 Member Function Documentation

4.1.3.1 virtual osg::Node* GetOSGNode () [inline, virtual]

Since this decorator does not add any geometry to the terrain, just return NULL here.

Implements **TerrainDecorationLayer** (p. 91).

4.1.3.2 void OnLoadTerrainTile (PagedTerrainTile & *tile*) [virtual]

Generates a base texture map for this tile using the current height-color mapping assigned to this decorator.
Parameters

tile The tile with which to generate the base texture.

Implements **TerrainDecorationLayer** (p. 91).

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

- **colormapdecorator.h**
- **colormapdecorator.cpp**

4.2 DrawableEntry Struct Reference

This structure represents a drawable tile.

```
#include <inc/dtTerrain/soarxterrainrenderer.h>
```

Public Attributes

- dtCore::RefPtr< osg::Texture2D > **baseGradientTexture**
- dtCore::RefPtr< **SoarXDrawable** > **drawable**
- dtCore::RefPtr< osg::MatrixTransform > **sceneNode**

4.2.1 Detailed Description

This structure represents a drawable tile. There is a one-to-one mapping of drawable entries to paged terrain tiles in the renderer.

4.2.2 Member Data Documentation

4.2.2.1 dtCore::RefPtr<osg::Texture2D> baseGradientTexture

4.2.2.2 dtCore::RefPtr<SoarXDrawable> drawable

4.2.2.3 dtCore::RefPtr<osg::MatrixTransform> sceneNode

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

- **soarxterrainrenderer.h**

4.3 DTEDLevelEnum Class Reference

This enumeration represents the different DTED levels available for loading.

```
#include <inc/dtTerrain/dtedterrainreader.h>
```

Public Member Functions

- int **GetNumeral ()** const
Gets the integer representation of this enumeration.

Static Public Attributes

- static const **DTEDLevelEnum FIVE**
DTED data with 1m resolution data posts.
- static const **DTEDLevelEnum FOUR**
DTED data with 3m resolution data posts.
- static const **DTEDLevelEnum ONE**
DTED data with 100m resolution data posts.
- static const **DTEDLevelEnum THREE**
DTED data with 10m resolution data posts.
- static const **DTEDLevelEnum TWO**
DTED data with 30m resolution data posts.
- static const **DTEDLevelEnum ZERO**
DTED data with 1km resolution data posts.

4.3.1 Detailed Description

This enumeration represents the different DTED levels available for loading.

4.3.2 Member Function Documentation

4.3.2.1 int GetNumeral () const [inline]

Gets the integer representation of this enumeration.

4.3.3 Member Data Documentation

4.3.3.1 const DTEDTerrainReader::DTEDLevelEnum FIVE [static]

DTED data with 1m resolution data posts.

4.3.3.2 const DTEDTerrainReader::DTEDLevelEnum FOUR [static]

DTED data with 3m resolution data posts.

4.3.3.3 const DTEDTerrainReader::DTEDLevelEnum ONE [static]

DTED data with 100m resolution data posts.

4.3.3.4 const DTEDTerrainReader::DTEDLevelEnum THREE [static]

DTED data with 10m resolution data posts.

4.3.3.5 const DTEDTerrainReader::DTEDLevelEnum TWO [static]

DTED data with 30m resolution data posts.

4.3.3.6 const DTEDTerrainReader::DTEDLevelEnum ZERO [static]

DTED data with 1km resolution data posts.

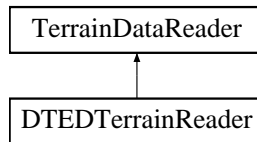
The documentation for this class was generated from the following files:

- **dtedterrainreader.h**
- **dtedterrainreader.cpp**

4.4 DTEDTerrainReader Class Reference

This data reader loads DTED data mapping to a specified latitude and longitude.

#include <inc/dtTerrain/dtedterrainreader.h> Inheritance diagram for DTEDTerrainReader::



Classes

- class **DTEDLevelEnum**

This enumeration represents the different DTED levels available for loading.

Public Member Functions

- **DTEDTerrainReader** (const **DTEDLevelEnum** &level=**DTEDLevelEnum::ONE**, const std::string &name="DTEDReader")
Constructs the terrain reader.
- const std::string **GenerateTerrainTileCachePath** (const **PagedTerrainTile** &tile)
This generates the cache path for the specified tile.
- const **TerrainDataType** & **GetDataType** () const
Gets the type of terrain data this reader supports.
- const **DTEDLevelEnum** & **GetDTEDLevel** () const
Gets the current DTED level this reader is trying to load.
- virtual bool **OnLoadTerrainTile** (**PagedTerrainTile** &tile)
Called by the terrain when a tile needs to be loaded.
- void **SetDTEDLevel** (const **DTEDLevelEnum** &level)
Sets the DTED level to load.

Protected Member Functions

- virtual ~**DTEDTerrainReader** ()
Destroys the terrain reader.
- std::string **GetDTEDFilePath** (const std::string &lat, const std::string &lon, int level)
Searches the list of file resource paths for DTED data of the given latitude, longitude and level.

4.4.1 Detailed Description

This data reader loads DTED data mapping to a specified latitude and longitude. Under the covers this reader uses GDAL to load the actual heightfield values. Note, to load specific DTED levels, one must specify the level to load. By default, the reader will attempt to load DTED level 1.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 DTEDTerrainReader (const DTEDLevelEnum & *level* = DTEDLevelEnum::ONE, const std::string & *name* = "DTEDReader")

Constructs the terrain reader. Parameters

level The level of DTED data to load.

4.4.2.2 ~DTEDTerrainReader () [protected, virtual]

Destroys the terrain reader.

4.4.3 Member Function Documentation

4.4.3.1 const std::string GenerateTerrainTileCachePath (const PagedTerrainTile & *tile*) [virtual]

This generates the cache path for the specified tile. The cache path is equal to: \$Latitude_\$Longitude_-\$DTEDLevel. Parameters

tile The tile with which to generate the path for.

Returns A string containing the generated cache path.

Implements **TerrainDataReader** (p. 82).

4.4.3.2 const TerrainDataType& GetDataType () const [inline, virtual]

Gets the type of terrain data this reader supports. Returns **TerrainDataType::DTED** (p. 89).

Implements **TerrainDataReader** (p. 82).

4.4.3.3 std::string GetDTEDFilePath (const std::string & *lat*, const std::string & *lon*, int *level*) [protected]

Searches the list of file resource paths for DTED data of the given latitude, longitude and level. Parameters

basePath String containing the root DTED path.

lat Latitude (ex. e90)

lon Longitude (ex. w150)

Returns The full path to the DTED file resource.

4.4.3.4 const DTEDLevelEnum& GetDTEDLevel () const [inline]

Gets the current DTED level this reader is trying to load. Returns An enumeration corresponding to appropriate DTED level.

4.4.3.5 bool OnLoadTerrainTile (PagedTerrainTile & *tile*) [virtual]

Called by the terrain when a tile needs to be loaded. This method will first check the tile's cache for the heightfield. If it is not found, this method will load the DTED data for the specified tile. Parameters

tile The tile to load. Based on its latitude and longitude position and the reader's max DTED level, the appropriate data will be loaded.

Note If any errors occur during the load process, the appropriate exception is thrown.

Implements **TerrainDataReader** (p. 83).

4.4.3.6 void SetDTEDLevel (const DTEDLevelEnum & *level*)

Sets the DTED level to load. Parameters

level The level of data to load. By default, this value is set to **DTEDLevelEnum::ONE** (p. 14).

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

- **dtedterrainreader.h**
- **dtedterrainreader.cpp**

4.5 FixedPointNoise Class Reference

This is a fixed point implementation of an improved Ken Perlin noise generator.

```
#include <inc/dtTerrain/fixedpointnoise.h>
```

Public Member Functions

- `int operator()` (int x)
- `int operator()` (int x, int y)
2D noise.
- `int operator()` (int x, int y, int z)
3D noise.

4.5.1 Detailed Description

This is a fixed point implementation of an improved Ken Perlin noise generator. This code comes primarily from an implementation found in the SoarX terrain rendering architecture developed by Andras Balogh and can be found at <http://web.interware.hu/bandi/ranger.html>.

4.5.2 Member Function Documentation

4.5.2.1 `int operator()` (int x)

4.5.2.2 `int operator()` (int x, int y)

2D noise.

4.5.2.3 `int operator()` (int x, int y, int z)

3D noise.

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

- `fixedpointnoise.h`
- `fixedpointnoise.cpp`

4.6 GeoCoordinates Class Reference

This class is intended to represent cartesian and geographic coordinates.

```
#include <inc/dtTerrain/geocoordinates.h>
```

Public Member Functions

- **GeoCoordinates** ()
Constructs the coordinate system.
- double **GetAltitude** () const
Gets this coordinate's altitude.
- void **GetCartesianPoint** (osg::Vec3 &point)
- const osg::Vec3 & **GetCartesianPoint** ()
Gets this coordinate's location in cartesian space.
- void **GetLatitude** (int °rees, int &minutes, int &seconds)
Gets the latitude origin of this coordinate.
- double **GetLatitude** () const
Gets the latitude origin of this coordinate.
- void **GetLongitude** (int °rees, int &minutes, int &seconds)
Gets the longitude origin of this coordinate.
- double **GetLongitude** () const
Gets the longitude origin of this coordinate.
- bool **operator<** (const **GeoCoordinates** &rhs) const
Performs a less than comparison on this coordinate's cartesian location with that of the other.
- bool **operator==** (const **GeoCoordinates** &rhs) const
Compares the two coordinate's cartesian location.
- void **SetAltitude** (double alt)
Sets the altitude of this coordinate.
- void **SetCartesianPoint** (const osg::Vec3 &newLocation)
Sets the location of this coordinate in cartesian space.
- void **SetLatitude** (double decimalDegrees)
Sets the latitude of this coordinate.
- void **SetLatitude** (int degrees, int minutes=0, int seconds=0)
Sets the latitude of this coordinate.
- void **SetLongitude** (double decimalDegrees)
Sets the longitude of this coordinate.
- void **SetLongitude** (int degrees, int minutes=0, int seconds=0)
Sets the longitude of this coordinate.
- std::string **Tostring** () const
- std::string **TostringAll** () const

Static Public Member Functions

- static void **GetOrigin** (**GeoCoordinates** &geoOrigin)
- static void **SetOrigin** (const **GeoCoordinates** &geoOrigin)

Static Public Attributes

- static const double **EQUATORIAL_RADIUS** = 6378137.0
Equatorial Radius in meters. (WGS84).
- static const double **FLATTENING** = 1.0/298.257223563
Flattening coefficient in meters. (WGS84).
- static const double **POLAR_RADIUS** = 6356752.3142451794975639668
Polar Radius in meters. (WGS84).

Protected Member Functions

- void **GetRawCartesianPoint** (osg::Vec3d &point)

Protected Attributes

- double **mAltitude**
Elevation origin.
- osg::Vec3 **mCartesianPoint**
Cached version of this coordinate systems's origin in cartesian space.
- double **mLatitude**
Latitude origin in decimal degrees.
- double **mLongitude**
Longitude origin in decimal degrees.
- bool **mUpdateCartesianPoint**
Flag to indicate the cartesian origin needs to be updated.

Static Protected Attributes

- static **GeoCoordinates** geoOrigin
- static osg::Vec3d **gOriginOffset**

4.6.1 Detailed Description

This class is intended to represent cartesian and geographic coordinates. The conversion between latitude and longitude to cartesian is rudimentary at best. Altitude is arbitrary. Future versions of this class should use a proper geodetic representation of latitude, longitude, and altitude but for now, this will suffice.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 GeoCoordinates ()

Constructs the coordinate system.

4.6.3 Member Function Documentation

4.6.3.1 double GetAltitude () const [inline]

Gets this coordinate's altitude. Returns The current altitude.

4.6.3.2 void GetCartesianPoint (osg::Vec3 & *point*)**4.6.3.3 const osg::Vec3 & GetCartesianPoint ()**

Gets this coordinate's location in cartesian space. Returns A 3D point in cartesian space. (meters). Note If the origin was previously set via any of the geocentric methods, the origin in cartesian space is calculated and cached as required.

4.6.3.4 void GetLatitude (int & *degrees*, int & *minutes*, int & *seconds*)

Gets the latitude origin of this coordinate. Parameters

degrees Stores the latitude degrees.

minutes Stores the latitude minutes.

seconds Stores the latitude seconds.

4.6.3.5 double GetLatitude () const [inline]

Gets the latitude origin of this coordinate. Returns The latitude in decimal degrees.

4.6.3.6 void GetLongitude (int & *degrees*, int & *minutes*, int & *seconds*)

Gets the longitude origin of this coordinate. Parameters

degrees Stores the latitude degrees.

minutes Stores the latitude minutes.

seconds Stores the latitude seconds.

4.6.3.7 double GetLongitude () const [inline]

Gets the longitude origin of this coordinate. Returns The longitude in decimal degrees.

4.6.3.8 void GetOrigin (GeoCoordinates & *geoOrigin*) [static]**4.6.3.9 void GetRawCartesianPoint (osg::Vec3d & *point*) [protected]****4.6.3.10 bool operator< (const GeoCoordinates & *rhs*) const [inline]**

Performs a less than comparison on this coordinate's cartesian location with that of the other. Parameters

rhs The other coordinate the check against.

Returns True if this is less than the other.

4.6.3.11 bool operator== (const GeoCoordinates & *rhs*) const [inline]

Compares the two coordinate's cartesian location. Returns True if they are equal, false otherwise.

4.6.3.12 void SetAltitude (double *alt*) [inline]

Sets the altitude of this coordinate. Parameters

alt Altitude. The interpretation of this value is application specific for the time being. Future versions of this class should represent altitude using as the height above the reference (WGS84) ellipsoid.

4.6.3.13 void SetCartesianPoint (const osg::Vec3 & *newLocation*)

Sets the location of this coordinate in cartesian space. Parameters

newLocation The new location in cartesian coordinates.

Note Internally, this origin in cartesian space is also converted to latitude, longitude coordinates.

4.6.3.14 void SetLatitude (double *decimalDegrees*) [inline]

Sets the latitude of this coordinate. Parameters

decimalDegrees The new latitude in decimal degrees.

4.6.3.15 void SetLatitude (int *degrees*, int *minutes* = 0, int *seconds* = 0)

Sets the latitude of this coordinate. This is stored internally as decimal degrees. Parameters

degrees Latitude degrees ranges from -90 (south pole) to +90 (north pole). If degrees is out of this range, an OUT_OF_BOUNDS exception is thrown.

minutes Ranges from 0 - 59. If minutes is out of this range, an OUT_OF_BOUNDS exception is thrown.

seconds Ranges from 0 - 59. If seconds is out of this range, an OUT_OF_BOUNDS exception is thrown.

4.6.3.16 void SetLongitude (double *decimalDegrees*) [inline]

Sets the longitude of this coordinate. Parameters

decimalDegrees The new longitude in decimal degrees.

4.6.3.17 void SetLongitude (int *degrees*, int *minutes* = 0, int *seconds* = 0)

Sets the longitude of this coordinate. This is stored internally as decimal degrees. Parameters

degrees Longitude degrees ranges from -180 (180 degrees west) to +180 (180 degrees east). If degrees is out of this range, an OUT_OF_BOUNDS exception is thrown.

minutes Ranges from 0 - 59. If minutes is out of this range, an OUT_OF_BOUNDS exception is thrown.

seconds Ranges from 0 - 59. If seconds is out of this range, an OUT_OF_BOUNDS exception is thrown.

4.6.3.18 void SetOrigin (const GeoCoordinates & *geoOrigin*) [static]**4.6.3.19 std::string ToString () const****4.6.3.20 std::string ToStringAll () const****4.6.4 Member Data Documentation****4.6.4.1 const double EQUATORIAL_RADIUS = 6378137.0 [static]**

Equatorial Radius in meters. (WGS84).

4.6.4.2 const double FLATTENING = 1.0/298.257223563 [static]

Flattening coefficient in meters. (WGS84).

4.6.4.3 GeoCoordinates geoOrigin [static, protected]**4.6.4.4 osg::Vec3d gOriginOffset [static, protected]****4.6.4.5 double mAltitude [protected]**

Elevation origin.

4.6.4.6 osg::Vec3 mCartesianPoint [protected]

Cached version of this coordinate systems's origin in cartesian space.

4.6.4.7 double mLatitude [protected]

Latitude origin in decimal degrees.

4.6.4.8 double mLongitude [protected]

Longitude origin in decimal degrees.

4.6.4.9 bool mUpdateCartesianPoint [protected]

Flag to indicate the cartesian origin needs to be updated.

4.6.4.10 const double POLAR_RADIUS = 6356752.3142451794975639668 [static]

Polar Radius in meters. (WGS84).

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

- **geocoordinates.h**
- **geocoordinates.cpp**

4.7 GeoCoordinatesException Class Reference

This class contains the various types of exceptions that the coordinate system class may throw during a given operation.

```
#include <inc/dtTerrain/geocoordinates.h>
```

Static Public Attributes

- static **GeoCoordinatesException OUT_OF_BOUNDS**
Thrown if the user attempts to set an invalid value.

Protected Member Functions

- **GeoCoordinatesException** (const std::string &name)

4.7.1 Detailed Description

This class contains the various types of exceptions that the coordinate system class may throw during a given operation.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 GeoCoordinatesException (const std::string & name) [inline, protected]

4.7.3 Member Data Documentation

4.7.3.1 GeoCoordinatesException OUT_OF_BOUNDS [static]

Thrown if the user attempts to set an invalid value.

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

- **geocoordinates.h**
- **geocoordinates.cpp**

4.8 GeospecificImage Struct Reference

```
#include <inc/dtTerrain/imageutils.h>
```

Public Attributes

- `std::string mFileName`
- `double mGeoTransform [6]`
- `dtCore::RefPtr< osg::Image > mImage`
- `double mInverseGeoTransform [6]`
- `int mMaxLatitude`
- `int mMaxLongitude`
- `int mMinLatitude`
- `int mMinLongitude`

4.8.1 Member Data Documentation

4.8.1.1 `std::string mFileName`

4.8.1.2 `double mGeoTransform[6]`

4.8.1.3 `dtCore::RefPtr< osg::Image > mImage`

4.8.1.4 `double mInverseGeoTransform[6]`

4.8.1.5 `int mMaxLatitude`

4.8.1.6 `int mMaxLongitude`

4.8.1.7 `int mMinLatitude`

4.8.1.8 `int mMinLongitude`

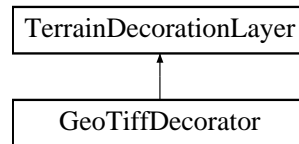
The documentation for this struct was generated from the following file:

- `imageutils.h`

4.9 GeoTiffDecorator Class Reference

This class is a terrain texturing decorator that supports the mapping of geospecific images onto terrain tiles.

#include <inc/dtTerrain/geotiffdecorator.h> Inheritance diagram for GeoTiffDecorator:



Public Member Functions

- **GeoTiffDecorator** (const std::string &name="GeoTiffDecorator")
Constructs the decoration layer.
- void **AddGeoSpecificImage** (const std::string &fileName)
Adds a new geo tiff image to the decorator.
- osg::Image * **CalculateBaseImage** (int lat, int lon)
Calculates an image based on the curret geospecific image set.
- virtual osg::Node * **GetOSGNode** ()
Since this decorator does not add any geometry to the terrain, just return NULL here.
- void **LoadAllGeoSpecificImages** ()
Loads any images in the image list.
- virtual void **OnLoadTerrainTile** (**PagedTerrainTile** &tile)
Based on the currently registered geo tiff images, this method will generate a base texture and assign it to the tile.
- void **SetResultingImageDimensions** (unsigned int w, unsigned int h)
Sets the image dimensions for the resulting terrain tile base texture.

Protected Member Functions

- virtual ~**GeoTiffDecorator** ()

4.9.1 Detailed Description

This class is a terrain texturing decorator that supports the mapping of geospecific images onto terrain tiles. A user may add as many geospecific images as they wish and the appropriate image region will be loaded and assigned to the terrain tile as needed.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 **GeoTiffDecorator** (const std::string & *name* = "GeoTiffDecorator")

Constructs the decoration layer.

4.9.2.2 virtual ~**GeoTiffDecorator** () [inline, protected, virtual]

4.9.3 Member Function Documentation

4.9.3.1 void **AddGeoSpecificImage** (const std::string & *fileName*) [inline]

Adds a new geo tiff image to the decorator.

4.9.3.2 `osg::Image * CalculateBaseImage (int lat, int lon)`

Calculates an image based on the current geospecific image set. Parameters

lat The latitude to build an image for.

lon The longitude to build an image for.

4.9.3.3 `virtual osg::Node* GetOSGNode () [inline, virtual]`

Since this decorator does not add any geometry to the terrain, just return NULL here.

Implements **TerrainDecorationLayer** (p. 91).

4.9.3.4 `void LoadAllGeoSpecificImages ()`

Loads any images in the image list.

4.9.3.5 `void OnLoadTerrainTile (PagedTerrainTile & tile) [virtual]`

Based on the currently registered geo tiff images, this method will generate a base texture and assign it to the tile.

Implements **TerrainDecorationLayer** (p. 91).

4.9.3.6 `void SetResultingImageDimensions (unsigned int w, unsigned int h) [inline]`

Sets the image dimensions for the resulting terrain tile base texture.

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

- **geotiffdecorator.h**
- **geotiffdecorator.cpp**

4.10 HeightColorMap Struct Reference

Maps height values to colors with interpolation/extrapolation.

```
#include <inc/dtTerrain/imageutils.h>
```

Inherits std::map< float, osg::Vec3 >.

Public Member Functions

- `osg::Vec3 GetColor` (float height) const
Gets the color corresponding to the specified height.

4.10.1 Detailed Description

Maps height values to colors with interpolation/extrapolation.

4.10.2 Member Function Documentation

4.10.2.1 `osg::Vec3 GetColor` (float *height*) const

Gets the color corresponding to the specified height. Parameters

height the height value to map

Returns the corresponding color

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

- `imageutils.h`
- `imageutils.cpp`

4.11 HeightField Class Reference

This class wraps an array of elevation posts.

```
#include <inc/dtTerrain/heightfield.h>
```

Public Member Functions

- **HeightField** (unsigned int numCols, unsigned int numRows)
Constructs the heightfield while at the same time allocated memory to store data of the desired number of rows and columns.
- **HeightField** ()
Constructs the heightfield.
- void **Allocate** (unsigned int numCols, unsigned int numRows)
Allocates enough memory to store data of the desired number of rows and columns.
- void **ConvertFromImage** (const osg::Image &image)
Converts the specified image into a valid heightfield.
- void **ConvertFromRaw** (unsigned int numColumns, unsigned int numRows, short *heightData)
Copies a chunk of raw data values into this heightfield.
- osg::Image * **ConvertToImage** () const
Converts this heightfield into a valid Image.
- short **GetHeight** (unsigned int c, unsigned int r) const
Gets the height stored at the given row and column.
- const short * **GetHeightFieldData** () const
Gets a pointer directly to the array of height values.
- float **GetInterpolatedHeight** (float x, float y) const
Gets a bi-linearly interpolated height value from the specified height field.
- unsigned int **GetNumColumns** () const
Gets the number of columns in this heightfield.
- unsigned int **GetNumRows** () const
Gets the number of rows in this heightfield.
- float **GetXInterval** () const
- float **GetYInterval** () const
- void **SetHeight** (unsigned int c, unsigned int r, short newHeight)
Sets the height stored at the given row and column.
- void **SetXInterval** (float interval)
- void **SetYInterval** (float interval)

Protected Member Functions

- virtual ~**HeightField** ()
Destroys the memory in use by the heightfield.

4.11.1 Detailed Description

This class wraps an array of elevation posts. The elevation data is 16-bit short values, therefore, the elevation can be a maximum of SHRT_MAX (32767) and a minimum of SHRT_MIN (-32768) with zero equaling "flat" or at sea-level. This range should satisfy the needs of most applications. For example, the highest peak in the world is located on Mount Everest which sits at 8850 meters or 29,035 feet.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 HeightField ()

Constructs the heightfield. Note, the data is invalid at this point until it gets allocated. See also **Allocate** (p. 31)

4.11.2.2 HeightField (unsigned int *numCols*, unsigned int *numRows*)

Constructs the heightfield while at the same time allocated memory to store data of the desired number of rows and columns. Parameters

numCols Number of columns stored in the heightfield.

numRows Number of rows stored in the heightfield.

See also **Allocate** (p. 31)

4.11.2.3 ~HeightField () [protected, virtual]

Destroys the memory in use by the heightfield.

4.11.3 Member Function Documentation

4.11.3.1 void Allocate (unsigned int *numCols*, unsigned int *numRows*)

Allocates enough memory to store data of the desired number of rows and columns. Any existing data is destroyed before allocating room for the new data. Parameters

numCols Number of columns stored in the heightfield.

numRows Number of rows stored in the heightfield.

Exceptions

HeightFieldException::OUT_OF_BOUNDS (p. 33) if either the number of rows or columns is equal to zero.

4.11.3.2 void ConvertFromImage (const osg::Image & *image*)

Converts the specified image into a valid heightfield. Parameters

image The image to convert.

Note The image should be a 24-bit image representing a 16-bit heightfield. Internally, this is converted into 16-bit shorts which are inserted into this heightfield.

4.11.3.3 void ConvertFromRaw (unsigned int *numColumns*, unsigned int *numRows*, short * *heightData*)

Copies a chunk of raw data values into this heightfield. Space for the data is automatically allocated so there is no need to call **Allocate()** (p. 31) on the heightfield first. Parameters

numColumns Number of columns in the data.

numRows Number of rows in the data.

heightData Pointer to the actual data values. If this is NULL, no action is taken.

4.11.3.4 osg::Image * ConvertToImage () const

Converts this heightfield into a valid Image. Returns A 16-bit (565) image.

4.11.3.5 short GetHeight (unsigned int c, unsigned int r) const

Gets the height stored at the given row and column. Parameters

c Column in the heightfield.

r Row in the heightfield.

Exceptions

HeightFieldException::OUT_OF_BOUNDS (p. 33) if *c* or *r* is greater than the number of rows or columns in the heightfield.

4.11.3.6 const short* GetHeightFieldData () const [inline]

Gets a pointer directly to the array of height values. Returns The height values array. Note, this cannot be modified or destroyed.

4.11.3.7 float GetInterpolatedHeight (float x, float y) const

Gets a bi-linearly interpolated height value from the specified height field.

4.11.3.8 unsigned int GetNumColumns () const [inline]

Gets the number of columns in this heightfield. Returns The current number of columns or zero if the data is invalid.

4.11.3.9 unsigned int GetNumRows () const [inline]

Gets the number of rows in this heightfield. Returns The current number of rows or zero if the data is invalid.

4.11.3.10 float GetXInterval () const [inline]

4.11.3.11 float GetYInterval () const [inline]

4.11.3.12 void SetHeight (unsigned int c, unsigned int r, short newHeight)

Sets the height stored at the given row and column. Parameters

c Column in the heightfield.

r Row in the heightfield.

Exceptions

HeightFieldException::OUT_OF_BOUNDS (p. 33) if *c* or *r* is greater than the number of rows or columns in the heightfield.

4.11.3.13 void SetXInterval (float interval) [inline]

4.11.3.14 void SetYInterval (float interval) [inline]

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

- **heightfield.h**
- **heightfield.cpp**

4.12 HeightFieldException Class Reference

This enumeration defines the different types of exceptions the height field may throw if an error occurs.

```
#include <inc/dtTerrain/heightfield.h>
```

Static Public Attributes

- static **HeightFieldException INVALID_HEIGHTFIELD**
Thrown if the heightfield data has not been allocated before it is used.
- static **HeightFieldException INVALID_IMAGE_FORMAT**
Thrown when an invalid image format is specified when converting it to a heightfield.
- static **HeightFieldException OUT_OF_BOUNDS**
Thrown when an operation occurs on a heightfield with dimensions smaller or larger than the heightfield has room for.

4.12.1 Detailed Description

This enumeration defines the different types of exceptions the height field may throw if an error occurs.

4.12.2 Member Data Documentation

4.12.2.1 HeightFieldException INVALID_HEIGHTFIELD [static]

Thrown if the heightfield data has not been allocated before it is used.

4.12.2.2 HeightFieldException INVALID_IMAGE_FORMAT [static]

Thrown when an invalid image format is specified when converting it to a heightfield.

4.12.2.3 HeightFieldException OUT_OF_BOUNDS [static]

Thrown when an operation occurs on a heightfield with dimensions smaller or larger than the heightfield has room for.

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

- **heightfield.h**
- **heightfield.cpp**

4.13 HeightFieldResizePolicy Class Reference

This enumeration specifies the valid resize policies the data readers should follow when loading a new heightfield.

```
#include <inc/dtTerrain/terraindatareader.h>
```

Static Public Attributes

- static const **HeightFieldResizePolicy NEAREST_POWER_OF_TWO**
Data should be resized such that it is a power of two in all dimensions closests to its original size.
- static const **HeightFieldResizePolicy NEAREST_POWER_OF_TWO_PLUS_ONE**
Data should be resized such that it is a power of two plus one in all dimensions closests to its original size.
- static const **HeightFieldResizePolicy NONE**
No resizing should occur.

Protected Member Functions

- **HeightFieldResizePolicy** (const std::string &name)

4.13.1 Detailed Description

This enumeration specifies the valid resize policies the data readers should follow when loading a new heightfield. This is to ensure that the data formats supported by the reader are imported in a manor suitable to the current renderer.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 HeightFieldResizePolicy (const std::string & name) [inline, protected]

4.13.3 Member Data Documentation

4.13.3.1 const HeightFieldResizePolicy NEAREST_POWER_OF_TWO [static]

Data should be resized such that it is a power of two in all dimensions closests to its original size. (2^n)

4.13.3.2 const HeightFieldResizePolicy NEAREST_POWER_OF_TWO_PLUS_ONE [static]

Data should be resized such that it is a power of two plus one in all dimensions closests to its original size. ($2^n + 1$)

4.13.3.3 const HeightFieldResizePolicy NONE [static]

No resizing should occur.

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

- **terraindatareader.h**
- **terraindatareader.cpp**

4.14 ImageUtilException Class Reference

Defines exceptions that may be thrown from the image utility methods.

```
#include <inc/dtTerrain/imageutils.h>
```

Static Public Attributes

- static **ImageUtilException INVALID_IMAGE_DIMENSIONS**
- static **ImageUtilException INVALID_RASTER_FORMAT**
Thrown if the geo-specific image is of the wrong format.
- static **ImageUtilException LOAD_FAILED**
Thrown if the analyzer could not load files needed for processing.

Protected Member Functions

- **ImageUtilException** (const std::string &name)

4.14.1 Detailed Description

Defines exceptions that may be thrown from the image utility methods.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 **ImageUtilException** (const std::string & *name*) [**inline**, **protected**]

4.14.3 Member Data Documentation

4.14.3.1 **ImageUtilException INVALID_IMAGE_DIMENSIONS** [**static**]

4.14.3.2 **ImageUtilException INVALID_RASTER_FORMAT** [**static**]

Thrown if the geo-specific image is of the wrong format.

4.14.3.3 **ImageUtilException LOAD_FAILED** [**static**]

Thrown if the analyzer could not load files needed for processing.

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

- **imageutils.h**
- **imageutils.cpp**

4.15 ImageUtils Class Reference

This class is a static class containing many methods for creating images procedurally, concatenating images, and building images from source height data.

```
#include <inc/dtTerrain/imageutils.h>
```

Classes

- struct **GeospecificImage**
- struct **HeightColorMap**
Maps height values to colors with interpolation/extrapolation.

Static Public Member Functions

- static dtCore::RefPtr< osg::Image > **ApplyMask** (const osg::Image &src_image, const osg::Image &mask_image)
Use an image to mask-out vegetation probability (set probability to 0%) Useful for masking-out bodies of water or user-created urban models.
- static unsigned short **CalculateDetailNoise** (int x, int y)
Calculates a noise value for a given x,y pair.
- static unsigned short **CalculateScaleMapNoise** (int x, int y)
- static dtCore::RefPtr< osg::Image > **CreateBaseGradientMap** (const **HeightField** &hf, float scale)
Generates a gradient map based on the given heightfield.
- static dtCore::RefPtr< osg::Image > **CreateDetailGradientMap** (unsigned int width, unsigned int height, float scale)
Generates a gradient map based on perlin noise.
- static dtCore::RefPtr< osg::Image > **CreateDetailMap** (unsigned int width, unsigned int height)
- static dtCore::RefPtr< osg::Image > **CreateDetailScaleMap** (unsigned int width, unsigned int height)
Creates a perlin noise map which is useful for adding perturbing existing noise values in a random fashion.
- static dtCore::RefPtr< osg::Image > **EnsurePow2Image** (const osg::Image *srcImage)
Nearest neighbor geometric image manipulation.
- static void **ImageStats** (const osg::Image *image, std::string *imagename)
Logs information about the given image.
- static void **LoadGeospecificLCCImage** (**ImageUtils::GeospecificImage** &gslcc)
- static dtCore::RefPtr< osg::Image > **MakeBaseColor** (const **HeightField** &hf, const **ImageUtils::HeightColorMap** &upperHeightColorMap, const **ImageUtils::HeightColorMap** &lowerHeightColorMap, float gamma=1.0f)
Generates an image which maps the height value in the heightfield to a color value given by one of the two color maps.
- static dtCore::RefPtr< osg::Image > **MakeFilteredImage** (const osg::Image &src_image, const osg::Vec3 &rgb_selected)
Create smoothed grayscale map of the terrain by LCC type Uses weighted next nearest neighbor to "fuzzy"-up the LCCImage data.
- static dtCore::RefPtr< osg::Image > **MakeRelativeElevationImage** (const **HeightField** &hf, float scale)
Creates a relative elevation map.
- static dtCore::RefPtr< osg::Image > **MakeSlopeAspectImage** (const **HeightField** &hf)
Create a slope map.

4.15.1 Detailed Description

This class is a static class containing many methods for creating images procedurally, concatenating images, and building images from source height data.

4.15.2 Member Function Documentation

4.15.2.1 dtCore::RefPtr< osg::Image > ApplyMask (const osg::Image & *src_image*, const osg::Image & *mask_image*) [static]

Use an image to mask-out vegetation probability (set probability to 0%) Useful for masking-out bodies of water or user-created urban models. Parameters

src_image the black/white LCC Image by LCC type

mask_image the black/white - using LCC type 11 as default

Returns the modified filtered image

4.15.2.2 unsigned short CalculateDetailNoise (int *x*, int *y*) [static]

Calculates a noise value for a given x,y pair. Parameters

x The x value.

y The y value.

4.15.2.3 unsigned short CalculateScaleMapNoise (int *x*, int *y*) [static]

4.15.2.4 dtCore::RefPtr< osg::Image > CreateBaseGradientMap (const HeightField & *hf*, float *scale*) [static]

Generates a gradient map based on the given heightfield. Parameters

hf The source heightfield.

scale The scale value to apply to the resulting gradient calculations.

Returns An RGB image encoded with gradient values cooresponding to the specified heightfield.

4.15.2.5 dtCore::RefPtr< osg::Image > CreateDetailGradientMap (unsigned int *width*, unsigned int *height*, float *scale*) [static]

Generates a gradient map based on perlin noise. This can be used to procedurally add lighting detail to surfaces. Parameters

width The width of the image.

height The height of the image.

float Scale value with with to scale the resulting gradient values.

Returns Image containing the gradient map.

4.15.2.6 dtCore::RefPtr< osg::Image > CreateDetailMap (unsigned int *width*, unsigned int *height*) [static]

Parameters

width The target width of the created image.

eight The target height of the created image.

4.15.2.7 dtCore::RefPtr< osg::Image > CreateDetailScaleMap (unsigned int *width*, unsigned int *height*) [static]

Creates a perlin noise map which is useful for adding perturbing existing noise values in a random fashion. Parameters

width Width of the image.

height Height of the image.

Returns An image containing perlin noise based scale values.

4.15.2.8 dtCore::RefPtr< osg::Image > EnsurePow2Image (const osg::Image * *srcImage*) [static]

Nearest neighbor geometric image manipulation. This assumes that the original image is nxn square. Parameters

srcImage The image to scale.

Returns Destination image with the correct power of 2 dimensions.

4.15.2.9 void ImageStats (const osg::Image * *image*, std::string * *imagename*) [static]

Logs information about the given image. (LogLevel is INFO). Parameters

image the handle to the image you want to examine

imagename a descriptive name to call this image

4.15.2.10 void LoadGeospecificLCCImage (ImageUtils::GeospecificImage & *gs/lcc*) [static]

4.15.2.11 dtCore::RefPtr< osg::Image > MakeBaseColor (const HeightField & *hf*, const ImageUtils::HeightColorMap & *upperHeightColorMap*, const ImageUtils::HeightColorMap & *lowerHeightColorMap*, float *gamma* = 1.0f) [static]

Generates an image which maps the height value in the heightfield to a color value given by one of the two color maps. Parameters

hf The heightfield with which to map colors.

upperHeightColorMap The color values to map the height values which are above 0 (sea-level).

lowerHeightColorMap The color values to map the height values which are are below or equal to zero.

gamma Gamma correction value. Greater than 1.0 to brighten, less than 1.0 to darken.

4.15.2.12 dtCore::RefPtr< osg::Image > MakeFilteredImage (const osg::Image & *src_image*, const osg::Vec3 & *rgb_selected*) [static]

Create smoothed grayscale map of the terrain by LCC type Uses weighted next nearest neighbor to "fuzzy"-up the LCCImage data. Parameters

src_image the black/white LCC Image by LCC type

rgb_selected the RGB color of the points to smooth (always 0,0,0 - black)

Returns the newly created image

4.15.2.13 dtCore::RefPtr< osg::Image > MakeRelativeElevationImage (const HeightField & *hf*, float *scale*) [static]

Creates a relative elevation map. Parameters

hf The heightfield with which to build the elevation map.

scale The amount to scale the resulting values by.

Returns the newly created image

4.15.2.14 dtCore::RefPtr< osg::Image > MakeSlopeAspectImage (const HeightField & hf) [static]

Create a slope map. Parameters

hf The heightfield heightfield

Returns the newly created image

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

- **imageutils.h**
- **imageutils.cpp**

4.16 Index Struct Reference

This structure represents a location on the terrain's 2D grid.

```
#include <inc/dtTerrain/soarxdrawable.h>
```

Public Member Functions

- **Index** (const **Index** &i, const **Index** &j, unsigned int parity, bool direction)
- **Index** (int qValue, int xValue, int yValue)
- **Index** (int xValue, int yValue)
Simple named constructor.
- **Index** ()
Simply sets all values to zero.
- void **clamp** (int value)
Clamps the x,y components of this index between zero and the value specified.
- void **operator&=** (int value)
Performs a bit-wise AND on the x,y components of this index.
- void **operator+=** (const **Index** &rhs)
In place addition of the x,y components of this index with the x,y components of the other.
- void **operator-=** (const **Index** &rhs)
In place subtraction of the x,y components of this index with the x,y components of the other.
- bool **operator<** (int value)
*Less-than comparison on this **Index** (p. 40) with another value.*
- void **operator<<=** (int bits)
Left shifts both x and y components of this index by the specified number of bits.
- bool **operator>** (int value)
*Greater-than comparison on this **Index** (p. 40) with another value.*
- void **operator>>=** (int bits)
Right shifts both x and y components of this index by the specified number of bits.

Public Attributes

- unsigned int **q**
- int **x**
- int **y**

4.16.1 Detailed Description

This structure represents a location on the terrain's 2D grid.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 **Index** () [inline]

Simply sets all values to zero.

4.16.2.2 **Index** (int xValue, int yValue) [inline]

Simple named constructor.

4.16.2.3 `Index (int qValue, int xValue, int yValue) [inline]`

4.16.2.4 `Index (const Index & i, const Index & j, unsigned int parity, bool direction)`

4.16.3 Member Function Documentation

4.16.3.1 `void clamp (int value) [inline]`

Clamps the x,y components of this index between zero and the value specified. Parameters

The max value to clamp this index to.

4.16.3.2 `void operator&= (int value) [inline]`

Performs a bit-wise AND on the x,y components of this index. Parameters

value The value to bit-wise AND with this index.

4.16.3.3 `void operator+= (const Index & rhs) [inline]`

In place addition of the x,y components of this index with the x,y components of the other. Parameters

rhs The index to add to this one.

4.16.3.4 `void operator-= (const Index & rhs) [inline]`

In place subtraction of the x,y components of this index with the x,y components of the other. Parameters

rhs The index to subtract from this one.

4.16.3.5 `bool operator< (int value) [inline]`

Less-than comparison on this **Index** (p. 40) with another value. Returns True if both x,y components of this index are less than the specified value.

4.16.3.6 `void operator<<= (int bits) [inline]`

Left shifts both x and y components of this index by the specified number of bits. Parameters

value Number of bits to shift the components by.

4.16.3.7 `bool operator> (int value) [inline]`

Greater-than comparison on this **Index** (p. 40) with another value. Returns True if both x,y components of this index are greater than the specified value.

4.16.3.8 `void operator>>= (int bits) [inline]`

Right shifts both x and y components of this index by the specified number of bits. Parameters

value Number of bits to shift the components by.

4.16.4 Member Data Documentation

4.16.4.1 `unsigned int q`

4.16.4.2 `int x`

4.16.4.3 `int y`

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

- `soarxdrawable.h`
- `soarxdrawable.cpp`

4.17 LCCAnalyzer Class Reference

The LCC Analyzer calculates LCC data for use in the vegetation decorator.

```
#include <inc/dtTerrain/lccanalyzer.h>
```

Public Member Functions

- **LCCAnalyzer ()**
Constructor.
- virtual **~LCCAnalyzer ()**
Destructor.
- void **AddGeospecificImage** (const std::string &fileName)
Adds a geospecific image to the LCC analyzers list of geo images.
- bool **CheckBaseLCCImages** (const **HeightField** &hf, int latitude, int longitude, const std::string &tileCachePath)
- void **CheckSlopeAndElevationMaps** (const **HeightField** &hf, const std::string &tileCachePath)
- void **Clear ()**
Clears all data cached in this analyzer.
- void **ComputeProbabilityMap** (const **HeightField** &hf, **LCCType** &type, int latitude, int longitude, const std::string &tileCachePath)
- dtCore::RefPtr< osg::Image > **GenerateBaseFilterImage** (**LCCType** &type, const std::string &tileCachePath)
- unsigned int **GetImageSize ()** const
Gets the current maximum image size used during LCC analysis.
- bool **IsGeoSpecificLCCImageLoaded** (const std::string &fileName)
Checks the list of currently loaded images to see if the specified image has already been loaded and processed.
- void **LCCHistogram** (const osg::Image &LCCbase, const osg::Image &image, const std::string &fileName, int binsize)
Buggy "histogram" of an image by a particular LCC type.
- void **LoadAllGeoSpecificImages ()**
Makes sure all the geospecific images that have been added to the analyzer are loaded and ready for use in the LCC data processing.
- dtCore::RefPtr< osg::Image > **MakeBaseLCCColor** (const **HeightField** &hf, int latitude, int longitude)
Makes the base color texture map for the specified heightfield.
- dtCore::RefPtr< osg::Image > **MakeCombinedImage** (dtTerrain::LCCType &l, const **HeightField** &hf, const osg::Image &f_image, const osg::Image &s_image, const osg::Image &r_image)
Create probability map of the likelihood for a particular LCC type.
- dtCore::RefPtr< osg::Image > **MakeLCCMask** (const osg::Image &src_image, unsigned char r, unsigned char g, unsigned char b)
Creates a mask image based on this LCC type where a hit corresponds to a black pixel and white corresponds to a miss.
- bool **OutputDebugImages ()**
Gets whether or not inter-process images processed during LCC analysis are saved for review.
- bool **ProcessLCCData** (const **PagedTerrainTile** &tile, **LCCType** &type)
- void **SetMaxImageSize** (unsigned int newSize)

Sets the maximum size of the images constructed during LCC analysis.

- void **SetOutputDebugImages** (bool value)
Sets whether or not inter-process images processed during LCC analysis are saved for review.

4.17.1 Detailed Description

The LCC Analyzer calculates LCC data for use in the vegetation decorator.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 LCCAnalyzer ()

Constructor.

4.17.2.2 ~LCCAnalyzer () [virtual]

Destructor.

4.17.3 Member Function Documentation

4.17.3.1 void AddGeospecificImage (const std::string & fileName) [inline]

Adds a geospecific image to the LCC analyzers list of geo images. There must be at least one valid LCC geospecific image in this list, else an exception will be thrown while processing LCC types. Parameters

fileName The name of the geospecific resource.

4.17.3.2 bool CheckBaseLCCImages (const HeightField & hf, int latitude, int longitude, const std::string & tileCachePath)

4.17.3.3 void CheckSlopeAndElevationMaps (const HeightField & hf, const std::string & tileCachePath)

4.17.3.4 void Clear () [inline]

Clears all data cached in this analyzer. This should be called before processing a different tile or region.

4.17.3.5 void ComputeProbabilityMap (const HeightField & hf, LCCType & type, int latitude, int longitude, const std::string & tileCachePath)

4.17.3.6 dtCore::RefPtr< osg::Image > GenerateBaseFilterImage (LCCType & type, const std::string & tileCachePath)

4.17.3.7 unsigned int GetImageSize () const [inline]

Gets the current maximum image size used during LCC analysis. Returns The current image size.

4.17.3.8 bool IsGeoSpecificLCCImageLoaded (const std::string & fileName)

Checks the list of currently loaded images to see if the specified image has already been loaded and processed. Parameters

fileName The filename of the image.

Returns True if already loaded, false otherwise.

4.17.3.9 void LCCHistogram (const osg::Image & LCCbase, const osg::Image & image, const std::string & fileName, int binsize)

Buggy "histogram" of an image by a particular LCC type. Parameters

LCCbase the black/white LCC image of picked points of a particular LCC type

image the slopemap, heightmap, or relative elevation

fileName the filename to save the histogram data

binsize the sampling size of the image (i.e. the delta height or slope).

4.17.3.10 void LoadAllGeoSpecificImages ()

Makes sure all the geospecific images that have been added to the analyzer are loaded and ready for use in the LCC data processing. Note If an image has already been loaded previously, it will not be reloaded.

4.17.3.11 dtCore::RefPtr< osg::Image > MakeBaseLCCColor (const HeightField & hf, int latitude, int longitude)

Makes the base color texture map for the specified heightfield. Parameters

hf the heightfield to process

latitude the latitude of the terrain segment

longitude the longitude of the terrain segment

Returns the newly created image

4.17.3.12 dtCore::RefPtr< osg::Image > MakeCombinedImage (dtTerrain::LCCType & l, const HeightField & hf, const osg::Image & f_image, const osg::Image & s_image, const osg::Image & r_image)

Create probability map of the likelihood for a particular LCC type. Parameters

l The LCC type to calculate probabilities for.

hf The heightfield.

f_image The masked LCC image containing placement statistics.

s_image the slopemap image

r_image the relative elevation image

Returns the newly created image

4.17.3.13 dtCore::RefPtr< osg::Image > MakeLCCMask (const osg::Image & src_image, unsigned char r, unsigned char g, unsigned char b)

Creates a mask image based on this LCC type where a hit corresponds to a black pixel and white corresponds to a miss. Parameters

src_image Image containing LCC coloration data.

r Red pixel color value with which to create the mask.

g Green pixel color value with which to create the mask.

b Blue pixel color value with which to create the mask.

Returns The mask image.

4.17.3.14 bool OutputDebugImages () [inline]

Gets whether or not inter-process images processed during LCC analysis are saved for review. Returns True if images are saved, false otherwise.

4.17.3.15 bool ProcessLCCData (const PagedTerrainTile & tile, LCCType & type)**4.17.3.16 void SetMaxImageSize (unsigned int newSize) [inline]**

Sets the maximum size of the images constructed during LCC analysis. Parameters

newSize The size in pixels of the image.

Note The default is 1024. Increasing this value will result in more accurate vegetation computations; however, will dramatically increase the time taken to generate the necessary data.

4.17.3.17 void SetOutputDebugImages (bool *value*) [inline]

Sets whether or not inter-process images processed during LCC analysis are saved for review. Parameters

value True if images should be saved, false otherwise.

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

- **lccanalyzer.h**
- **lccanalyzer.cpp**

4.18 LCCAnalyzerException Class Reference

Defines the exceptions thrown by the lcc analyzer.

```
#include <inc/dtTerrain/lccanalyzer.h>
```

Static Public Attributes

- static **LCCAnalyzerException INVALID_CACHE**
Thrown if the user failed to enable caching.
- static **LCCAnalyzerException NO_VALID_GEO_IMAGES**
Thrown if LCC analyzing occurs before valid geographical images are assigned to the analyzer.

Protected Member Functions

- **LCCAnalyzerException** (const std::string &name)

4.18.1 Detailed Description

Defines the exceptions thrown by the lcc analyzer.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 LCCAnalyzerException (const std::string & name) [inline, protected]

4.18.3 Member Data Documentation

4.18.3.1 LCCAnalyzerException INVALID_CACHE [static]

Thrown if the user failed to enable caching. Currently terrain data caching must be enabled for the analyzer to work properly.

4.18.3.2 LCCAnalyzerException NO_VALID_GEO_IMAGES [static]

Thrown if LCC analyzing occurs before valid geographical images are assigned to the analyzer.

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

- **lccanalyzer.h**
- **lccanalyzer.cpp**

4.19 LCCAnalyzerResourceName Class Reference

This enumeration identifies the resources that are cached and managed by the vegetation decorator layer.

```
#include <inc/dtTerrain/lccanalyzer.h>
```

Static Public Attributes

- static const **LCCAnalyzerResourceName** **BASE_COLOR**
- static const **LCCAnalyzerResourceName** **BASE_FILTER_NAME**
- static const **LCCAnalyzerResourceName** **BASE_LCC_COLOR**
- static const **LCCAnalyzerResourceName** **COMPOSITE_LCC_IMAGE**
- static const **LCCAnalyzerResourceName** **IMAGE_EXT**
- static const **LCCAnalyzerResourceName** **LCC_IMAGE_NAME**
- static const **LCCAnalyzerResourceName** **REL_ELEV_IMAGE**
- static const **LCCAnalyzerResourceName** **SCENE_GRAPH**
- static const **LCCAnalyzerResourceName** **SLOPE_IMAGE**
- static const **LCCAnalyzerResourceName** **WATER_MASK**

Protected Member Functions

- **LCCAnalyzerResourceName** (const std::string &name)

4.19.1 Detailed Description

This enumeration identifies the resources that are cached and managed by the vegetation decorator layer.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 **LCCAnalyzerResourceName** (const std::string & *name*) [inline, protected]

4.19.3 Member Data Documentation

4.19.3.1 const **LCCAnalyzerResourceName** **BASE_COLOR** [static]

4.19.3.2 const **LCCAnalyzerResourceName** **BASE_FILTER_NAME** [static]

4.19.3.3 const **LCCAnalyzerResourceName** **BASE_LCC_COLOR** [static]

4.19.3.4 const **LCCAnalyzerResourceName** **COMPOSITE_LCC_IMAGE** [static]

4.19.3.5 const **LCCAnalyzerResourceName** **IMAGE_EXT** [static]

4.19.3.6 const **LCCAnalyzerResourceName** **LCC_IMAGE_NAME** [static]

4.19.3.7 const **LCCAnalyzerResourceName** **REL_ELEV_IMAGE** [static]

4.19.3.8 const **LCCAnalyzerResourceName** **SCENE_GRAPH** [static]

4.19.3.9 const **LCCAnalyzerResourceName** **SLOPE_IMAGE** [static]

4.19.3.10 const **LCCAnalyzerResourceName** **WATER_MASK** [static]

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

- **lccanalyzer.h**
- **lccanalyzer.cpp**

4.20 LCCModel Struct Reference

Describes an LCC's geometric model info.

```
#include <inc/dtTerrain/lcctype.h>
```

Public Member Functions

- **LCCModel ()**

Public Attributes

- std::string **name**
- float **scale**
- dtCore::RefPtr< osg::Node > **sceneNode**

4.20.1 Detailed Description

Describes an LCC's geometric model info.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 LCCModel () [inline]

4.20.3 Member Data Documentation

4.20.3.1 std::string name

4.20.3.2 float scale

4.20.3.3 dtCore::RefPtr<osg::Node> sceneNode

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

- **lcctype.h**

4.21 LCCType Class Reference

```
#include <inc/dtTerrain/lcctype.h>
```

Classes

- struct **LCCModel**
Describes an LCC's geometric model info.

Public Member Functions

- **LCCType** (unsigned int index, const std::string &name)
Constructor.
- **~LCCType** ()
Destructor.
- void **AddModel** (const std::string &name, float scale=1.0)
This adds a model to the list of available models for this vegetation type.
- void **ClearModels** ()
Removes all the LCC models from this LCC type.
- float **GetAspect** () const
- int **GetIndex** () const
This returns the index of the lcc type.
- const std::string & **GetLCCName** () const
Gets the LCC name of this type.
- float **GetMaxElevation** () const
- float **GetMaxSlope** () const
- float **GetMinElevation** () const
- float **GetMinSlope** () const
- **LCCModel** * **GetModel** (const std::string &name)
Looks up an LCC model on this type by name.
- **LCCModel** * **GetModel** (unsigned int index)
Retrives the model at the given index.
- unsigned int **GetNumberOfModels** () const
Gets the number of models registered under this LCC type.
- void **GetRGB** (unsigned char &red, unsigned char &green, unsigned char &blue)
Gets the RGB color identifier for this type.
- void **RemoveModel** (const std::string &name)
Removes the LCC model from this LCC type's list of models.
- void **SetAspect** (int aspect)
- void **SetElevation** (float min, float max, float sharpness)
- void **SetRelativeElevation** (float min, float max, float sharpness)
- void **SetRGB** (unsigned char red, unsigned char green, unsigned char blue)
Sets the RGB color value of this type.
- void **SetSlope** (float min, float max, float sharpness)

4.21.1 Constructor & Destructor Documentation

4.21.1.1 LCCType (unsigned int *index*, const std::string & *name*)

Constructor.

4.21.1.2 ~LCCType () [inline]

Destructor.

4.21.2 Member Function Documentation

4.21.2.1 void AddModel (const std::string & *name*, float *scale* = 1.0)

This adds a model to the list of available models for this vegetation type. When placing vegetation of this LCC type, a model is randomly chosen from the model list. Parameters

name File name of the model.

scale Uniform scaling factor for this model.

Note The model is loaded when vegetation is placed.

4.21.2.2 void ClearModels () [inline]

Removes all the LCC models from this LCC type.

4.21.2.3 float GetAspect () const [inline]

Returns aspect

4.21.2.4 int GetIndex () const [inline]

This returns the index of the lcc type. Returns index of the lcc type as defined in lcc configuration data

4.21.2.5 const std::string& GetLCCName () const [inline]

Gets the LCC name of this type. Returns the name of the LCC type, such as snow, water, evergreen, snow.

4.21.2.6 float GetMaxElevation () const [inline]

Returns maximum elevation

4.21.2.7 float GetMaxSlope () const [inline]

Returns maximum slope

4.21.2.8 float GetMinElevation () const [inline]

Returns minimum elevation

4.21.2.9 float GetMinSlope () const [inline]

Returns minimum slope

4.21.2.10 LCCType::LCCModel * GetModel (const std::string & *name*)

Looks up an LCC model on this type by name. Parameters

name The name of the LCC model to retrieve.

Returns A valid model or NULL if the model was not found.

4.21.2.11 LCCType::LCCModel * GetModel (unsigned int *index*)

Retrieves the model at the given index. Parameters

index An index into the list of models.

Returns If index is valid, a pointer to the model is returned, else this method will return NULL.

4.21.2.12 unsigned int GetNumberOfModels () const [inline]

Gets the number of models registered under this LCC type. Returns The number of models.

4.21.2.13 void GetRGB (unsigned char & *red*, unsigned char & *green*, unsigned char & *blue*) [inline]

Gets the RGB color identifier for this type.

4.21.2.14 void RemoveModel (const std::string & *name*)

Removes the LCC model from this LCC type's list of models. Parameters

The name of the model. If this name is not found in the list of models, this method does nothing.

4.21.2.15 void SetAspect (int *aspect*) [inline]

Parameters

aspect

4.21.2.16 void SetElevation (float *min*, float *max*, float *sharpness*) [inline]

Parameters

minimum elevation

maximum elevation

elevation sharpness

4.21.2.17 void SetRelativeElevation (float *min*, float *max*, float *sharpness*) [inline]

Parameters

minimum elevation

maximum elevation

relative elevation sharpness

4.21.2.18 void SetRGB (unsigned char *red*, unsigned char *green*, unsigned char *blue*) [inline]

Sets the RGB color value of this type. The color value maps the the color defined by the land classification cover image map.

4.21.2.19 void SetSlope (float *min*, float *max*, float *sharpness*) [inline]

Parameters

minimum slope

maximum slope

slope sharpness

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

- **lcctype.h**
- **lcctype.cpp**

4.22 MathUtils Class Reference

```
#include <inc/dtTerrain/mathutils.h>
```

Static Public Member Functions

- static float **F2F** (int a)
- static int **F2I** (int a)
- static int **Fade** (int t)
- static int **I2F** (int a)
Fixed point math functions.
- static int **ILerp** (int t, int a, int b)
- static int **IMul** (int a, int b)
- template<typename Real >
static Real **Lerp** (Real t, Real v1, Real v2)

4.22.1 Member Function Documentation

4.22.1.1 static float F2F (int a) [inline, static]

4.22.1.2 static int F2I (int a) [inline, static]

4.22.1.3 static int Fade (int t) [inline, static]

4.22.1.4 static int I2F (int a) [inline, static]

Fixed point math functions.

4.22.1.5 static int ILerp (int t, int a, int b) [inline, static]

4.22.1.6 static int IMul (int a, int b) [inline, static]

4.22.1.7 static Real Lerp (Real t, Real v1, Real v2) [inline, static]

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

- **mathutils.h**

4.23 PagedTerrainTile Class Reference

This class is the base class for a tile of data.

```
#include <inc/dtTerrain/pagedterraintile.h>
```

Public Member Functions

- `osg::Image * GetBaseTextureImage ()`
Gets the image currently being used by this tile as its base texture.
- `const std::string & GetCachePath () const`
Gets the cache path assigned to this tile.
- `const GeoCoordinates & GetGeoCoordinates () const`
Gets this tile's coordinate system.
- `const HeightField * GetHeightField () const`
Gets the heightfield currently assigned to this terrain tile.
- `Terrain * GetParentTerrain ()`
Gets the terrain that created this terrain tile and therefore owns it.
- `const Terrain * GetParentTerrain () const`
Gets the terrain that created this terrain tile and therefore owns it.
- `bool GetUpdateCache () const`
Gets whether or not this tile's cache is out of sync with its data.
- `bool IsCachingEnabled () const`
Gets the status of this tile's caching.
- `virtual void ReadFromCache ()`
This method provides the paged tile with an opportunity to restore any data that was previously cached.
- `void SetBaseTextureImage (osg::Image *image)`
Sets the base texture for this terrain tile.
- `void SetGeoCoordinates (const GeoCoordinates &cs)`
Sets the origin with which this tile maps to.
- `void SetHeightField (HeightField *hf)`
Sets the heightfield that maps to this terrain tile.
- `void SetUpdateCache (bool flag)`
Sets whether or not the cache needs to be updated to reflect the current state of the tile.
- `virtual void WriteToCache ()`
This method provides the paged tile with an opportunity to cache any data that it wants to retrieve later.

Protected Member Functions

- `PagedTerrainTile (Terrain *terrain)`
Constructs a terrain tile.
- `virtual ~PagedTerrainTile ()`

Destroys the terrain tile.

- void **SetCachePath** (const std::string &path)
Sets the cache path for this tile.

Friends

- class **PagedTerrainTileFactory**
Allow the factory and terrain to have access to this class.
- class **Terrain**

4.23.1 Detailed Description

This class is the base class for a tile of data. Instances of this class are the atomic data elements the terrain manages when navigating through its data. Paging and disk caching (if enabled) is handled transparently when needed by the terrain and it provides, by default, access to its heightfield, base texture, and other fundamental data structures. Although this base class has many features, application dependent data and behavior can still be added to this base tile through subclasses.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 PagedTerrainTile (Terrain * terrain) [protected]

Constructs a terrain tile. Note, the constrmUpdateCacheuctor is protected as to enforce the use of the terrain class in order to create a new tile.

4.23.2.2 ~PagedTerrainTile () [protected, virtual]

Destroys the terrain tile. Since most internal data is managed by smart pointers, this method does practically nothing.

4.23.3 Member Function Documentation

4.23.3.1 osg::Image* GetBaseTextureImage () [inline]

Gets the image currently being used by this tile as its base texture. Returns A pointer to the image or NULL if it is not valid.

4.23.3.2 const std::string& GetCachePath () const [inline]

Gets the cache path assigned to this tile. Returns The current cache path or the empty string if caching is disabled.

4.23.3.3 const GeoCoordinates& GetGeoCoordinates () const [inline]

Gets this tile's coordinate system. Returns Constant reference to this tile's coordinate system.

4.23.3.4 const HeightField* GetHeightField () const [inline]

Gets the heightfield currently assigned to this terrain tile. Returns A pointer to the heightfield or NULL if it is not valid.

4.23.3.5 Terrain* GetParentTerrain () [inline]

Gets the terrain that created this terrain tile and therefore owns it. Returns A pointer to the parent terrain.

4.23.3.6 const Terrain* GetParentTerrain () const [inline]

Gets the terrain that created this terrain tile and therefore owns it. Returns A const pointer to the parent terrain.

4.23.3.7 bool GetUpdateCache () const [inline]

Gets whether or not this tile's cache is out of sync with its data. Returns True if this tile's cache needs updating.

4.23.3.8 bool IsCachingEnabled () const [inline]

Gets the status of this tile's caching. Returns True if enabled, false otherwise.

4.23.3.9 void ReadFromCache () [virtual]

This method provides the paged tile with an opportunity to restore any data that was previously cached. Note The terrain readers and renderers handle data tracking of the tile's basic needs. This method provides a hook into the tile loading process by giving the tile an opportunity to load any application specific resources it may need.

Any errors should be tracked by throwing exceptions.

4.23.3.10 void SetBaseTextureImage (osg::Image * image) [inline]

Sets the base texture for this terrain tile. The base texture will be draped over the tile when it is rendered. Parameters

image The image to use as the base texture of this tile.

4.23.3.11 void SetCachePath (const std::string & path) [protected]

Sets the cache path for this tile. Any data corresponding to this tile can be cached by terrain readers and renderers when needed. Parameters

path The path assigned to this tile.

Note This path is most likely set by the terrain reader loading it. In most cases, this should not be modified by any other classes.

4.23.3.12 void SetGeoCoordinates (const GeoCoordinates & cs) [inline]

Sets the origin with which this tile maps to. Parameters

cs The coordinate system containing this tile's origin.

4.23.3.13 void SetHeightField (HeightField * hf) [inline]

Sets the heightfield that maps to this terrain tile. Parameters

hf The new heightfield.

4.23.3.14 void SetUpdateCache (bool flag) [inline]

Sets whether or not the cache needs to be updated to reflect the current state of the tile. This should be set to true when new data is set or added to the tile that needs to be cached. Parameters

flag True if the cache should be updated, false otherwise.

Note If this value is false, the parent terrain will not call the tile's **WriteToCache()** (p. 55) method.

4.23.3.15 void WriteToCache () [virtual]

This method provides the paged tile with an opportunity to cache any data that it wants to retrieve later. Note The terrain reader and renderers handle data tracking of the tile's basic needs. For example, the terrain reader will load/save the heightfield to the cache when needed. This method is intended to be used for application specific data in subclasses of the pagedterrain tile.

Any errors should be tracked by throwing exceptions.

4.23.4 Friends And Related Function Documentation**4.23.4.1 friend class PagedTerrainTileFactory [friend]**

Allow the factory and terrain to have access to this class.

4.23.4.2 friend class Terrain [friend]

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

- `pagedterraitile.h`
- `pagedterraitile.cpp`

4.24 PagedTerrainTileFactory Class Reference

This class is the base for a simple factory used to create new terrain tiles.

```
#include <inc/dtTerrain/pagedterraintilefactory.h>
```

Public Member Functions

- **PagedTerrainTileFactory** ()
Empty constructor...
- virtual **PagedTerrainTile * CreateNewTile** (const **GeoCoordinates** &coords, **Terrain** &owner)
This method creates a new terrain tile.

Protected Member Functions

- virtual **~PagedTerrainTileFactory** ()
Empty destructor...

4.24.1 Detailed Description

This class is the base for a simple factory used to create new terrain tiles. The terrain relies on this factory when it needs to create a new tile. Specific applications which rely on custom terrain tiles may wish to build a custom factory for the terrain to use.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 PagedTerrainTileFactory () [inline]

Empty constructor...

4.24.2.2 virtual ~PagedTerrainTileFactory () [inline, protected, virtual]

Empty destructor...

4.24.3 Member Function Documentation

4.24.3.1 virtual PagedTerrainTile* CreateNewTile (const GeoCoordinates & coords, Terrain & owner) [inline, virtual]

This method creates a new terrain tile. The default implementation merely creates a new tile and assigns its coordinates. Parameters

coords The coordinates mapped to this tile's location.

owner The terrain instance that owns this tile.

Returns A newly created tile.

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

- **pagedterraintilefactory.h**

4.25 PagedTerrainTileLoadState Class Reference

This enumeration defines the different states a tile could be in throughout its lifetime.

```
#include <inc/dtTerrain/pagedterraintile.h>
```

Static Public Attributes

- static const **PagedTerrainTileLoadState LOADED**
Tile is loaded and resides in memory.
- static const **PagedTerrainTileLoadState NOT_LOADED**
Tile is not resident in memory.
- static const **PagedTerrainTileLoadState PAGING**
Tile is in the process of being loaded from disk or other medium.

Protected Member Functions

- **PagedTerrainTileLoadState** (const std::string &name)

4.25.1 Detailed Description

This enumeration defines the different states a tile could be in throughout its lifetime.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 PagedTerrainTileLoadState (const std::string & name) [inline, protected]

4.25.3 Member Data Documentation

4.25.3.1 const PagedTerrainTileLoadState LOADED [static]

Tile is loaded and resides in memory.

4.25.3.2 const PagedTerrainTileLoadState NOT_LOADED [static]

Tile is not resident in memory. If a tile is in this state and the application needs its data, the terrain will automatically page it from disk.

4.25.3.3 const PagedTerrainTileLoadState PAGING [static]

Tile is in the process of being loaded from disk or other medium.

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

- **pagedterraintile.h**
- **pagedterraintile.cpp**

4.26 PagedTerrainTileResourceName Class Reference

This enumeration is used to identity built in resources that the terrain tiles are aware of.

```
#include <inc/dtTerrain/pagedterraintile.h>
```

Static Public Attributes

- static const **PagedTerrainTileResourceName HEIGHTFIELD_FILENAME**
This is a constant that defines the file name assigned to a heightfield resource for a particular tile.

Protected Member Functions

- **PagedTerrainTileResourceName** (const std::string &name)

4.26.1 Detailed Description

This enumeration is used to identity built in resources that the terrain tiles are aware of. These provide a unique identifier for the resources that need to be cached and uncached when terrain tiles are loaded and unloaded.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 PagedTerrainTileResourceName (const std::string & *name*) [*inline*, *protected*]

4.26.3 Member Data Documentation

4.26.3.1 const PagedTerrainTileResourceName HEIGHTFIELD_FILENAME [*static*]

This is a constant that defines the file name assigned to a heightfield resource for a particular tile. This should be used when writing and reading a tile's heightmap from its cache.

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

- **pagedterraintile.h**
- **pagedterraintile.cpp**

4.27 RawData Struct Reference

This structure is the raw data on disk.

```
#include <inc/dtTerrain/soarxdrawable.h>
```

Public Attributes

- float **error**
- float **height**
- float **radius**
- float **scale**

4.27.1 Detailed Description

This structure is the raw data on disk. It may be compressed or encoded. The data in this structure is converted to vertex data before it is rendered.

4.27.2 Member Data Documentation

4.27.2.1 float error

4.27.2.2 float height

4.27.2.3 float radius

4.27.2.4 float scale

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

- **soarxdrawable.h**

4.28 SoarXCacheResourceName Class Reference

```
#include <inc/dtTerrain/soarxterrainrenderer.h>
```

Static Public Attributes

- static const **SoarXCacheResourceName** **BASE_GRADIENT_TEXTURE**
Identifier for the cached preprocessed gradient map generated on a per tile basis.
- static const **SoarXCacheResourceName** **DETAIL_GRADIENT_TEXTURE**
Identifier for the cached preprocessed detail gradient texture used to add more detail to the terrain texturing.
- static const **SoarXCacheResourceName** **DETAIL_SCALE_MAP**
Identifier for the cached preprocessed detail scale map used when adding more texture detail to the terrain.
- static const **SoarXCacheResourceName** **DETAIL_VERTEX_NOISE**
Identifier for cached preprocessed noise data for dynamically adding noise to terrain vertex data.
- static const **SoarXCacheResourceName** **VERTEX_DATA**
Identifier for cached preprocessed vertex data.

Protected Member Functions

- **SoarXCacheResourceName** (const std::string &name)

4.28.1 Constructor & Destructor Documentation

4.28.1.1 **SoarXCacheResourceName** (const std::string & *name*) [inline, protected]

4.28.2 Member Data Documentation

4.28.2.1 **const SoarXCacheResourceName** **BASE_GRADIENT_TEXTURE** [static]

Identifier for the cached preprocessed gradient map generated on a per tile basis.

4.28.2.2 **const SoarXCacheResourceName** **DETAIL_GRADIENT_TEXTURE** [static]

Identifier for the cached preprocessed detail gradient texture used to add more detail to the terrain texturing.

4.28.2.3 **const SoarXCacheResourceName** **DETAIL_SCALE_MAP** [static]

Identifier for the cached preprocessed detail scale map used when adding more texture detail to the terrain.

4.28.2.4 **const SoarXCacheResourceName** **DETAIL_VERTEX_NOISE** [static]

Identifier for cached preprocessed noise data for dynamically adding noise to terrain vertex data.

4.28.2.5 **const SoarXCacheResourceName** **VERTEX_DATA** [static]

Identifier for cached preprocessed vertex data.

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

- **soarxterrainrenderer.h**
- **soarxterrainrenderer.cpp**

4.29 SoarXCullCallback Struct Reference

This callback is used by the SoarX renderer to update the camera's position or eyepoint which is needed by the refinement process to calculate projected vertex errors.

Public Member Functions

- virtual bool **cull** (osg::NodeVisitor *visitor, osg::Drawable *drawable, osg::State *state) const

4.29.1 Detailed Description

This callback is used by the SoarX renderer to update the camera's position or eyepoint which is needed by the refinement process to calculate projected vertex errors.

4.29.2 Member Function Documentation

4.29.2.1 virtual bool cull (osg::NodeVisitor * *visitor*, osg::Drawable * *drawable*, osg::State * *state*) const [inline, virtual]

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

- **soarxdrawable.cpp**

4.30 SoarXDrawable Class Reference

This class encapsulates the SoarX rendering algorithm itself.

```
#include <inc/dtTerrain/soarxdrawable.h>
```

Classes

- struct **Index**
This structure represents a location on the terrain's 2D grid.
- struct **RawData**
This structure is the raw data on disk.
- struct **Vertex**
This is the structure for a single processed vertex in the terrain.

Public Member Functions

- **SoarXDrawable** (const **SoarXDrawable** &rhs, const osg::CopyOp ©Op=osg::CopyOp::SHALLOW_COPY)
Makes a copy of the right hand side drawable.
- **SoarXDrawable** (int baseBits, float horizontalResolution)
Sets up some default rendering parameters used during the rendering and progressive refinement.
- bool **Build** (const **PagedTerrainTile** &tile)
Builds the internal data structures for use in rendering and managing terrain data for a tile.
- void **Clear** ()
Clears any memory used by the internal rendering structures.
- virtual osg::Object * **clone** (const osg::CopyOp ©op) const
Makes a new instance of this class by calling its copy constructor.
- virtual osg::Object * **cloneType** () const
Calls the default constructor on this class and returns a new instance.
- virtual osg::BoundingBox **computeBound** () const
Computes the dimensions of the currently loaded terrain.
- virtual void **drawImplementation** (osg::RenderInfo &renderInfo) const
Draws the current view of the terrain dataset.
- float **GetBaseHorizontalResolution** () const
Gets the base horizontal resolution.
- float **GetBaseVerticalResolution** () const
Gets the base vertical resolution.
- float **GetDetailHorizontalResolution** () const
Gets the detail map's horizontal resolution.
- float **GetDetailMultiplier** () const
Gets the current detail multiplier.

- int **GetDetailSize** () const
Gets the size of the detail map.
- float **GetDetailVerticalResolution** () const
Gets the detail map's vertical resolution.
- float **GetHeight** (float x, float y)
Gets the elevation at the current (x,y) coordinates.
- float **GetThreshold** () const
Gets the current error threshold.
- bool **RestoreDataFromCache** (const **PagedTerrainTile** &tile)
- float **SetDetailMultiplier** (float value)
Calculates and caches values used to determine error approximation in the rendering/refinement process.
- void **SetDetailNoise** (int detailBits, float detailVerticalResolution, float *detailData)
Sets the noise data used to procedurally generate additional detail vertices when rendering the terrain.
- void **SetEyePoint** (const osg::Vec3 &pos)
Sets the eye point of the camera.
- float **SetThreshold** (float value)
Sets the threshold value used during error projection and refinement in the rendering process.
- virtual bool **supports** (osg::PrimitiveFunctor &) const
- void **WriteDataToCache** (const **PagedTerrainTile** &tile)

Protected Member Functions

- virtual ~**SoarXDrawable** ()
Simply destroys the terrain drawable.
- bool **Active** (**Vertex** &v, unsigned int &planes)
Decides is a vertex to be in the resulting rendered terrain mesh.
- void **Append** (**Vertex** &v)
Adds a vertex and associated index to the vertex and index array.
- float **CalculateError** (**Index** i, **Index** j)
Calculates the object space error value of a vertex using the two indices provided.
- void **CalculateRadii** (float f)
- void **CalculateVertexErrors** ()
Precalculates vertex bounding spheres based on an object space error metric.
- void **CalculateVertexErrorsHelper** (**Index** i, **Index** j)
Recursive method used by the above method to build the error values.
- void **CheckChildren1** (**Index** index, unsigned int shift)
- void **CheckChildren2** (**Index** index, unsigned int shift)
- **RawData** * **GetRawData** (**Index** index)
Gets the raw data at the given index.
- void **GetVertex** (**Vertex** &v)
Retrieves a vertex from the procedural terrain mesh.

- **Vertex GetVertex (Index index)**
Gets a vertex based on the provided index.
- void **LeftRefine (Vertex &v1, Vertex &v2, bool in, unsigned int planes)**
Recursively refines the left of the two child triangles of the root refinement procedure.
- void **Refine (Vertex &v1, Vertex &v2, bool in, bool out, unsigned int planes)**
Starts the procedural view-dependent refinement process.
- void **Render (osg::State &state)**
Refines and renders the terrain.
- void **Repair (Index i, Index c)**
- void **RepairBoundingSphereHierarchy ()**
Ensures that the bounding spheres for each vertex satisfy the requirement that a vertex's bounding sphere contains all of its descendant vertices' bounding spheres.
- void **RightRefine (Vertex v1, Vertex &v2, bool out, unsigned int planes)**
Recursively refines the right of the two child triangles of the root refinement procedure.
- void **TurnCorner ()**
Inserts a degenerate triangle in the terrain vertex stream in order to generate a maximum length triangle strip during the refinement process.

Friends

- struct **SoarXCullCallback**

4.30.1 Detailed Description

This class encapsulates the SoarX rendering algorithm itself. See also **SoarXTerrainRenderer** (p. 69)

4.30.2 Constructor & Destructor Documentation

4.30.2.1 SoarXDrawable (int *baseBits*, float *horizontalResolution*)

Sets up some default rendering parameters used during the rendering and progressive refinement. Parameters

baseBits The size of the terrain chunk. (size=(2^{baseBits}))

horizontalResolution The distance between each height value.

4.30.2.2 SoarXDrawable (const SoarXDrawable & *rhs*, const osg::CopyOp & *copyOp* = osg::CopyOp::SHALLOW_COPY)

Makes a copy of the right hand side drawable. Parameters

rhs The **SoarXDrawable** (p. 63) to copy.

copyOp Specifies whether a deep copy or a shallow copy should occur.

4.30.2.3 ~SoarXDrawable () [protected, virtual]

Simply destroys the terrain drawable.

4.30.3 Member Function Documentation

4.30.3.1 bool Active (Vertex & *v*, unsigned int & *planes*) [protected]

Decides is a vertex to be in the resulting rendered terrain mesh. This is performed by calculating its position and its 2D/3D projected error. Hierarchical view-frustum culling is also performed here. Parameters

v The vertex to check.

planes A bitmask designating which planes to cull to.

level Current level of recursion.

4.30.3.2 void Append (Vertex & *v*) [protected]

Adds a vertex and associated index to the vertex and index array.

4.30.3.3 bool Build (const PagedTerrainTile & *tile*)

Builds the internal data structures for use in rendering and managing terrain data for a tile. Parameters

The source terrain tile.

Returns True if the data was paged from the tile's cache, false otherwise. Note This method will load pre-computed data from the tile's cache if present. If the cache is not present, the data will be precomputed and then cached if the tile has its caching enabled.

4.30.3.4 float CalculateError (Index *i*, Index *j*) [protected]

Calculates the object space error value of a vertex using the two indices provided.

4.30.3.5 void CalculateRadii (float *f*) [protected]

4.30.3.6 void CalculateVertexErrors () [protected]

Precalculates vertex bounding spheres based on an object space error metric.

4.30.3.7 void CalculateVertexErrorsHelper (Index *i*, Index *j*) [protected]

Recursive method used by the above method to build the error values.

4.30.3.8 void CheckChildren1 (Index *index*, unsigned int *shift*) [protected]

4.30.3.9 void CheckChildren2 (Index *index*, unsigned int *shift*) [protected]

4.30.3.10 void Clear ()

Clears any memory used by the internal rendering structures.

4.30.3.11 osg::Object * clone (const osg::CopyOp & *copyop*) const [virtual]

Makes a new instance of this class by calling its copy constructor.

4.30.3.12 osg::Object * cloneType () const [virtual]

Calls the default constructor on this class and returns a new instance.

4.30.3.13 osg::BoundingBox computeBound () const [virtual]

Computes the dimensions of the currently loaded terrain.

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

Draws the current view of the terrain dataset. Parameters

state The current OpenGL state.

4.30.3.15 float GetBaseHorizontalResolution () const [inline]

Gets the base horizontal resolution. This number determines the overall scaling for the terrain's X and Y dimensions. Returns The current horizontal resolution.

4.30.3.16 float GetBaseVerticalResolution () const [inline]

Gets the base vertical resolution. This number determines the overall scaling for height values in the terrain. Returns The current vertical resolution.

4.30.3.17 float GetDetailHorizontalResolution () const [inline]

Gets the detail map's horizontal resolution.

4.30.3.18 float GetDetailMultiplier () const [inline]

Gets the current detail multiplier. Returns The detail multiplier.

4.30.3.19 int GetDetailSize () const [inline]

Gets the size of the detail map.

4.30.3.20 float GetDetailVerticalResolution () const [inline]

Gets the detail map's vertical resolution.

4.30.3.21 float GetHeight (float x, float y)

Gets the elevation at the current (x,y) coordinates.

4.30.3.22 RawData* GetRawData (Index index) [inline, protected]

Gets the raw data at the given index.

4.30.3.23 float GetThreshold () const [inline]

Gets the current error threshold. Returns The current value.

4.30.3.24 void GetVertex (Vertex & v) [protected]

Retrieves a vertex from the procedural terrain mesh.

4.30.3.25 SoarXDrawable::Vertex GetVertex (Index index) [protected]

Gets a vertex based on the provided index.

4.30.3.26 void LeftRefine (Vertex & v1, Vertex & v2, bool in, unsigned int planes) [protected]

Recursively refines the left of the two child triangles of the root refinement procedure.

4.30.3.27 void Refine (Vertex & v1, Vertex & v2, bool in, bool out, unsigned int planes) [protected]

Starts the procedural view-dependent refinement process. The SoarX refinement process is based on the longest edge bisection of isosceles right triangles. It subdivides each triangle about its hypotenuse, creating two smaller triangles.

4.30.3.28 void Render (osg::State & state) [protected]

Refines and renders the terrain.

(111111) bitmask instructing the start of the refinement process to check all 6 planes.

4.30.3.29 void Repair (Index i, Index c) [protected]**4.30.3.30 void RepairBoundingSphereHierarchy () [protected]**

Ensures that the bounding spheres for each vertex satisfy the requirement that a vertex's bounding sphere contains all of its descendant vertices' bounding spheres. This ensures a proper sphere tree for all the vertices.

4.30.3.31 bool RestoreDataFromCache (const PagedTerrainTile & tile)**4.30.3.32 void RightRefine (Vertex v1, Vertex & v2, bool out, unsigned int planes) [protected]**

Recursively refines the right of the two child triangles of the root refinement procedure.

4.30.3.33 float SetDetailMultiplier (float *value*) [inline]

Calculates and caches values used to determine error approximation in the rendering/refinement process. Parameters

value

Returns The clamped multiplier value. Note The value given is clamped within the range 1-20.

4.30.3.34 void SetDetailNoise (int *detailBits*, float *detailVerticalResolution*, float * *detailData*)

Sets the noise data used to procedurally generate additional detail vertices when rendering the terrain. Parameters

detailBits The size of the detail data. The amount of data should be $(2^{\text{detailBits}} * 2^{\text{detailBits}})$.

detailVerticalResolution A scale value used when calculating the final detail vertex position.

detailData The detail noise values.

4.30.3.35 void SetEyePoint (const osg::Vec3 & *pos*) [inline]

Sets the eye point of the camera. This is used during the refinement process to calculate vertex projection errors. Parameters

pos The camera position.

4.30.3.36 float SetThreshold (float *value*) [inline]

Sets the threshold value used during error projection and refinement in the rendering process. Parameters

value The new threshold value.

Returns The clamped value. Note The value given is clamped within the range 1-10

4.30.3.37 virtual bool supports (osg::PrimitiveFunctor &) const [inline, virtual]**4.30.3.38 void TurnCorner () [protected]**

Inserts a degenerate triangle in the terrain vertex stream in order to generate a maximum length triangle strip during the refinement process.

4.30.3.39 void WriteDataToCache (const PagedTerrainTile & *tile*)**4.30.4 Friends And Related Function Documentation****4.30.4.1 friend struct SoarXCullCallback [friend]**

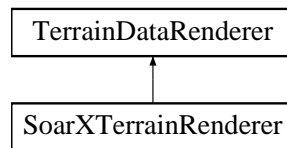
The documentation for this class was generated from the following files:

- **soarxdrawable.h**
- **soarxdrawable.cpp**

4.31 SoarXTerrainRenderer Class Reference

This renderer is an integration/implementation of the SoarX terrain rendering algorithm.

#include <inc/dtTerrain/soarxterrainrenderer.h> Inheritance diagram for SoarXTerrainRenderer::



Classes

- struct **DrawableEntry**
This structure represents a drawable tile.

Public Types

- typedef std::map< dtCore::RefPtr< **PagedTerrainTile** >, **DrawableEntry** > **DrawableMap**
Help minimize some typing...

Public Member Functions

- **SoarXTerrainRenderer** (const std::string &name="SoarXRenderer")
Constructs the SoarX renderer.
- float **GetDetailMultiplier** () const
Gets the current detail multiplier.
- float **GetHeight** (float x, float y)
Gets the height of the terrain at the specified (x,y) coordinates.
- osg::Vec3 **GetNormal** (float x, float y)
Gets the normal vector at the specified point.
- osg::Group * **GetRootDrawable** ()
Returns a scene node that encapsulates the renderable terrain.
- const std::string & **GetTerrainFragmentShader** () const
Gets the file name containing the fragment shader currently in use by the terrain renderer.
- float **GetThreshold** () const
*Gets the threshold value of the **SoarXDrawable** (p. 63).*
- void **OnLoadTerrainTile** (**PagedTerrainTile** &tile)
*This method constructs a **SoarXDrawable** (p. 63) for the new terrain tile that needs to be loaded.*
- void **OnUnloadTerrainTile** (**PagedTerrainTile** &tile)
This method updates an internal map of tiles and drawables based on what tiles were unloaded from the terrain.
- void **SetDetailMultiplier** (float value)
Sets the detail multiplier on the SoarX drawable.
- void **SetEnableFog** (bool pEnableFog)

Tells the terrain whether or not to use the fog variables when rendering the terrain by default there is no fog.

- void **SetTerrainFragmentShader** (const std::string &filePath)
Sets the file containing the fragment shader to use when rendering the terrain.
- void **SetThreshold** (float value)
*Sets the threshold value of the **SoarXDrawable** (p. 63).*

Static Public Attributes

- static const float **GRADIENT_SCALE** = 32.0f
Used to scale the precalculated gradient values used to render the terrain.

Protected Member Functions

- virtual ~**SoarXTerrainRenderer** ()
Destroy the renderer.
- void **CalculateDetailNoise** ()
Builds an array of noise values used to dynamically add more detail to the terrain.
- void **CheckBaseGradientCache** (const **PagedTerrainTile** &tile, **DrawableEntry** &entry)
This method is called for each tile that is loaded.
- void **CheckDetailGradientCache** ()
Checks the cache for a detail gradient texture.
- void **CheckDetailNoiseCache** ()
Attempts to load the detail noise data from the terrain cache.
- void **CheckDetailScaleCache** ()
Checks the cache for a detail scale map.
- void **CreateFragmentShader** ()
Creates the GLSL program object.
- void **InitializeRenderer** ()
This method initializes data for this renderer that is shared amongst the terrain tiles.
- void **SetupRenderState** (**PagedTerrainTile** &tile, **DrawableEntry** &entry, osg::StateSet &ss)
Configures the specified drawable and stateset to use the proper textures and render state for SoarX terrain rendering.

4.31.1 Detailed Description

This renderer is an integration/implementation of the SoarX terrain rendering algorithm. The SoarX algorithm is an extension of the original SOAR (Stateless One-pass Adaptive Refinement) algorithm developed by Peter Lindstrom and Valerio Pascucci. The SOAR algorithm itself has several components and features including adaptive refinement, on-the-fly triangle stripping and smooth geomorphing based on projected error. The SoarX implementation extends and improves on some of the shortcomings of the original SOAR terrain rendering solution. The original SoarX algorithm and demo application developed by Andras Balogh can be found at <http://web.interware.hu/bandi/ranger.html>.

4.31.2 Member Typedef Documentation

4.31.2.1 typedef std::map<dtCore::RefPtr<PagedTerrainTile>,DrawableEntry> DrawableMap

Help minimize some typing...

4.31.3 Constructor & Destructor Documentation

4.31.3.1 SoarXTerrainRenderer (const std::string & name = "SoarXRenderer")

Constructs the SoarX renderer. Parameters

name Simple name to assign to this object. Default = "SoarXRenderer"

4.31.3.2 ~SoarXTerrainRenderer () [protected, virtual]

Destroy the renderer.

4.31.4 Member Function Documentation

4.31.4.1 void CalculateDetailNoise () [protected]

Builds an array of noise values used to dynamically add more detail to the terrain.

4.31.4.2 void CheckBaseGradientCache (const PagedTerrainTile & tile, DrawableEntry & entry) [protected]

This method is called for each tile that is loaded. It checks to see if there is a gradient texture in the tile's cache path and if so loads it. If not, the gradient texture is generated and cached if enabled.

4.31.4.3 void CheckDetailGradientCache () [protected]

Checks the cache for a detail gradient texture. If found, it will load it else it will be generated and then cached if enabled.

4.31.4.4 void CheckDetailNoiseCache () [protected]

Attempts to load the detail noise data from the terrain cache. If it is not present, the data will be generated.

4.31.4.5 void CheckDetailScaleCache () [protected]

Checks the cache for a detail scale map. If found, it will load it, else the data will be generated then cached if enabled.

4.31.4.6 void CreateFragmentShader () [protected]

Creates the GLSL program object.

4.31.4.7 float GetDetailMultiplier () const

Gets the current detail multiplier. Returns The current detail multiplier. This will equal zero if the **SoarXDrawable** (p. 63) is not valid.

4.31.4.8 float GetHeight (float x, float y) [virtual]

Gets the height of the terrain at the specified (x,y) coordinates. Returns The height of the terrain at the specified point.

Implements **TerrainDataRenderer** (p. 87).

4.31.4.9 osg::Vec3 GetNormal (float x, float y) [virtual]

Gets the normal vector at the specified point. Returns A vector perpendicular to terrain at the given point.

Implements **TerrainDataRenderer** (p. 87).

4.31.4.10 osg::Group* GetRootDrawable () [inline, virtual]

Returns a scene node that encapsulates the renderable terrain. Returns An OpenSceneGraph group node that contains all the information needed to render the terrain. Note Initialize() is guaranteed to be called before this method by the parent terrain.

Implements **TerrainDataRenderer** (p. 87).

4.31.4.11 const std::string& GetTerrainFragmentShader () const [inline]

Gets the file name containing the fragment shader currently in use by the terrain renderer.

4.31.4.12 float GetThreshold () const

Gets the threshold value of the **SoarXDrawable** (p. 63). See also **SoarXDrawable::GetThreshold()** (p. 67)

4.31.4.13 void InitializeRenderer () [protected]

This method initializes data for this renderer that is shared amongst the terrain tiles. This includes shaders, textures, some of which are read from the cache if it exists.

4.31.4.14 void OnLoadTerrainTile (PagedTerrainTile & tile) [virtual]

This method constructs a **SoarXDrawable** (p. 63) for the new terrain tile that needs to be loaded. Parameters

tile The new tile.

See also **SoarXDrawable** (p. 63)

Implements **TerrainDataRenderer** (p. 87).

4.31.4.15 void OnUnloadTerrainTile (PagedTerrainTile & tile) [virtual]

This method updates an internal map of tiles and drawables based on what tiles were unloaded from the terrain. Parameters

tile The tile being unloaded.

Reimplemented from **TerrainDataRenderer** (p. 87).

4.31.4.16 void SetDetailMultiplier (float value)

Sets the detail multiplier on the SoarX drawable. See also **SoarXDrawable::SetDetailMultiplier** (p. 68).

4.31.4.17 void SetEnableFog (bool pEnableFog)

Tells the terrain whether or not to use the fog variables when rendering the terrain by default there is no fog.

4.31.4.18 void SetTerrainFragmentShader (const std::string & filePath) [inline]

Sets the file containing the fragment shader to use when rendering the terrain. Parameters

filePath The path to the shader file. This path is relative to the Delta3D data path list. The default path is "shaders/terrain.frag".

Note This has no affect if the use of fragment shaders has been disabled.

4.31.4.19 void SetThreshold (float value)

Sets the threshold value of the **SoarXDrawable** (p. 63). See also **SoarXDrawable::SetThreshold()** (p. 68)

4.31.4.20 void SetupRenderState (PagedTerrainTile & tile, DrawableEntry & entry, osg::StateSet & ss) [protected]

Configures the specified drawable and stateset to use the proper textures and render state for SoarX terrain rendering.

4.31.5 Member Data Documentation

4.31.5.1 `const float GRADIENT_SCALE = 32.0f` [static]

Used to scale the precalculated gradient values used to render the terrain.

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

- `soarxterrainrenderer.h`
- `soarxterrainrenderer.cpp`

4.32 Terrain Class Reference

This is the base terrain class that provides most of the functionality required for terrain rendering.

```
#include <inc/dtTerrain/terrain.h>
```

Public Member Functions

- **Terrain** (const std::string &name="Terrain")
Default Constructor - Sets the name and performs some routine registration.
- void **AddDecorationLayer** (**TerrainDecorationLayer** *newLayer)
Adds a new decoration layer to this terrain.
- void **AddResourcePath** (const std::string &path)
Adds a search path to the list of file paths to search for when loading terrain data or other resources.
- void **ClearDecorationLayers** ()
Removes all the decorator layers from this terrain.
- virtual **PagedTerrainTile** * **CreateTerrainTile** (const **GeoCoordinates** &coords)
- virtual void **EnsureTileVisibility** (const std::set< **GeoCoordinates** > &coordList)
- void **FindAllResources** (const std::string &path, std::vector< std::string > &resourcePaths)
Searches the terrain resources path list for all resources matching the specified path.
- const std::string **FindResource** (const std::string &path)
Searches the terrains resource path list for the specified resource.
- const std::string & **GetCachePath** () const
Gets the current terrain cache.
- const **TerrainDataReader** * **GetDataReader** () const
Gets the current terrain data reader.
- **TerrainDataReader** * **GetDataReader** ()
Gets the current terrain data reader.
- const **TerrainDataRenderer** * **GetDataRenderer** () const
Gets the current terrain data renderer.
- **TerrainDataRenderer** * **GetDataRenderer** ()
Gets the current terrain data renderer.
- const **TerrainDecorationLayer** * **GetDecorationLayer** (const std::string &name) const
Gets the decoration layer who's name matches the specified parameter.
- **TerrainDecorationLayer** * **GetDecorationLayer** (const std::string &name)
Gets the decoration layer whos name matches the specified parameter.
- void **GetDecorationLayers** (std::vector< dtCore::RefPtr< **TerrainDecorationLayer** > > &layers)
Fills a vector with the decoration layers currently attached to this terrain.
- float **GetHeight** (float x, float y)
Gets the height of the terrain at the given (x,y) coordinates.
- float **GetLineOfSightSpacing** () const
- float **GetLoadDistance** () const

- unsigned int **GetNumDecorationLayers** () const
Gets the number of decorator layers currently overlaid on this terrain.
- const std::list< std::string > & **GetResourcePathList** () const
Gets the terrain's current list of resource search paths.
- **PagedTerrainTileFactory** * **GetTerrainTileFactory** ()
- void **HideDecorationLayer** (**TerrainDecorationLayer** *toHide)
Hides the specified decoration layer.
- void **HideDecorationLayer** (const std::string &name)
Hides the decoration layer that matches the specified name.
- bool **IsClearLineOfSight** (const osg::Vec3 &pointOne, const osg::Vec3 &pointTwo)
Given pointOne and pointTwo, both in world space and with all coordinates in Cartesian space (essentially in meters along X, Y and Z), returns true if there is a clear line of sight and false if the view is blocked.
- virtual bool **IsTerrainTileResident** (const **GeoCoordinates** &coords)
- virtual void **LoadTerrainTile** (**PagedTerrainTile** &newTile)
- virtual void **OnMessage** (dtCore::Base::MessageData *data)
- void **RemoveDecorationLayer** (const std::string &name)
Removes a decoration layer matching the specified name.
- void **RemoveDecorationLayer** (**TerrainDecorationLayer** *toRemove)
Removes the specified decoration layer from this terrain.
- void **RemoveResourcePath** (const std::string &path)
Removes a resource path from the list of known resource paths.
- bool **SetCachePath** (const std::string &path)
Sets the path of the terrain cache.
- void **SetDataReader** (**TerrainDataReader** *reader)
Sets the terrain data reader.
- void **SetDataRenderer** (**TerrainDataRenderer** *renderer)
Sets the terrain data renderer.
- void **SetLineOfSightSpacing** (float spacing)
- void **SetLoadDistance** (float value)
- void **SetTerrainTileFactory** (**PagedTerrainTileFactory** &factory)
- void **ShowDecorationLayer** (**TerrainDecorationLayer** *toShow)
Makes the specified decoration layer visible in the scene.
- void **ShowDecorationLayer** (const std::string &name)
Makes the decoration layer matching the specified name visible.
- virtual void **UnloadAllTerrainTiles** ()
- virtual void **UnloadTerrainTile** (**PagedTerrainTile** &toRemove)

Protected Member Functions

- virtual ~**Terrain** ()
Cleans up the terrain and its components.
- virtual void **PostFrame** (double frameTime)
- virtual void **PreFrame** (double frameTime)

Protected Attributes

- TerrainTileMap **mResidentTiles**
List of terrain tiles which are currently loaded into memory.
- std::queue< dtCore::RefPtr< **PagedTerrainTile** > > **mTilesToLoadQ**
List of terrain tiles currently queued up for loading.
- std::queue< dtCore::RefPtr< **PagedTerrainTile** > > **mTilesToUnloadQ**
Queue of terrain tiles that need to be cached or destroyed.

4.32.1 Detailed Description

This is the base terrain class that provides most of the functionality required for terrain rendering. This class uses a component or aggregate based design philosophy by utilizing terrain data readers and terrain data renderers. Each reader loads a specific type of terrain data and the renderer implements a particular terrain rendering algorithm. Each of these components can be dynamically changed on each instance of this terrain class.

4.32.2 Constructor & Destructor Documentation

4.32.2.1 Terrain (const std::string & name = "Terrain")

Default Constructor - Sets the name and performs some routine registration. Parameters

name The name of this terrain object.

4.32.2.2 ~Terrain () [protected, virtual]

Cleans up the terrain and its components.

4.32.3 Member Function Documentation

4.32.3.1 void AddDecorationLayer (TerrainDecorationLayer * newLayer)

Adds a new decoration layer to this terrain. Parameters

newLayer The new layer to add.

Note This will also ask the layer for its scene node (GetOSGNode()) and add it as a child to the terrain.

If the new layer's name is not unique, the name is appended with a unique identifier and a warning is logged.

4.32.3.2 void AddResourcePath (const std::string & path)

Adds a search path to the list of file paths to search for when loading terrain data or other resources. This path must be relative to a path in the Delta3D data path list. Parameters

The new search path.

4.32.3.3 void ClearDecorationLayers ()

Removes all the decorator layers from this terrain.

4.32.3.4 PagedTerrainTile * CreateTerrainTile (const GeoCoordinates & coords) [virtual]

4.32.3.5 void EnsureTileVisibility (const std::set< GeoCoordinates > & coordList) [virtual]

4.32.3.6 void FindAllResources (const std::string & path, std::vector< std::string > & resourcePaths)

Searches the terrain resources path list for all resources matching the specified path. The path to the resource to load. This should be either a filename or a path relative to the resource paths registered with the terrain. The result list filled with absolute paths to all resources matching the specified path.

4.32.3.7 const std::string FindResource (const std::string & path)

Searches the terrains resource path list for the specified resource. The first resource matching the specified path is returned. Parameters

The path to the resource to load. This should be either a filename or a path relative to the resource paths registered with the terrain.

Returns An absolute path to the specified resource. If the resource was not found, an empty string is returned.

4.32.3.8 const std::string& GetCachePath () const [inline]

Gets the current terrain cache. Returns The base cache path for this terrain.

4.32.3.9 const TerrainDataReader* GetDataReader () const [inline]

Gets the current terrain data reader. Returns A const data reader object.

4.32.3.10 TerrainDataReader* GetDataReader () [inline]

Gets the current terrain data reader. Returns A data reader object.

4.32.3.11 const TerrainDataRenderer* GetDataRenderer () const [inline]

Gets the current terrain data renderer. Returns A const data renderer object.

4.32.3.12 TerrainDataRenderer* GetDataRenderer () [inline]

Gets the current terrain data renderer. Returns A data renderer object.

4.32.3.13 const TerrainDecorationLayer * GetDecorationLayer (const std::string & name) const

Gets the decoration layer who's name matches the specified parameter. Parameters

name The name of the layer to retrieve.

Returns A valid decoration layer or NULL if the layer could not be found.

4.32.3.14 TerrainDecorationLayer * GetDecorationLayer (const std::string & name)

Gets the decoration layer whos name matches the specified parameter. Parameters

name The name of the layer to retrieve.

Returns A valid decoration layer or NULL if the layer could not be found.

4.32.3.15 void GetDecorationLayers (std::vector< dtCore::RefPtr< TerrainDecorationLayer > > & layers)

Fills a vector with the decoration layers currently attached to this terrain. Parameters

layers The vector to be filled.

Note The vector is cleared before it is filled so any data that is passed to this method will be destroyed.

4.32.3.16 float GetHeight (float x, float y)

Gets the height of the terrain at the given (x,y) coordinates. Note This just calls the current terrain renderer's GetHeight method. If the renderer is invalid, the height returned is 0.0f.

4.32.3.17 float GetLineOfSightSpacing () const [inline]**4.32.3.18 float GetLoadDistance () const [inline]****4.32.3.19 unsigned int GetNumDecorationLayers () const [inline]**

Gets the number of decorator layers currently overlaid on this terrain. Returns The number of layers.

4.32.3.20 const std::list<std::string>& GetResourcePathList () const [inline]

Gets the terrain's current list of resource search paths. Returns The list of string paths.

4.32.3.21 PagedTerrainTileFactory* GetTerrainTileFactory () [inline]**4.32.3.22 void HideDecorationLayer (TerrainDecorationLayer * *toHide*)**

Hides the specified decoration layer. Parameters

toHide The layer to hide.

4.32.3.23 void HideDecorationLayer (const std::string & *name*)

Hides the decoration layer that matches the specified name. Parameters

name The name of the layer to hide.

4.32.3.24 bool IsClearLineOfSight (const osg::Vec3 & *pointOne*, const osg::Vec3 & *pointTwo*)

Given *pointOne* and *pointTwo*, both in world space and with all coordinates in Cartesian space (essentially in meters along X, Y and Z), returns true if there is a clear line of sight and false if the view is blocked. Parameters

pointOne The start point.

pointTwo The end point.

Returns Returns true if there is a clear line of sight from *pointOne* to *pointTwo* and false if the view is blocked.

4.32.3.25 bool IsTerrainTileResident (const GeoCoordinates & *coords*) [virtual]**4.32.3.26 void LoadTerrainTile (PagedTerrainTile & *newTile*) [virtual]****4.32.3.27 void OnMessage (dtCore::Base::MessageData * *data*) [virtual]****4.32.3.28 void PostFrame (double *frameTime*) [protected, virtual]****4.32.3.29 void PreFrame (double *frameTime*) [protected, virtual]****4.32.3.30 void RemoveDecorationLayer (const std::string & *name*)**

Removes a decoration layer matching the specified name. Parameters

name The name of the terrain layer to remove.

4.32.3.31 void RemoveDecorationLayer (TerrainDecorationLayer * *toRemove*)

Removes the specified decoration layer from this terrain. Parameters

toRemove The layer to remove.

Note This will also remove the scene node for the decoration layer from this terrain.

4.32.3.32 void RemoveResourcePath (const std::string & *path*)

Removes a resource path from the list of known resource paths. Parameters

path The path to remove. If it is not currently in the list this method does nothing.

4.32.3.33 bool SetCachePath (const std::string & path)

Sets the path of the terrain cache. The terrain cache is a directory somewhere on the hard drive which is used to store on the fly data calculations which can then be reused on successive runs. If this path is not set, no data caching will occur. Note The directory will be created if it does not already exist.

This path will be the root cache path for any terrain tiles that need to have their data cached. Each tile's cache path is a subdirectory of this one.

This path should only be used directly when data needs to be cached that is independant of a particular terrain tile. Tile specific data should be cached in the tile's cache directory.

Parameters

path The path to use for the terrain cache.

Exceptions

Exception if the path does not exist and cannot be created.

4.32.3.34 void SetDataReader (TerrainDataReader * reader)

Sets the terrain data reader. This must be set before any terrain can be loaded.

4.32.3.35 void SetDataRenderer (TerrainDataRenderer * renderer)

Sets the terrain data renderer. This must be set before any terrain can be loaded and rendered.

4.32.3.36 void SetLineOfSightSpacing (float spacing) [inline]**4.32.3.37 void SetLoadDistance (float value) [inline]****4.32.3.38 void SetTerrainTileFactory (PagedTerrainTileFactory & factory) [inline]****4.32.3.39 void ShowDecorationLayer (TerrainDecorationLayer * toShow)**

Makes the specified decoration layer visible in the scene. Parameters

The layer to make visible.

4.32.3.40 void ShowDecorationLayer (const std::string & name)

Makes the decoration layer matching the specified name visible. Parameters

name The name of the layer to show.

4.32.3.41 void UnloadAllTerrainTiles () [virtual]**4.32.3.42 void UnloadTerrainTile (PagedTerrainTile & toRemove) [virtual]****4.32.4 Member Data Documentation****4.32.4.1 TerrainTileMap mResidentTiles [protected]**

List of terrain tiles which are currently loaded into memory.

4.32.4.2 std::queue<dtCore::RefPtr<PagedTerrainTile>> mTilesToLoadQ [protected]

List of terrain tiles currently queued up for loading.

4.32.4.3 std::queue<dtCore::RefPtr<PagedTerrainTile>> mTilesToUnloadQ [protected]

Queue of terrain tiles that need to be cached or destroyed.

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

- terrain.h
- terrain.cpp

4.33 TerrainCullCallback Class Reference

Public Member Functions

- **TerrainCullCallback** (**Terrain** *terrain)
- virtual void **operator()** (**osg::Node** *node, **osg::NodeVisitor** *nv)

4.33.1 Constructor & Destructor Documentation

4.33.1.1 **TerrainCullCallback** (**Terrain** * *terrain*) [**inline**]

4.33.2 Member Function Documentation

4.33.2.1 **virtual void operator()** (**osg::Node** * *node*, **osg::NodeVisitor** * *nv*) [**inline**, **virtual**]

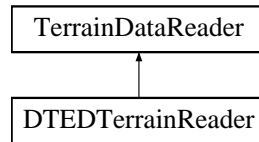
The documentation for this class was generated from the following file:

- **terrain.cpp**

4.34 TerrainDataReader Class Reference

This is mostly an interface class to the data reader functionality of Delta3D terrains.

#include <inc/dtTerrain/terraindatareader.h>Inheritance diagram for TerrainDataReader::



Public Member Functions

- **TerrainDataReader** (const std::string &name="TerrainDataReader")
Constructs the terrain data reader.
- virtual const std::string **GenerateTerrainTileCachePath** (const **PagedTerrainTile** &tile)=0
This method is used by the terrain reader to generate a unique path name for a terrain tile's cached data.
- virtual const **TerrainDataType** & **GetDataType** () const =0
Gets the type of data this reader supports.
- const **Terrain** * **GetParentTerrain** () const
Gets a read-only terrain object that currently owns this reader.
- **Terrain** * **GetParentTerrain** ()
Gets the terrain object that currently owns this reader.
- const **HeightFieldResizePolicy** & **GetResizePolicy** () const
Gets the current heightfield resize policy.
- virtual bool **OnLoadTerrainTile** (**PagedTerrainTile** &tile)=0
This method is called when the parent terrain needs to load a tile.
- virtual void **OnUnloadTerrainTile** (**PagedTerrainTile** &tile)
This method is called when the parent terrain wishes to unload a terrain tile from its list of resident tiles.
- void **SetResizePolicy** (const **HeightFieldResizePolicy** &policy)
Sets the heightfield resize policy for this reader.

Protected Member Functions

- virtual ~**TerrainDataReader** ()
Empty destructor...
- **HeightField** * **ConvertHeightField** (osg::HeightField *hf)
Converts an OpenSceneGraph floating point heightfield to an internal 16-bit heightfield.
- float **GetInterpolatedHeight** (const osg::HeightField *hf, float x, float y)
Performs a bi-linear interpolation at the specified coordinate to sample a height value.
- dtCore::RefPtr< osg::HeightField > **ScaleOSGHeightField** (osg::HeightField *hf)
Scales an OpenSceneGraph heightfield according to the current resize policy.

Friends

- class **Terrain**

Allow the terrain to have access to this class.

4.34.1 Detailed Description

This is mostly an interface class to the data reader functionality of Delta3D terrains. This allows the terrain to support an arbitrary number of terrain formats as long as the reader can parse the data and convert it to a terrain tile. TerrainDataReaders are primarily in charge of loading heightfield data, however, some may load more than that depending on a specific applications needs.

4.34.2 Constructor & Destructor Documentation

4.34.2.1 TerrainDataReader (const std::string & *name* = "TerrainDataReader") [inline]

Constructs the terrain data reader. Sets the default resize policy to NEAREST_POWER_OF_TWO_PLUS_ONE since this is a fairly common size requirement for terrain rendering architectures.

4.34.2.2 virtual ~TerrainDataReader () [inline, protected, virtual]

Empty destructor...

4.34.3 Member Function Documentation

4.34.3.1 HeightField * ConvertHeightField (osg::HeightField * *hf*) [protected]

Converts an OpenSceneGraph floating point heightfield to an internal 16-bit heightfield. Parameters

hf The OSG heightfield.

Returns A 16-bit heightfield. See also **HeightField** (p. 30)

4.34.3.2 virtual const std::string GenerateTerrainTileCachePath (const PagedTerrainTile & *tile*) [pure virtual]

This method is used by the terrain reader to generate a unique path name for a terrain tile's cached data. This path is a subdirectory under the parent terrain's cache path and should be unique on a per tile basis. This id should also be consistent across application invocations. Parameters

tile The tile with which to generate the path for.

Returns The tile specific cache path. Note As an example, the **DTEDTerrainReader** (p. 16) generates this path by concatenating the latitude, longitude, and DTED level.

On a side note, I would have preferred that the parent terrain generate this cache path, however, I was unable to come up with a generic mechanism for this to occur so I thought the reader would be the next most likely candidate.

Implemented in **DTEDTerrainReader** (p. 17).

4.34.3.3 virtual const TerrainDataType& GetDataType () const [pure virtual]

Gets the type of data this reader supports. Returns **TerrainDataType** (p. 89) The data type for this reader.

Implemented in **DTEDTerrainReader** (p. 17).

4.34.3.4 float GetInterpolatedHeight (const osg::HeightField * *hf*, float *x*, float *y*) [protected]

Performs a bi-linear interpolation at the specified coordinate to sample a height value. Parameters

hf The heightfield to query.

x The x coordinate to sample from.

y The y coordinate to sample from.

4.34.3.5 const Terrain* GetParentTerrain () const [inline]

Gets a read-only terrain object that currently owns this reader. Returns A const pointer to the parent terrain.

4.34.3.6 Terrain* GetParentTerrain () [inline]

Gets the terrain object that currently owns this reader. Returns A pointer to the parent terrain.

4.34.3.7 const HeightFieldResizePolicy& GetResizePolicy () const [inline]

Gets the current heightfield resize policy. Returns The current policy.

4.34.3.8 virtual bool OnLoadTerrainTile (PagedTerrainTile & tile) [pure virtual]

This method is called when the parent terrain needs to load a tile. Parameters

tile The new tile. The loader should populate the fields of the tile as appropriate.

Note Should throw an exception if any errors occur.

At this point, the terrain tiles have had a chance to load data from their cache. Therefore, readers implementing this method may want to check the status of a tile's data before loading since it may have been retrieved from the cache.

Resources may be looked up using the parent terrain's resource path list. In most cases, the terrain will be made aware of any resource locations its readers may need. See also **Terrain** (p. 74)

PagedTerrainTile (p. 53)

Implemented in **DTEDTerrainReader** (p. 17).

4.34.3.9 virtual void OnUnloadTerrainTile (PagedTerrainTile & tile) [inline, virtual]

This method is called when the parent terrain wishes to unload a terrain tile from its list of resident tiles. Parameters

tile The tile being unloaded.

Note Should throw an exception if any errors occur.

The default implementation does nothing as most renderers will probably only process data and have no need to respond to this method, however, it is available if needed. Also note, that the tile itself is responsible for caching its data, however, any data a renderer wishes to control may be cached in this method. See also **Terrain** (p. 74)

PagedTerrainTile (p. 53)

4.34.3.10 dtCore::RefPtr< osg::HeightField > ScaleOSGHeightField (osg::HeightField * hf) [protected]

Scales an OpenSceneGraph heightfield according to the current resize policy. Parameters

hf The OSG heightfield to resize.

Returns The resized heightfield. If the heightfield to resize meets the current resize criteria or the current policy is NONE, this will be a pointer to the same data that was passed as a parameter.

4.34.3.11 void SetResizePolicy (const HeightFieldResizePolicy & policy) [inline]

Sets the heightfield resize policy for this reader. This is used when loading a new tile. If the policy is not set to NONE, the loaded heightfield will be resized according to the current policy. Parameters

policy The new policy.

Note This is useful as many renderers require their heightfield data to have certain dimensions in order for their refinement and or data hierarchy to work. By default, the policy is set to POWER_OF_TWO_PLUS_ONE.

4.34.4 Friends And Related Function Documentation

4.34.4.1 friend class Terrain [friend]

Allow the terrain to have access to this class.

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

- **terraindatareader.h**
- **terraindatareader.cpp**

4.35 TerrainDataReaderException Class Reference

These exceptions are used by terrain data readers when an error occurs during the load process.

```
#include <inc/dtTerrain/terraindatareader.h>
```

Static Public Attributes

- static **TerrainDataReaderException** **COULD_NOT_READ_DATA**
Thrown if an error was encountered while reading the terrain data.
- static **TerrainDataReaderException** **DATA_RESOURCE_NOT_FOUND**
Thrown when a particular data file was not found.
- static **TerrainDataReaderException** **READER_PLUGIN_NOT_FOUND**
Thrown if a third party data reader plugin could not be found.

Protected Member Functions

- **TerrainDataReaderException** (const std::string &name)
Simple enumeration constructor.

4.35.1 Detailed Description

These exceptions are used by terrain data readers when an error occurs during the load process.

4.35.2 Constructor & Destructor Documentation

4.35.2.1 TerrainDataReaderException (const std::string & name) [inline, protected]

Simple enumeration constructor.

4.35.3 Member Data Documentation

4.35.3.1 TerrainDataReaderException COULD_NOT_READ_DATA [static]

Thrown if an error was encountered while reading the terrain data.

4.35.3.2 TerrainDataReaderException DATA_RESOURCE_NOT_FOUND [static]

Thrown when a particular data file was not found.

4.35.3.3 TerrainDataReaderException READER_PLUGIN_NOT_FOUND [static]

Thrown if a third party data reader plugin could not be found.

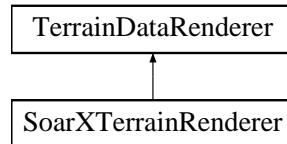
The documentation for this class was generated from the following files:

- **terraindatareader.h**
- **terraindatareader.cpp**

4.36 TerrainDataRenderer Class Reference

This class is the interface for a terrain renderer.

`#include <inc/dtTerrain/terraindatarenderer.h>`Inheritance diagram for TerrainDataRenderer::



Public Member Functions

- **TerrainDataRenderer** (const std::string &name="TerrainRenderer")
Constructs the terrain renderer.
- virtual float **GetHeight** (float x, float y)=0
Gets the height of the terrain at the specified (x,y) coordinates.
- virtual osg::Vec3 **GetNormal** (float x, float y)=0
Gets the normal vector at the specified point.
- const **Terrain** * **GetParentTerrain** () const
Gets a read-only terrain object that currently owns this reader.
- **Terrain** * **GetParentTerrain** ()
Gets the terrain object that currently owns this reader.
- virtual osg::Group * **GetRootDrawable** ()=0
Returns a scene node that encapsulates the renderable terrain.
- virtual void **OnLoadTerrainTile** (**PagedTerrainTile** &tile)=0
This method is called when the parent terrain needs to load a tile.
- virtual void **OnUnloadTerrainTile** (**PagedTerrainTile** &tile)
This method is called when the parent terrain wishes to unload a terrain tile from its list of resident tiles.

Protected Member Functions

- virtual ~**TerrainDataRenderer** ()
Empty destructor...

Friends

- class **Terrain**
Allow the terrain to have access to this class.

4.36.1 Detailed Description

This class is the interface for a terrain renderer. A terrain renderer handles rendering of the terrain. It can access the terrain reader through its parent terrain handle if data paging and such is required. This class enables different terrain rendering algorithms to operate transparently and independent of the underlying terrain data so they can be interchanged when needed.

4.36.2 Constructor & Destructor Documentation

4.36.2.1 TerrainDataRenderer (const std::string & name = "TerrainRenderer")

Constructs the terrain renderer.

4.36.2.2 ~TerrainDataRenderer () [protected, virtual]

Empty destructor...

4.36.3 Member Function Documentation

4.36.3.1 virtual float GetHeight (float x, float y) [pure virtual]

Gets the height of the terrain at the specified (x,y) coordinates. Returns The height of the terrain at the specified point.

Implemented in [SoarXTerrainRenderer](#) (p. 71).

4.36.3.2 virtual osg::Vec3 GetNormal (float x, float y) [pure virtual]

Gets the normal vector at the specified point. Returns A vector perpendicular to terrain at the given point.

Implemented in [SoarXTerrainRenderer](#) (p. 71).

4.36.3.3 const Terrain* GetParentTerrain () const [inline]

Gets a read-only terrain object that currently owns this reader. Returns A const pointer to the parent terrain.

4.36.3.4 Terrain* GetParentTerrain () [inline]

Gets the terrain object that currently owns this reader. Returns A pointer to the parent terrain.

4.36.3.5 virtual osg::Group* GetRootDrawable () [pure virtual]

Returns a scene node that encapsulates the renderable terrain. This is the entry point by which the terrain is added to the scene. Returns An OpenSceneGraph group node containing the renderer's scene graph. Note It is OK to return a group node with no children. This is most likely the case since this method is called before any tiles are inserted in the terrain's load queue.

Implemented in [SoarXTerrainRenderer](#) (p. 72).

4.36.3.6 virtual void OnLoadTerrainTile (PagedTerrainTile & tile) [pure virtual]

This method is called when the parent terrain needs to load a tile. Parameters

tile The new tile. The loader should populate the fields of the tile as appropriate.

Note Should throw an exception if any errors occur.

At this point, the terrain tiles have had a chance to load data from their cache. Therefore, readers implementing this method may want to check the status of a tile's data before loading since it may have been retrieved from the cache.

Resources may be looked up using the parent terrain's resource path list. In most cases, the terrain will be made aware of any resource locations its readers may need. See also [Terrain](#) (p. 74)

[PagedTerrainTile](#) (p. 53)

Implemented in [SoarXTerrainRenderer](#) (p. 72).

4.36.3.7 virtual void OnUnloadTerrainTile (PagedTerrainTile & tile) [inline, virtual]

This method is called when the parent terrain wishes to unload a terrain tile from its list of resident tiles. Parameters

tile The tile being unloaded.

Note Should throw an exception if any errors occur.

The default implementation does nothing as most renderers will probably only process data and have no need to respond to this method, however, it is available if needed. Also note, that the tile itself is responsible for caching its data, however, any data a renderer wishes to control may be cached in this method. See also [Terrain](#) (p. 74)

PagedTerrainTile (p. 53)

Reimplemented in **SoarXTerrainRenderer** (p. 72).

4.36.4 Friends And Related Function Documentation

4.36.4.1 friend class Terrain [friend]

Allow the terrain to have access to this class.

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

- **terraindatarenderer.h**
- **terraindatarenderer.cpp**

4.37 TerrainDataType Class Reference

This class defines the various types of terrain data supported by default in Delta3D.

```
#include <inc/dtTerrain/terraindatatype.h>
```

Static Public Attributes

- static const **TerrainDataType DTED**
DTED levels 0,1,2.

Protected Member Functions

- **TerrainDataType** (const std::string &name)
Simple constructor for enumerations.

4.37.1 Detailed Description

This class defines the various types of terrain data supported by default in Delta3D. In order to support your own formats, you must extend this enumeration and return the appropriate type in your data readers and renderers.

4.37.2 Constructor & Destructor Documentation

4.37.2.1 TerrainDataType (const std::string & name) [inline, protected]

Simple constructor for enumerations.

4.37.3 Member Data Documentation

4.37.3.1 const TerrainDataType DTED [static]

DTED levels 0,1,2.

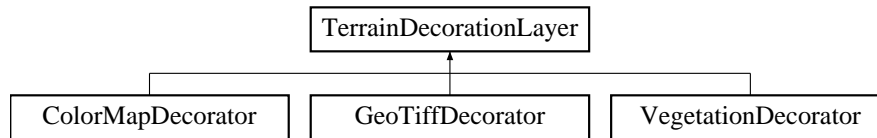
The documentation for this class was generated from the following files:

- **terraindatatype.h**
- **terraindatatype.cpp**

4.38 TerrainDecorationLayer Class Reference

This class is a terrain decoration layer.

#include <inc/dtTerrain/terraindecorationlayer.h> Inheritance diagram for TerrainDecorationLayer::



Public Member Functions

- **TerrainDecorationLayer** (const std::string &name="TerrainDecorationLayer")
Constructs a new terrain decoration layer.
- virtual osg::Node * **GetOSGNode** ()=0
This method returns the root scene node of the decoration layer scene graph.
- const dtTerrain::Terrain * **GetParentTerrain** () const
Gets a read-only terrain object that currently owns this reader.
- dtTerrain::Terrain * **GetParentTerrain** ()
Gets the terrain object that currently owns this reader.
- bool **IsVisible** () const
Checks to see if this decoration layer is visible in the scene.
- virtual void **OnLoadTerrainTile** (PagedTerrainTile &tile)=0
This method is called when the parent terrain needs to load a tile.
- virtual void **OnTerrainTileResident** (PagedTerrainTile &tile)
This method is called when a tile has been completely loaded by the terrain reader, terrain renderer, and all decorator layers.
- virtual void **OnUnloadTerrainTile** (PagedTerrainTile &tile)
This method is called when the parent terrain wishes to unload a terrain tile from its list of resident tiles.
- void **SetVisible** (bool show)
Sets the view status of this decoration layer.

Protected Member Functions

- virtual ~**TerrainDecorationLayer** ()
Destroys this decoration layer.

Friends

- class **Terrain**
Allow the terrain to have access to this class.

4.38.1 Detailed Description

This class is a terrain decoration layer. It provides the interface for providing layers of decorations on the terrain from trees and shrubs to roads and buildings. Note that any number of these layers may be added to the terrain so one may create many layers each loading or manipulating an entirely different data set for that particular layer.

4.38.2 Constructor & Destructor Documentation

4.38.2.1 TerrainDecorationLayer (const std::string & *name* = "TerrainDecorationLayer")

Constructs a new terrain decoration layer.

4.38.2.2 ~TerrainDecorationLayer () [protected, virtual]

Destroys this decoration layer.

4.38.3 Member Function Documentation

4.38.3.1 virtual osg::Node* GetOSGNode () [pure virtual]

This method returns the root scene node of the decoration layer scene graph. This node then gets added as a child to its parent terrain. Returns Any OpenSceneGraph node. Most likely a group node, but it could be any type of node.

Implemented in **ColorMapDecorator** (p. 11), **GeoTiffDecorator** (p. 28), and **VegetationDecorator** (p. 97).

4.38.3.2 const dtTerrain::Terrain* GetParentTerrain () const [inline]

Gets a read-only terrain object that currently owns this reader. Returns A const pointer to the parent terrain.

4.38.3.3 dtTerrain::Terrain* GetParentTerrain () [inline]

Gets the terrain object that currently owns this reader. Returns A pointer to the parent terrain.

4.38.3.4 bool IsVisible () const [inline]

Checks to see if this decoration layer is visible in the scene. Returns True if this layer is currently visible, false otherwise.

4.38.3.5 virtual void OnLoadTerrainTile (PagedTerrainTile & *tile*) [pure virtual]

This method is called when the parent terrain needs to load a tile. Parameters

tile The new tile. The loader should populate the fields of the tile as appropriate.

Note Should throw an exception if any errors occur.

At this point, the terrain tiles have had a chance to load data from their cache. Therefore, readers implementing this method may want to check the status of a tile's data before loading since it may have been retrieved from the cache.

Resources may be looked up using the parent terrain's resource path list. In most cases, the terrain will be made aware of any resource locations its readers may need. See also **Terrain** (p. 74)

PagedTerrainTile (p. 53)

Implemented in **ColorMapDecorator** (p. 12), **GeoTiffDecorator** (p. 28), and **VegetationDecorator** (p. 98).

4.38.3.6 virtual void OnTerrainTileResident (PagedTerrainTile & *tile*) [inline, virtual]

This method is called when a tile has been completely loaded by the terrain reader, terrain renderer, and all decorator layers. This is useful if certain operations need to be performed on a tile that is dependent on the tile being fully loaded by all terrain components. Parameters

The tile that was just loaded.

Reimplemented in **VegetationDecorator** (p. 98).

4.38.3.7 virtual void OnUnloadTerrainTile (PagedTerrainTile & *tile*) [inline, virtual]

This method is called when the parent terrain wishes to unload a terrain tile from its list of resident tiles. Parameters

tile The tile being unloaded.

Note Should throw an exception if any errors occur.

The default implementation does nothing as most renderers will probably only process data and have no need to respond to this method, however, it is available if needed. Also note, that the tile itself is responsible for caching its data, however, any data a renderer wishes to control may be cached in this method. See also **Terrain** (p. 74)

PagedTerrainTile (p. 53)

Reimplemented in **VegetationDecorator** (p. 98).

4.38.3.8 void SetVisible (bool *show*)

Sets the view status of this decoration layer. If set to false, it will not show up on the terrain. Parameters

show If true, the decoration layer will be visible.

4.38.4 Friends And Related Function Documentation

4.38.4.1 friend class Terrain [friend]

Allow the terrain to have access to this class.

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

- **terraindecorationlayer.h**
- **terraindecorationlayer.cpp**

4.39 TerrainException Class Reference

This class enumerates the different exceptions that can be thrown by terrain instances.

```
#include <inc/dtTerrain/terrain.h>
```

Static Public Attributes

- static **TerrainException INVALID_DATA_READER**
Thrown if the terrain requires a data reader but it is currently invalid.
- static **TerrainException INVALID_DATA_RENDERER**
Thrown if the terrain encounters an invalid data renderer.
- static **TerrainException INVALID_DECORATION_LAYER**
Thrown if an invalid decoration layer is added to the terrain.
- static **TerrainException INVALID_RESOURCE_PATH**
Thrown if a specified resource could not be read or loaded.
- static **TerrainException NULL_POINTER**
Thrown if an invalid pointer was encountered during terrain operations.
- static **TerrainException UNSUPPORTED_DATA_FORMAT**
Thrown if the current data reader does not support the specified resource.

Protected Member Functions

- **TerrainException** (const std::string &name)
Simple enumeration constructor.

4.39.1 Detailed Description

This class enumerates the different exceptions that can be thrown by terrain instances.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 TerrainException (const std::string & name) [inline, protected]

Simple enumeration constructor.

4.39.3 Member Data Documentation

4.39.3.1 TerrainException INVALID_DATA_READER [static]

Thrown if the terrain requires a data reader but it is currently invalid.

4.39.3.2 TerrainException INVALID_DATA_RENDERER [static]

Thrown if the terrain encounters an invalid data renderer.

4.39.3.3 TerrainException INVALID_DECORATION_LAYER [static]

Thrown if an invalid decoration layer is added to the terrain.

4.39.3.4 TerrainException INVALID_RESOURCE_PATH [static]

Thrown if a specified resource could not be read or loaded.

4.39.3.5 TerrainException NULL_POINTER [static]

Thrown if an invalid pointer was encountered during terrain operations.

4.39.3.6 TerrainException UNSUPPORTED_DATA_FORMAT [static]

Thrown if the current data reader does not support the specified resource.

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

- **terrain.h**
- **terrain.cpp**

4.40 TerrainRendererException Class Reference

This enumeration contains generic errors that could occur in terrain renderers.

```
#include <inc/dtTerrain/terraindatarenderer.h>
```

Static Public Attributes

- static **TerrainRendererException INVALID_HEIGHTFIELD_DATA**
Thrown if the heightfield data is invalid when the renderer initializes itself.

Protected Member Functions

- **TerrainRendererException** (const std::string &name)
Simple enumeration constructor.

4.40.1 Detailed Description

This enumeration contains generic errors that could occur in terrain renderers. Subclass this enumeration for specific error handling identification.

4.40.2 Constructor & Destructor Documentation

4.40.2.1 TerrainRendererException (const std::string & name) [inline, protected]

Simple enumeration constructor.

4.40.3 Member Data Documentation

4.40.3.1 TerrainRendererException INVALID_HEIGHTFIELD_DATA [static]

Thrown if the heightfield data is invalid when the renderer initializes itself.

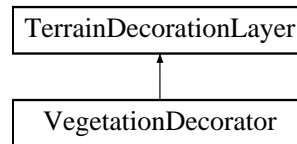
The documentation for this class was generated from the following files:

- **terraindatarenderer.h**
- **terraindatarenderer.cpp**

4.41 VegetationDecorator Class Reference

This class is the vegetation decorator layer.

#include <inc/dtTerrain/vegetationdecorator.h> Inheritance diagram for VegetationDecorator::



Public Types

- typedef std::map< dtCore::RefPtr< **PagedTerrainTile** >, dtCore::RefPtr< osg::Group > > **Vegetation-Map**
Help minimize some typing...

Public Member Functions

- **VegetationDecorator** (const std::string &name="VegetationDecoratorLayer")
Constructs a new terrain decoration layer.
- dtCore::RefPtr< osg::Group > **AddVegetation** (**PagedTerrainTile** &tile)
Place the vegetation into cell specified by lat-long.
- **LCCAnalyzer** & **GetLCCAnalyzer** ()
Gets a reference to the LCCAnalyzer.
- int **GetNumLooks** (const osg::Image &compositelImage, const osg::Image &slopeMap, int x, int y, float good_angle, int maxlooks, float maxslope)
- virtual osg::Node * **GetOSGNode** ()
Gets the root scenegraph node for the vegetation.
- bool **GetVegetation** (osg::Image &mCimage, int x, int y, int limit)
Determine whether vegetation exists at coord x,y.
- int **GetVegType** (const osg::Image *mCimage, int x, int y, float good_angle)
Determine type/age of vegetation type (1-3; young-old).
- virtual void **OnLoadTerrainTile** (**PagedTerrainTile** &tile)
Calculates various LCC images and procedurally builds a list of trees and other vegetation models which are then placed about the specified terrain tile.
- virtual void **OnTerrainTileResident** (**PagedTerrainTile** &tile)
Places the actual vegetation geometry on the terrain.
- virtual void **OnUnloadTerrainTile** (**PagedTerrainTile** &tile)
Removes the vegetation models from the map of currently visible tiles/vegetation.
- void **SetGeospecificImage** (const std::string &geolImageFilename)
- void **SetLCCTypes** (std::vector< dtTerrain::LCCType > &lccTypes)
- void **SetMaxVegetationPerCell** (int newMax)
Sets the limit on the number of vegetation models placed per terrain tile.
- void **SetRandomSeed** (const int &seed)

- void **SetVegetationDistance** (const float distance)

Sets the distance between vegetation objects.

Protected Member Functions

- virtual **~VegetationDecorator** ()

Destructor.

- dtCore::RefPtr< osg::Group > **BuildVegetationForType** (osg::Image &compositelImage, osg::Image &slopeMap, osg::Vec3 &cellOrigin, **LCCType** &type)

4.41.1 Detailed Description

This class is the vegetation decorator layer. It is responsible for processing the probabilities of vegetation placement based on LCC data recieved from the **LCCAnalyzer** (p. 42).

4.41.2 Member Typedef Documentation

4.41.2.1 typedef std::map<dtCore::RefPtr<PagedTerrainTile>,dtCore::RefPtr<osg::Group> > VegetationMap

Help minimize some typing...

4.41.3 Constructor & Destructor Documentation

4.41.3.1 VegetationDecorator (const std::string & name = "VegetationDecoratorLayer")

Constructs a new terrain decoration layer.

4.41.3.2 ~VegetationDecorator () [protected, virtual]

Destructor.

4.41.4 Member Function Documentation

4.41.4.1 dtCore::RefPtr< osg::Group > AddVegetation (PagedTerrainTile & tile)

Place the vegetation into cell specified by lat-long. Parameters

The terrain tile with which to place vegetation.

Returns A group node containing the vegetation scene for the specified tile.

4.41.4.2 dtCore::RefPtr< osg::Group > BuildVegetationForType (osg::Image & compositelImage, osg::Image & slopeMap, osg::Vec3 & cellOrigin, LCCType & type) [protected]

4.41.4.3 LCCAnalyzer& GetLCCAnalyzer () [inline]

Gets a reference to the LCCAnalyzer. Returns The **LCCAnalyzer** (p. 42) for this vegetation decorator.

4.41.4.4 int GetNumLooks (const osg::Image & compositelImage, const osg::Image & slopeMap, int x, int y, float good_angle, int maxlooks, float maxslope)

4.41.4.5 virtual osg::Node* GetOSGNode () [inline, virtual]

Gets the root scenegraph node for the vegetation. Returns A group node holding the vegetation hierarchy for the current visible set of tiles.

Implements **TerrainDecorationLayer** (p. 91).

4.41.4.6 bool GetVegetation (osg::Image & mCImage, int x, int y, int limit)

Determine whether vegetation exists at coord x,y. Parameters

mCImage the LCC type's combined image)

x the x coordinate to check

y the y coordinate to check

limit the probability rolled

Returns boolean on existence of vegetaton at x,y

4.41.4.7 int GetVegType (const osg::Image * mCImage, int x, int y, float good_angle)

Determine type/age of vegetation type (1-3; young-old). Parameters

mCImage the LCC type's combined image)

x the x coordinate to check

y the y coordinate to check

pref_angle the preferred angle of aspect in degrees for optimum growth

Returns the age bias of the vegetation type

4.41.4.8 void OnLoadTerrainTile (PagedTerrainTile & tile) [virtual]

Calculates various LCC images and procedurally builds a list of trees and other vegetation models which are then placed about the specified terrain tile.

Implements **TerrainDecorationLayer** (p. 91).

4.41.4.9 void OnTerrainTileResident (PagedTerrainTile & tile) [virtual]

Places the actual vegetation geometry on the terrain. Note, the vegetation placement occurs here since it is dependent on querying for the height of the terrain in order to place the vegetation. The terrain height can only be queried once it has been loaded by the renderer. Performing this here ensures that the renderer has loaded this tile's data.

Reimplemented from **TerrainDecorationLayer** (p. 91).

4.41.4.10 void OnUnloadTerrainTile (PagedTerrainTile & tile) [virtual]

Removes the vegetation models from the map of currently visible tiles/vegetation.

Reimplemented from **TerrainDecorationLayer** (p. 92).

4.41.4.11 void SetGeospecificImage (const std::string & geoImageFilename) [inline]**4.41.4.12 void SetLCCTypes (std::vector< dtTerrain::LCCType > & lccTypes) [inline]****4.41.4.13 void SetMaxVegetationPerCell (int newMax) [inline]**

Sets the limit on the number of vegetation models placed per terrain tile. Parameters

newMax The new limit.

4.41.4.14 void SetRandomSeed (const int & seed) [inline]**4.41.4.15 void SetVegetationDistance (const float distance) [inline]**

Sets the distance between vegetation objects. Parameters

distance between vegetation objects

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

- **vegetationdecorator.h**
- **vegetationdecorator.cpp**

4.42 VegetationException Class Reference

Defines the exception used by the vegetation decorator.

```
#include <inc/dtTerrain/vegetationdecorator.h>
```

Static Public Attributes

- static **VegetationException INVALID_LCC_TYPES**
Thrown if no LCC data was specified before analyzing occurs.
- static **VegetationException INVALID_SLOPE_ASPECT_IMAGE**
Thrown if the slope-aspect image could not be loaded when processing vegetation placement.

Protected Member Functions

- **VegetationException** (const std::string &name)

4.42.1 Detailed Description

Defines the exception used by the vegetation decorator.

4.42.2 Constructor & Destructor Documentation

4.42.2.1 VegetationException (const std::string & name) [inline, protected]

4.42.3 Member Data Documentation

4.42.3.1 VegetationException INVALID_LCC_TYPES [static]

Thrown if no LCC data was specified before analyzing occurs.

4.42.3.2 VegetationException INVALID_SLOPE_ASPECT_IMAGE [static]

Thrown if the slope-aspect image could not be loaded when processing vegetation placement.

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

- **vegetationdecorator.h**
- **vegetationdecorator.cpp**

4.43 Vertex Struct Reference

This is the structure for a single processed vertex in the terrain.

```
#include <inc/dtTerrain/soarxdrawable.h>
```

Public Member Functions

- **Vertex (Index i)**
- **Vertex ()**

Public Attributes

- float **error**
- **Index index**
- osg::Vec3 **position**
- float **radius**

4.43.1 Detailed Description

This is the structure for a single processed vertex in the terrain.

4.43.2 Constructor & Destructor Documentation

4.43.2.1 **Vertex ()** [inline]

4.43.2.2 **Vertex (Index i)** [inline]

4.43.3 Member Data Documentation

4.43.3.1 **float error**

4.43.3.2 **Index index**

4.43.3.3 **osg::Vec3 position**

4.43.3.4 **float radius**

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

- **soarxdrawable.h**

File Documentation

5.1 colormapdecorator.cpp File Reference

```
#include <dtTerrain/colormapdecorator.h>
#include <dtUtil/log.h>
#include <dtUtil/fileutils.h>
#include <osgDB/WriteFile>
#include <osgDB/ReadFile>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Variables

- const std::string **COLOR_MAP_IMAGE_NAME** = "colormap_decorator_image.rgb"

5.2 colormapdecorator.h File Reference

```
#include "dtTerrain/terraindecorationlayer.h"  
#include "dtTerrain/imageutils.h"  
#include "dtTerrain/terrain_export.h"
```

Classes

- class **ColorMapDecorator**
This terrain decorator is a terrain tile base texture decorator.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.3 dtedterrainreader.cpp File Reference

```
#include <sstream>
#include <iomanip>
#include <algorithm>
#include <cctype>
#include <dtCore/scene.h>
#include <dtUtil/log.h>
#include <dtUtil/fileutils.h>
#include <osgDB/ReadFile>
#include <osgDB/WriteFile>
#include <osgDB/FileUtils>
#include <osgDB/FileNameUtils>
#include <dtTerrain/terraindatareader.h>
#include <dtTerrain/terraindatarenderer.h>
#include <dtTerrain/terraindecorationlayer.h>
#include <dtTerrain/dtedterrainreader.h>
#include <dtTerrain/imageutils.h>
#include <dtTerrain/terrain.h>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (DTEDTerrainReader::DTEDLevelEnum)

5.4 dtedterrainreader.h File Reference

```
#include <osg/Shape>
#include "dtTerrain/terraindatareader.h"
#include "dtTerrain/heightfield.h"
```

Classes

- class **DTEDLevelEnum**
This enumeration represents the different DTED levels available for loading.
- class **DTEDTerrainReader**
This data reader loads DTED data mapping to a specified latitude and longitude.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.5 fixedpointnoise.cpp File Reference

```
#include "dtTerrain/fixedpointnoise.h"
```

```
#include "dtTerrain/mathutils.h"
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Variables

- const int **I** = (1 << 12)

5.6 fixedpointnoise.h File Reference

Classes

- class **FixedPointNoise**

This is a fixed point implementation of an improved Ken Perlin noise generator.

Namespaces

- namespace **dtTerrain**

An extensible library for rendering terrain databases.

5.7 geocoordinates.cpp File Reference

```
#include "dtTerrain/geocoordinates.h"
#include "dtUtil/exception.h"
#include <osg/Math>
#include <iostream>
#include <sstream>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (GeoCoordinatesException)
- std::string **vec3dToString** (const osg::Vec3d &pt)
- std::string **vec3ToString** (const osg::Vec3 &pt)

5.8 geocoordinates.h File Reference

```
#include <osg/Vec3>
#include <osg/Vec3d>
#include <dtUtil/enumeration.h>
#include <dtTerrain/terrain_export.h>
```

Classes

- class **GeoCoordinates**
This class is intended to represent cartesian and geographic coordinates.
- class **GeoCoordinatesException**
This class contains the various types of exceptions that the coordinate system class may throw during a given operation.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.9 geotiffdecorator.cpp File Reference

```
#include <dtTerrain/geotiffdecorator.h>
#include <osg/Image>
#include <osg/Math>
#include <dtCore/refptr.h>
#include <dtUtil/log.h>
#include <dtUtil/fileutils.h>
#include <osgDB/ReadFile>
#include <osgDB/WriteFile>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Variables

- const std::string **GEOTIFF_IMAGE_NAME** = "geo_tiff_decorator_image.rgb"

5.10 geotiffdecorator.h File Reference

```
#include <vector>
#include <osg/Image>
#include "dtTerrain/imageutils.h"
#include "dtTerrain/terraindecorationlayer.h"
#include "dtTerrain/terrain_export.h"
```

Classes

- class **GeoTiffDecorator**

This class is a terrain texturing decorator that supports the mapping of geospecific images onto terrain tiles.

Namespaces

- namespace **dtTerrain**

An extensible library for rendering terrain databases.

5.11 heightfield.cpp File Reference

```
#include "dtTerrain/heightfield.h"  
#include "dtUtil/exception.h"  
#include <iostream>  
#include <sstream>  
#include <cstring>  
#include <climits>  
#include <osgDB/WriteFile>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (HeightFieldException)

5.12 heightfield.h File Reference

```
#include <vector>
#include <osg/Referenced>
#include <osg/Image>
#include "dtUtil/enumeration.h"
```

Classes

- class **HeightField**
This class wraps an array of elevation posts.
- class **HeightFieldException**
This enumeration defines the different types of exceptions the height field may throw if an error occurs.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.13 imageutils.cpp File Reference

```
#include <sstream>
#include <osg/Vec3>
#include <osg/Texture2D>
#include <dtCore/refptr.h>
#include <dtCore/physical.h>
#include <dtCore/scene.h>
#include <dtUtil/log.h>
#include <osgDB/ReadFile>
#include <ogrsf_frmts.h>
#include <gdal_priv.h>
#include <gdalwarper.h>
#include <dtUtil/exception.h>
#include <dtTerrain/imageutils.h>
#include <dtTerrain/mathutils.h>
#include <dtTerrain/fixedpointnoise.h>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (ImageUtilException)

5.14 imageutils.h File Reference

```
#include <osg/Vec3>
#include <dtCore/refptr.h>
#include <dtTerrain/heightfield.h>
#include <dtTerrain/terrain_export.h>
#include <map>
```

Classes

- struct **GeospecificImage**
- struct **HeightColorMap**
Maps height values to colors with interpolation/extrapolation.
- class **ImageUtilException**
Defines exceptions that may be thrown from the image utility methods.
- class **ImageUtils**
This class is a static class containing many methods for creating images procedurally, concatenating images, and building images from source height data.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.15 lccanalyzer.cpp File Reference

```
#include <dtCore/refptr.h>
#include <dtCore/deltadrawable.h>
#include <dtCore/scene.h>
#include <osg/Image>
#include <osg/Math>
#include <osgDB/ImageOptions>
#include <osgDB/ReadFile>
#include <osgDB/WriteFile>
#include <dtUtil/log.h>
#include <dtUtil/exception.h>
#include <dtUtil/fileutils.h>
#include <dtTerrain/mathutils.h>
#include <dtTerrain/imageutils.h>
#include <dtTerrain/lccanalyzer.h>
#include <ogrsf_frmts.h>
#include <gdal_priv.h>
#include <gdalwarper.h>
#include <sstream>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (LCCAnalyzerResourceName)
- **IMPLEMENT_ENUM** (LCCAnalyzerException)

5.16 Iccanalyzer.h File Reference

```
#include <string>
#include <osg/Vec3>
#include <osg/Image>
#include "dtTerrain/terrain_export.h"
#include "dtTerrain/imageutils.h"
#include "dtTerrain/lcctype.h"
#include "dtTerrain/pagedterraintile.h"
```

Classes

- class **LCCAnalyzer**
The LCC Analyzer calculates LCC data for use in the vegetation decorator.
- class **LCCAnalyzerException**
Defines the exceptions thrown by the lcc analyzer.
- class **LCCAnalyzerResourceName**
This enumeration identifies the resources that are cached and managed by the vegetation decorator layer.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.17 lcctype.cpp File Reference

```
#include <dtTerrain/lcctype.h>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.18 lcctype.h File Reference

```
#include <map>
#include <string>
#include <osg/Node>
#include <dtCore/refptr.h>
#include <dtTerrain/terrain_export.h>
```

Classes

- struct **LCCModel**
Describes an LCC's geometric model info.
- class **LCCType**

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.19 mainpage.h File Reference

5.19.1 Detailed Description

This file contains Doxygen special commands and text for the **Main Page** (p. ??) and some other minor aspects of this documentation. It is not part of Delta3D.

5.20 mathutils.cpp File Reference

```
#include "dtTerrain/mathutils.h"
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.21 mathutils.h File Reference

```
#include <osg/Math>
```

```
#include "dtTerrain/terrain_export.h"
```

Classes

- class **MathUtils**

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.22 pagedterrintile.cpp File Reference

```
#include <dtTerrain/pagedterrintile.h>
#include <dtTerrain/heightfield.h>
#include <dtTerrain/terrain.h>
#include <dtTerrain/terraindatareader.h>
#include <dtTerrain/terraindatarenderer.h>
#include <dtTerrain/terraindecorationlayer.h>
#include <dtUtil/fileutils.h>
#include <dtUtil/exception.h>
#include <dtCore/scene.h>
#include <iostream>
#include <fstream>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (PagedTerrainTileLoadState)
- **IMPLEMENT_ENUM** (PagedTerrainTileResourceName)

5.23 pagedterraintile.h File Reference

```
#include <string>
#include <osg/Referenced>
#include <osg/Image>
#include <dtCore/refptr.h>
#include <dtCore/uniqueid.h>
#include "dtUtil/enumeration.h"
#include "dtTerrain/terrain_export.h"
#include "dtTerrain/geocoordinates.h"
#include "dtTerrain/heightfield.h"
```

Classes

- class **PagedTerrainTile**
This class is the base class for a tile of data.
- class **PagedTerrainTileLoadState**
This enumeration defines the different states a tile could be in throughout its lifetime.
- class **PagedTerrainTileResourceName**
This enumeration is used to identity built in resources that the terrain tiles are aware of.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.24 pagedterraintilefactory.h File Reference

```
#include <osg/Referenced>
#include "dtTerrain/terrain_export.h"
#include "dtTerrain/geocoordinates.h"
#include "dtTerrain/pagedterraintile.h"
```

Classes

- class **PagedTerrainTileFactory**

This class is the base for a simple factory used to create new terrain tiles.

Namespaces

- namespace **dtTerrain**

An extensible library for rendering terrain databases.

5.25 soarxdrawable.cpp File Reference

```
#include <osg/NodeVisitor>
#include <dtTerrain/soarxdrawable.h>
#include <dtTerrain/imageutils.h>
#include <dtTerrain/mathutils.h>
#include <dtTerrain/terrain.h>
#include <dtUtil/log.h>
#include <dtCore/scene.h>
#include <osg/Image>
#include <osg/io_utils>
#include <osgDB/WriteFile>
#include <osg/Texture2D>
#include <iostream>
#include <sstream>
#include <dtTerrain/terraindecorationlayer.h>
#include <dtTerrain/terraindatareader.h>
#include <dtTerrain/soarxterrainrenderer.h>
#include <dtTerrain/heightfield.h>
```

Classes

- struct **SoarXCullCallback**

This callback is used by the SoarX renderer to update the camera's position or eyepoint which is needed by the refinement process to calculate projected vertex errors.

Namespaces

- namespace **dtTerrain**

An extensible library for rendering terrain databases.

Variables

- const bool **BOTTOM** = false

Constants used during the refinement process.

- const bool **SIDE** = true

5.26 soarxdrawable.h File Reference

```
#include <osg/Geometry>
#include <dtCore/refptr.h>
#include "dtTerrain/pagedterraintile.h"
#include "dtTerrain/terrain_export.h"
```

Classes

- struct **Index**
This structure represents a location on the terrain's 2D grid.
- struct **RawData**
This structure is the raw data on disk.
- class **SoarXDrawable**
This class encapsulates the SoarX rendering algorithm itself.
- struct **Vertex**
This is the structure for a single processed vertex in the terrain.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.27 soarxterrainrenderer.cpp File Reference

```
#include <sstream>
#include <dtCore/scene.h>
#include <dtCore/globals.h>
#include <osg/Group>
#include <osg/Shape>
#include <osg/Uniform>
#include <osg/Program>
#include <osg/StateSet>
#include <osg/Texture2D>
#include <osg/TexGen>
#include <osg/PolygonMode>
#include <osg/CullFace>
#include <osg/FrontFace>
#include <osg/io_utils>
#include <osgDB/WriteFile>
#include <osgDB/ReadFile>
#include <dtUtil/fileutils.h>
#include <dtTerrain/terraindatareader.h>
#include <dtTerrain/terraindecorationlayer.h>
#include <dtTerrain/terrain.h>
#include <dtTerrain/soarxterrainrenderer.h>
#include <dtTerrain/imageutils.h>
#include <dtTerrain/soarxdrawable.h>
#include <dtTerrain/geocoordinates.h>
#include <dtTerrain/pagedterrintile.h>
#include <dtTerrain/heightfield.h>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (SoarXCacheResourceName)

5.28 soarxterrainrenderer.h File Reference

```
#include <osg/Image>
#include <osg/StateSet>
#include <osg/Program>
#include <osg/MatrixTransform>
#include <dtTerrain/terraindatarenderer.h>
#include <dtTerrain/soarxdrawable.h>
```

Classes

- struct **DrawableEntry**
This structure represents a drawable tile.
- class **SoarXCacheResourceName**
- class **SoarXTerrainRenderer**
This renderer is an integration/implementation of the SoarX terrain rendering algorithm.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.29 terrain.cpp File Reference

```
#include <osgDB/FileUtils>
#include <osg/MatrixTransform>
#include <dtCore/scene.h>
#include <dtCore/system.h>
#include <dtUtil/fileutils.h>
#include <dtUtil/exception.h>
#include <dtCore/collisioncategorydefaults.h>
#include <dtTerrain/terrain.h>
#include <dtTerrain/terraindatareader.h>
#include <dtTerrain/terraindatarenderer.h>
#include <dtTerrain/terraindecorationlayer.h>
#include <dtTerrain/vegetationdecorator.h>
#include <dtTerrain/pagedterraintile.h>
#include <dtTerrain/heightfield.h>
#include <sstream>
#include <algorithm>
```

Classes

- class **TerrainCullCallback**

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (TerrainException)
- **IMPLEMENT_MANAGEMENT_LAYER** (Terrain)
- bool **SimpleLineOfSight** (**dtTerrain::Terrain** *terrain, const osg::Vec3 &pointOne, const osg::Vec3 &pointTwo)

Trivial Line Of Sight calculator.

5.30 terrain.h File Reference

```
#include <map>
#include <string>
#include <queue>
#include <list>
#include <dtCore/physical.h>
#include <dtUtil/enumeration.h>
#include <dtTerrain/geocoordinates.h>
#include <dtTerrain/pagedterraitilefactory.h>
#include <dtTerrain/terrain_export.h>
```

Classes

- class **Terrain**

This is the base terrain class that provides most of the functionality required for terrain rendering.

- class **TerrainException**

This class enumerates the different exceptions that can be thrown by terrain instances.

Namespaces

- namespace **dtTerrain**

An extensible library for rendering terrain databases.

Functions

- bool **SimpleLineOfSight** (**dtTerrain::Terrain** *terrain, const osg::Vec3 &pointOne, const osg::Vec3 &pointTwo)

Trivial Line Of Sight calculator.

5.31 terrain_export.h File Reference

Defines

- #define **DT_TERRAIN_EXPORT**

This is modeled from the DT_EXPORT macro found in dtCore/export.h.

5.31.1 Define Documentation

5.31.1.1 #define DT_TERRAIN_EXPORT

This is modeled from the DT_EXPORT macro found in dtCore/export.h. We define another due to conflicts with using the DT_EXPORT while trying to import Delta3D symbols. The DT_TERRAIN_EXPORT macro should be used in front of any classes that are to be exported from the terrain library. Also note that DT_TERRAIN_LIBRARY should be defined in the compiler preprocessor defines.

5.32 terraindatareader.cpp File Reference

```
#include <dtCore/scene.h>
#include <sstream>
#include <climits>
#include <iomanip>
#include <dtTerrain/terraindatarender.h>
#include <dtTerrain/terraindatareader.h>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (HeightFieldResizePolicy)
- **IMPLEMENT_ENUM** (TerrainDataReaderException)

5.33 terraindatareader.h File Reference

```
#include <osg/Shape>
#include <dtCore/base.h>
#include <dtCore/refptr.h>
#include <dtUtil/enumeration.h>
#include <dtTerrain/terraindatatype.h>
#include <dtTerrain/pagedterrintile.h>
```

Classes

- class **HeightFieldResizePolicy**
This enumeration specifies the valid resize policies the data readers should follow when loading a new heightfield.
- class **TerrainDataReader**
This is mostly an interface class to the data reader functionality of Delta3D terrains.
- class **TerrainDataReaderException**
These exceptions are used by terrain data readers when an error occurs during the load process.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.34 terraindatarenderer.cpp File Reference

```
#include "dtTerrain/terraindatarenderer.h"
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (TerrainRendererException)

5.35 terraindatarenderer.h File Reference

```
#include <osg/Vec3>
#include <osg/Group>
#include "dtCore/base.h"
#include "dtCore/refptr.h"
#include "dtUtil/exception.h"
#include "dtTerrain/pagedterrintile.h"
#include "dtTerrain/terrain_export.h"
```

Classes

- class **TerrainDataRenderer**
This class is the interface for a terrain renderer.
- class **TerrainRendererException**
This enumeration contains generic errors that could occur in terrain renderers.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.36 terraindatatype.cpp File Reference

```
#include "dtTerrain/terraindatatype.h"
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (TerrainDataType)

5.37 terraindatatype.h File Reference

```
#include "dtUtil/enumeration.h"
```

```
#include "dtTerrain/terrain_export.h"
```

Classes

- class **TerrainDataType**

This class defines the various types of terrain data supported by default in Delta3D.

Namespaces

- namespace **dtTerrain**

An extensible library for rendering terrain databases.

5.38 terraindecorationlayer.cpp File Reference

```
#include <dtCore/scene.h>
#include "dtTerrain/terraindecorationlayer.h"
#include "dtTerrain/terraindatarenderer.h"
#include "dtTerrain/terraindatareader.h"
#include "dtTerrain/terrain.h"
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.39 terraindecorationlayer.h File Reference

```
#include <osg/Node>
#include "dtCore/base.h"
#include "dtCore/refptr.h"
#include "dtTerrain/pagedterraintile.h"
#include "dtTerrain/terrain_export.h"
```

Classes

- class **TerrainDecorationLayer**
This class is a terrain decoration layer.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

5.40 vegetationdecorator.cpp File Reference

```
#include <dtUtil/fileutils.h>
#include <dtCore/refptr.h>
#include <dtCore/scene.h>
#include <dtTerrain/terraindatareader.h>
#include <dtTerrain/terraindatarenderer.h>
#include <dtTerrain/terrain.h>
#include <dtTerrain/vegetationdecorator.h>
#include <dtTerrain/lccanalyzer.h>
#include <osg/io_utils>
#include <osgDB/ReadFile>
#include <osgDB/WriteFile>
#include <osgDB/ImageOptions>
#include <osg/PositionAttitudeTransform>
#include <osgUtil/TriStripVisitor>
#include <osg/Texture2D>
#include <dtTerrain/soarxterrainrenderer.h>
#include <dtTerrain/lcctype.h>
#include <sstream>
#include <cmath>
```

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Functions

- **IMPLEMENT_ENUM** (VegetationException)

5.41 vegetationdecorator.h File Reference

```
#include <osg/Vec3>
#include <osg/Group>
#include <osg/Image>
#include <dtCore/refptr.h>
#include <dtUtil/enumeration.h>
#include <dtTerrain/terraindecorationlayer.h>
#include <dtTerrain/lccanalyzer.h>
#include <dtTerrain/lcctype.h>
```

Classes

- class **VegetationDecorator**
This class is the vegetation decorator layer.
- class **VegetationException**
Defines the exception used by the vegetation decorator.

Namespaces

- namespace **dtTerrain**
An extensible library for rendering terrain databases.

Index

- Symbols -

- ~DTEDTerrainReader
 - dtTerrain::DTEDTerrainReader, 17
- ~GeoTiffDecorator
 - dtTerrain::GeoTiffDecorator, 27
- ~HeightField
 - dtTerrain::HeightField, 31
- ~LCCAnalyzer
 - dtTerrain::LCCAnalyzer, 43
- ~LCCType
 - dtTerrain::LCCType, 50
- ~PagedTerrainTile
 - dtTerrain::PagedTerrainTile, 54
- ~PagedTerrainTileFactory
 - dtTerrain::PagedTerrainTileFactory, 57
- ~SoarXDrawable
 - dtTerrain::SoarXDrawable, 65
- ~SoarXTerrainRenderer
 - dtTerrain::SoarXTerrainRenderer, 71
- ~Terrain
 - dtTerrain::Terrain, 76
- ~TerrainDataReader
 - dtTerrain::TerrainDataReader, 82
- ~TerrainDataRenderer
 - dtTerrain::TerrainDataRenderer, 87
- ~TerrainDecorationLayer
 - dtTerrain::TerrainDecorationLayer, 91
- ~VegetationDecorator
 - dtTerrain::VegetationDecorator, 97

- A -

- Active
 - dtTerrain::SoarXDrawable, 66
- AddDecorationLayer
 - dtTerrain::Terrain, 76
- AddGeoSpecificImage
 - dtTerrain::GeoTiffDecorator, 27
- AddGeospecificImage
 - dtTerrain::LCCAnalyzer, 43
- AddModel
 - dtTerrain::LCCType, 50
- AddResourcePath
 - dtTerrain::Terrain, 76
- AddVegetation
 - dtTerrain::VegetationDecorator, 97
- Allocate
 - dtTerrain::HeightField, 31
- Append
 - dtTerrain::SoarXDrawable, 66
- ApplyMask
 - dtTerrain::ImageUtils, 37

- B -

- BASE_COLOR
 - dtTerrain::LCCAnalyzerResourceName, 47
- BASE_FILTER_NAME
 - dtTerrain::LCCAnalyzerResourceName, 47
- BASE_GRADIENT_TEXTURE
 - dtTerrain::SoarXCacheResourceName, 61
- BASE_LCC_COLOR

- dtTerrain::LCCAnalyzerResourceName, 47
- baseGradientTexture
 - dtTerrain::SoarXTerrainRenderer::DrawableEntry, 13
- BOTTOM
 - dtTerrain, 10
- Build
 - dtTerrain::SoarXDrawable, 66
- BuildVegetationForType
 - dtTerrain::VegetationDecorator, 97

- C -

- CalculateBaseImage
 - dtTerrain::GeoTiffDecorator, 27
- CalculateDetailNoise
 - dtTerrain::ImageUtils, 37
 - dtTerrain::SoarXTerrainRenderer, 71
- CalculateError
 - dtTerrain::SoarXDrawable, 66
- CalculateRadii
 - dtTerrain::SoarXDrawable, 66
- CalculateScaleMapNoise
 - dtTerrain::ImageUtils, 37
- CalculateVertexErrors
 - dtTerrain::SoarXDrawable, 66
- CalculateVertexErrorsHelper
 - dtTerrain::SoarXDrawable, 66
- CheckBaseGradientCache
 - dtTerrain::SoarXTerrainRenderer, 71
- CheckBaseLCCImages
 - dtTerrain::LCCAnalyzer, 43
- CheckChildren1
 - dtTerrain::SoarXDrawable, 66
- CheckChildren2
 - dtTerrain::SoarXDrawable, 66
- CheckDetailGradientCache
 - dtTerrain::SoarXTerrainRenderer, 71
- CheckDetailNoiseCache
 - dtTerrain::SoarXTerrainRenderer, 71
- CheckDetailScaleCache
 - dtTerrain::SoarXTerrainRenderer, 71
- CheckSlopeAndElevationMaps
 - dtTerrain::LCCAnalyzer, 43
- clamp
 - dtTerrain::SoarXDrawable::Index, 41
- Clear
 - dtTerrain::LCCAnalyzer, 43
 - dtTerrain::SoarXDrawable, 66
- ClearDecorationLayers
 - dtTerrain::Terrain, 76
- ClearModels
 - dtTerrain::LCCType, 50
- clone
 - dtTerrain::SoarXDrawable, 66
- cloneType
 - dtTerrain::SoarXDrawable, 66
- COLOR_MAP_IMAGE_NAME
 - dtTerrain, 10
- ColorMapDecorator
 - dtTerrain::ColorMapDecorator, 11
- colormapdecorator.cpp, 101

- colormapdecorator.h, 102
- COMPOSITE_LCC_IMAGE
 - dtTerrain::LCCAnalyzerResourceName, 47
- computeBound
 - dtTerrain::SoarXDrawable, 66
- ComputeProbabilityMap
 - dtTerrain::LCCAnalyzer, 43
- ConvertFromImage
 - dtTerrain::HeightField, 31
- ConvertFromRaw
 - dtTerrain::HeightField, 31
- ConvertHeightField
 - dtTerrain::TerrainDataReader, 82
- ConvertToImage
 - dtTerrain::HeightField, 31
- COULD_NOT_READ_DATA
 - dtTerrain::TerrainDataReaderException, 85
- CreateBaseGradientMap
 - dtTerrain::ImageUtils, 37
- CreateDetailGradientMap
 - dtTerrain::ImageUtils, 37
- CreateDetailMap
 - dtTerrain::ImageUtils, 37
- CreateDetailScaleMap
 - dtTerrain::ImageUtils, 37
- CreateFragmentShader
 - dtTerrain::SoarXTerrainRenderer, 71
- CreateNewTile
 - dtTerrain::PagedTerrainTileFactory, 57
- CreateTerrainTile
 - dtTerrain::Terrain, 76
- cull
 - dtTerrain::SoarXCullCallback, 62
- D -**
- DATA_RESOURCE_NOT_FOUND
 - dtTerrain::TerrainDataReaderException, 85
- DETAIL_GRADIENT_TEXTURE
 - dtTerrain::SoarXCacheResourceName, 61
- DETAIL_SCALE_MAP
 - dtTerrain::SoarXCacheResourceName, 61
- DETAIL_VERTEX_NOISE
 - dtTerrain::SoarXCacheResourceName, 61
- drawable
 - dtTerrain::SoarXTerrainRenderer::DrawableEntry, 13
- DrawableMap
 - dtTerrain::SoarXTerrainRenderer, 71
- drawImplementation
 - dtTerrain::SoarXDrawable, 66
- DT_TERRAIN_EXPORT
 - terrain_export.h, 131
- DTED
 - dtTerrain::TerrainDataType, 89
- DTEDTerrainReader
 - dtTerrain::DTEDTerrainReader, 17
- dtedterrainreader.cpp, 103
- dtedterrainreader.h, 104
- dtTerrain, 7
 - BOTTOM, 10
 - COLOR_MAP_IMAGE_NAME, 10
 - GEOTIFF_IMAGE_NAME, 10
 - I, 10
 - IMPLEMENT_ENUM, 10
 - IMPLEMENT_MANAGEMENT_LAYER, 10
 - SIDE, 10
 - SimpleLineOfSight, 10
 - vec3dToString, 10
 - vec3ToString, 10
- dtTerrain::ColorMapDecorator, 11
 - ColorMapDecorator, 11
 - GetOSGNode, 11
 - OnLoadTerrainTile, 11
- dtTerrain::DTEDTerrainReader, 16
 - ~DTEDTerrainReader, 17
 - DTEDTerrainReader, 17
 - GenerateTerrainTileCachePath, 17
 - GetDataType, 17
 - GetDTEDFilePath, 17
 - GetDTEDLevel, 17
 - OnLoadTerrainTile, 17
 - SetDTEDLevel, 17
- dtTerrain::DTEDTerrainReader::DTEDLevelEnum, 14
 - FIVE, 14
 - FOUR, 14
 - GetNumeral, 14
 - ONE, 14
 - THREE, 14
 - TWO, 14
 - ZERO, 14
- dtTerrain::FixedPointNoise, 19
 - operator(), 19
- dtTerrain::GeoCoordinates, 20
 - EQUATORIAL_RADIUS, 23
 - FLATTENING, 23
 - GeoCoordinates, 21
 - geoOrigin, 23
 - GetAltitude, 21
 - GetCartesianPoint, 21, 22
 - GetLatitude, 22
 - GetLongitude, 22
 - GetOrigin, 22
 - GetRawCartesianPoint, 22
 - gOriginOffset, 23
 - mAltitude, 23
 - mCartesianPoint, 23
 - mLatitude, 23
 - mLongitude, 23
 - mUpdateCartesianPoint, 23
 - operator<, 22
 - operator==, 22
 - POLAR_RADIUS, 24
 - SetAltitude, 22
 - SetCartesianPoint, 22
 - SetLatitude, 22, 23
 - SetLongitude, 23
 - SetOrigin, 23
 - ToString, 23
 - ToStringAll, 23
- dtTerrain::GeoCoordinatesException, 25
 - GeoCoordinatesException, 25
 - OUT_OF_BOUNDS, 25
- dtTerrain::GeoTiffDecorator, 27
 - ~GeoTiffDecorator, 27
 - AddGeoSpecificImage, 27
 - CalculateBaselImage, 27
 - GeoTiffDecorator, 27
 - GetOSGNode, 28
 - LoadAllGeoSpecificImages, 28
 - OnLoadTerrainTile, 28
 - SetResultingImageDimensions, 28
- dtTerrain::HeightField, 30
 - ~HeightField, 31

- Allocate, 31
- ConvertFromImage, 31
- ConvertFromRaw, 31
- ConvertToImage, 31
- GetHeight, 32
- GetHeightFieldData, 32
- GetInterpolatedHeight, 32
- GetNumColumns, 32
- GetNumRows, 32
- GetXInterval, 32
- GetYInterval, 32
- HeightField, 31
- SetHeight, 32
- SetXInterval, 32
- SetYInterval, 32
- dtTerrain::HeightFieldException, 33
 - INVALID_HEIGHTFIELD, 33
 - INVALID_IMAGE_FORMAT, 33
 - OUT_OF_BOUNDS, 33
- dtTerrain::HeightFieldResizePolicy, 34
 - HeightFieldResizePolicy, 34
 - NEAREST_POWER_OF_TWO, 34
 - NEAREST_POWER_OF_TWO_PLUS_ONE, 34
 - NONE, 34
- dtTerrain::ImageUtilException, 35
 - ImageUtilException, 35
 - INVALID_IMAGE_DIMENSIONS, 35
 - INVALID_RASTER_FORMAT, 35
 - LOAD_FAILED, 35
- dtTerrain::ImageUtils, 36
 - ApplyMask, 37
 - CalculateDetailNoise, 37
 - CalculateScaleMapNoise, 37
 - CreateBaseGradientMap, 37
 - CreateDetailGradientMap, 37
 - CreateDetailMap, 37
 - CreateDetailScaleMap, 37
 - EnsurePow2Image, 38
 - ImageStats, 38
 - LoadGeospecificLCCImage, 38
 - MakeBaseColor, 38
 - MakeFilteredImage, 38
 - MakeRelativeElevationImage, 38
 - MakeSlopeAspectImage, 38
- dtTerrain::ImageUtils::GeospecificImage, 26
 - mFileName, 26
 - mGeoTransform, 26
 - mImage, 26
 - mInverseGeoTransform, 26
 - mMaxLatitude, 26
 - mMaxLongitude, 26
 - mMinLatitude, 26
 - mMinLongitude, 26
- dtTerrain::ImageUtils::HeightColorMap, 29
 - GetColor, 29
- dtTerrain::LCCAnalyzer, 42
 - ~LCCAnalyzer, 43
 - AddGeospecificImage, 43
 - CheckBaseLCCImages, 43
 - CheckSlopeAndElevationMaps, 43
 - Clear, 43
 - ComputeProbabilityMap, 43
 - GenerateBaseFilterImage, 43
 - GetImageSize, 43
 - IsGeoSpecificLCCImageLoaded, 43
 - LCCAnalyzer, 43
 - LCCHistogram, 43
 - LoadAllGeoSpecificImages, 43
 - MakeBaseLCCColor, 44
 - MakeCombinedImage, 44
 - MakeLCCMask, 44
 - OutputDebugImages, 44
 - ProcessLCCData, 44
 - SetMaxImageSize, 44
 - SetOutputDebugImages, 44
- dtTerrain::LCCAnalyzerException, 46
 - INVALID_CACHE, 46
 - LCCAnalyzerException, 46
 - NO_VALID_GEO_IMAGES, 46
- dtTerrain::LCCAnalyzerResourceName, 47
 - BASE_COLOR, 47
 - BASE_FILTER_NAME, 47
 - BASE_LCC_COLOR, 47
 - COMPOSITE_LCC_IMAGE, 47
 - IMAGE_EXT, 47
 - LCC_IMAGE_NAME, 47
 - LCCAnalyzerResourceName, 47
 - REL_ELEV_IMAGE, 47
 - SCENE_GRAPH, 47
 - SLOPE_IMAGE, 47
 - WATER_MASK, 47
- dtTerrain::LCCType, 49
 - ~LCCType, 50
 - AddModel, 50
 - ClearModels, 50
 - GetAspect, 50
 - GetIndex, 50
 - GetLCCName, 50
 - GetMaxElevation, 50
 - GetMaxSlope, 50
 - GetMinElevation, 50
 - GetMinSlope, 50
 - GetModel, 50
 - GetNumberOfModels, 50
 - GetRGB, 51
 - LCCType, 50
 - RemoveModel, 51
 - SetAspect, 51
 - SetElevation, 51
 - SetRelativeElevation, 51
 - SetRGB, 51
 - SetSlope, 51
- dtTerrain::LCCType::LCCModel, 48
 - LCCModel, 48
 - name, 48
 - scale, 48
 - sceneNode, 48
- dtTerrain::MathUtils, 52
 - F2F, 52
 - F2I, 52
 - Fade, 52
 - I2F, 52
 - ILerp, 52
 - IMul, 52
 - Lerp, 52
- dtTerrain::PagedTerrainTile, 53
 - ~PagedTerrainTile, 54
 - GetBaseTextureImage, 54
 - GetCachePath, 54
 - GetGeoCoordinates, 54
 - GetHeightField, 54
 - GetParentTerrain, 54

- GetUpdateCache, 54
- IsCachingEnabled, 54
- PagedTerrainTile, 54
- PagedTerrainTileFactory, 55
- ReadFromCache, 55
- SetBaseTextureImage, 55
- SetCachePath, 55
- SetGeoCoordinates, 55
- SetHeightField, 55
- SetUpdateCache, 55
- Terrain, 55
- WriteToCache, 55
- dtTerrain::PagedTerrainTileFactory, 57
 - ~PagedTerrainTileFactory, 57
 - CreateNewTile, 57
 - PagedTerrainTileFactory, 57
- dtTerrain::PagedTerrainTileLoadState, 58
 - LOADED, 58
 - NOT_LOADED, 58
 - PagedTerrainTileLoadState, 58
 - PAGING, 58
- dtTerrain::PagedTerrainTileResourceName, 59
 - HEIGHTFIELD_FILENAME, 59
 - PagedTerrainTileResourceName, 59
- dtTerrain::SoarXCacheResourceName, 61
 - BASE_GRADIENT_TEXTURE, 61
 - DETAIL_GRADIENT_TEXTURE, 61
 - DETAIL_SCALE_MAP, 61
 - DETAIL_VERTEX_NOISE, 61
 - SoarXCacheResourceName, 61
 - VERTEX_DATA, 61
- dtTerrain::SoarXCullCallback, 62
 - cull, 62
- dtTerrain::SoarXDrawable, 63
 - ~SoarXDrawable, 65
 - Active, 66
 - Append, 66
 - Build, 66
 - CalculateError, 66
 - CalculateRadii, 66
 - CalculateVertexErrors, 66
 - CalculateVertexErrorsHelper, 66
 - CheckChildren1, 66
 - CheckChildren2, 66
 - Clear, 66
 - clone, 66
 - cloneType, 66
 - computeBound, 66
 - drawImplementation, 66
 - GetBaseHorizontalResolution, 66
 - GetBaseVerticalResolution, 66
 - GetDetailHorizontalResolution, 67
 - GetDetailMultiplier, 67
 - GetDetailSize, 67
 - GetDetailVerticalResolution, 67
 - GetHeight, 67
 - GetRawData, 67
 - GetThreshold, 67
 - GetVertex, 67
 - LeftRefine, 67
 - Refine, 67
 - Render, 67
 - Repair, 67
 - RepairBoundingSphereHierarchy, 67
 - RestoreDataFromCache, 67
 - RightRefine, 67
 - SetDetailMultiplier, 67
 - SetDetailNoise, 68
 - SetEyePoint, 68
 - SetThreshold, 68
 - SoarXCullCallback, 68
 - SoarXDrawable, 65
 - supports, 68
 - TurnCorner, 68
 - WriteDataToCache, 68
- dtTerrain::SoarXDrawable::Index, 40
 - clamp, 41
 - Index, 40, 41
 - operator<, 41
 - operator<=<=, 41
 - operator>, 41
 - operator>=>=, 41
 - operator+&, 41
 - operator-&, 41
 - operator&=, 41
 - q, 41
 - x, 41
 - y, 41
- dtTerrain::SoarXDrawable::RawData, 60
 - error, 60
 - height, 60
 - radius, 60
 - scale, 60
- dtTerrain::SoarXDrawable::Vertex, 100
 - error, 100
 - index, 100
 - position, 100
 - radius, 100
 - Vertex, 100
- dtTerrain::SoarXTerrainRenderer, 69
 - ~SoarXTerrainRenderer, 71
 - CalculateDetailNoise, 71
 - CheckBaseGradientCache, 71
 - CheckDetailGradientCache, 71
 - CheckDetailNoiseCache, 71
 - CheckDetailScaleCache, 71
 - CreateFragmentShader, 71
 - DrawableMap, 71
 - GetDetailMultiplier, 71
 - GetHeight, 71
 - GetNormal, 71
 - GetRootDrawable, 71
 - GetTerrainFragmentShader, 72
 - GetThreshold, 72
 - GRADIENT_SCALE, 73
 - InitializeRenderer, 72
 - OnLoadTerrainTile, 72
 - OnUnloadTerrainTile, 72
 - SetDetailMultiplier, 72
 - SetEnableFog, 72
 - SetTerrainFragmentShader, 72
 - SetThreshold, 72
 - SetupRenderState, 72
 - SoarXTerrainRenderer, 71
- dtTerrain::SoarXTerrainRenderer::DrawableEntry, 13
 - baseGradientTexture, 13
 - drawable, 13
 - sceneNode, 13
- dtTerrain::Terrain, 74
 - ~Terrain, 76
 - AddDecorationLayer, 76
 - AddResourcePath, 76

- ClearDecorationLayers, 76
 - CreateTerrainTile, 76
 - EnsureTileVisibility, 76
 - FindAllResources, 76
 - FindResource, 76
 - GetCachePath, 77
 - GetDataReader, 77
 - GetDataRenderer, 77
 - GetDecorationLayer, 77
 - GetDecorationLayers, 77
 - GetHeight, 77
 - GetLineOfSightSpacing, 77
 - GetLoadDistance, 77
 - GetNumDecorationLayers, 77
 - GetResourcePathList, 77
 - GetTerrainTileFactory, 78
 - HideDecorationLayer, 78
 - IsClearLineOfSight, 78
 - IsTerrainTileResident, 78
 - LoadTerrainTile, 78
 - mResidentTiles, 79
 - mTilesToLoadQ, 79
 - mTilesToUnloadQ, 79
 - OnMessage, 78
 - PostFrame, 78
 - PreFrame, 78
 - RemoveDecorationLayer, 78
 - RemoveResourcePath, 78
 - SetCachePath, 78
 - SetDataReader, 79
 - SetDataRenderer, 79
 - SetLineOfSightSpacing, 79
 - SetLoadDistance, 79
 - SetTerrainTileFactory, 79
 - ShowDecorationLayer, 79
 - Terrain, 76
 - UnloadAllTerrainTiles, 79
 - UnloadTerrainTile, 79
 - dtTerrain::TerrainCullCallback, 80
 - operator(), 80
 - TerrainCullCallback, 80
 - dtTerrain::TerrainDataReader, 81
 - ~TerrainDataReader, 82
 - ConvertHeightField, 82
 - GenerateTerrainTileCachePath, 82
 - GetDataType, 82
 - GetInterpolatedHeight, 82
 - GetParentTerrain, 82, 83
 - GetResizePolicy, 83
 - OnLoadTerrainTile, 83
 - OnUnloadTerrainTile, 83
 - ScaleOSGHeightField, 83
 - SetResizePolicy, 83
 - Terrain, 84
 - TerrainDataReader, 82
 - dtTerrain::TerrainDataReaderException, 85
 - COULD_NOT_READ_DATA, 85
 - DATA_RESOURCE_NOT_FOUND, 85
 - READER_PLUGIN_NOT_FOUND, 85
 - TerrainDataReaderException, 85
 - dtTerrain::TerrainDataRenderer, 86
 - ~TerrainDataRenderer, 87
 - GetHeight, 87
 - GetNormal, 87
 - GetParentTerrain, 87
 - GetRootDrawable, 87
 - OnLoadTerrainTile, 87
 - OnUnloadTerrainTile, 87
 - Terrain, 88
 - TerrainDataRenderer, 87
 - dtTerrain::TerrainDataType, 89
 - DTED, 89
 - TerrainDataType, 89
 - dtTerrain::TerrainDecorationLayer, 90
 - ~TerrainDecorationLayer, 91
 - GetOSGNode, 91
 - GetParentTerrain, 91
 - IsVisible, 91
 - OnLoadTerrainTile, 91
 - OnTerrainTileResident, 91
 - OnUnloadTerrainTile, 91
 - SetVisible, 92
 - Terrain, 92
 - TerrainDecorationLayer, 91
 - dtTerrain::TerrainException, 93
 - INVALID_DATA_READER, 93
 - INVALID_DATA_RENDERER, 93
 - INVALID_DECORATION_LAYER, 93
 - INVALID_RESOURCE_PATH, 93
 - NULL_POINTER, 93
 - TerrainException, 93
 - UNSUPPORTED_DATA_FORMAT, 93
 - dtTerrain::TerrainRendererException, 95
 - INVALID_HEIGHTFIELD_DATA, 95
 - TerrainRendererException, 95
 - dtTerrain::VegetationDecorator, 96
 - ~VegetationDecorator, 97
 - AddVegetation, 97
 - BuildVegetationForType, 97
 - GetLCCAnalyzer, 97
 - GetNumLooks, 97
 - GetOSGNode, 97
 - GetVegetation, 97
 - GetVegType, 98
 - OnLoadTerrainTile, 98
 - OnTerrainTileResident, 98
 - OnUnloadTerrainTile, 98
 - SetGeospecificImage, 98
 - SetLCCTypes, 98
 - SetMaxVegetationPerCell, 98
 - SetRandomSeed, 98
 - SetVegetationDistance, 98
 - VegetationDecorator, 97
 - VegetationMap, 97
 - dtTerrain::VegetationException, 99
 - INVALID_LCC_TYPES, 99
 - INVALID_SLOPE_ASPECT_IMAGE, 99
 - VegetationException, 99
- E -**
- EnsurePow2Image
 - dtTerrain::ImageUtils, 38
 - EnsureTileVisibility
 - dtTerrain::Terrain, 76
 - EQUATORIAL_RADIUS
 - dtTerrain::GeoCoordinates, 23
 - error
 - dtTerrain::SoarXDrawable::RawData, 60

- dtTerrain::SoarXDrawable::Vertex, 100
- F -**
- F2F
 - dtTerrain::MathUtils, 52
- F2I
 - dtTerrain::MathUtils, 52
- Fade
 - dtTerrain::MathUtils, 52
- FindAllResources
 - dtTerrain::Terrain, 76
- FindResource
 - dtTerrain::Terrain, 76
- FIVE
 - dtTerrain::DTEDTerrainReader::DTEDLevelEnum, 14
- fixedpointnoise.cpp, 105
- fixedpointnoise.h, 106
- FLATTENING
 - dtTerrain::GeoCoordinates, 23
- FOUR
 - dtTerrain::DTEDTerrainReader::DTEDLevelEnum, 14
- G -**
- GenerateBaseFilterImage
 - dtTerrain::LCCAnalyzer, 43
- GenerateTerrainTileCachePath
 - dtTerrain::DTEDTerrainReader, 17
 - dtTerrain::TerrainDataReader, 82
- GeoCoordinates
 - dtTerrain::GeoCoordinates, 21
- geocoordinates.cpp, 107
- geocoordinates.h, 108
- GeoCoordinatesException
 - dtTerrain::GeoCoordinatesException, 25
- geoOrigin
 - dtTerrain::GeoCoordinates, 23
- GEOTIFF_IMAGE_NAME
 - dtTerrain, 10
- GeoTiffDecorator
 - dtTerrain::GeoTiffDecorator, 27
- geotiffdecorator.cpp, 109
- geotiffdecorator.h, 110
- GetAltitude
 - dtTerrain::GeoCoordinates, 21
- GetAspect
 - dtTerrain::LCCType, 50
- GetBaseHorizontalResolution
 - dtTerrain::SoarXDrawable, 66
- GetBaseTextureImage
 - dtTerrain::PagedTerrainTile, 54
- GetBaseVerticalResolution
 - dtTerrain::SoarXDrawable, 66
- GetCachePath
 - dtTerrain::PagedTerrainTile, 54
 - dtTerrain::Terrain, 77
- GetCartesianPoint
 - dtTerrain::GeoCoordinates, 21, 22
- GetColor
 - dtTerrain::ImageUtils::HeightColorMap, 29
- GetDataReader
 - dtTerrain::Terrain, 77
- GetDataRenderer
 - dtTerrain::Terrain, 77
- GetDataType
 - dtTerrain::DTEDTerrainReader, 17
- dtTerrain::TerrainDataReader, 82
- GetDecorationLayer
 - dtTerrain::Terrain, 77
- GetDecorationLayers
 - dtTerrain::Terrain, 77
- GetDetailHorizontalResolution
 - dtTerrain::SoarXDrawable, 67
- GetDetailMultiplier
 - dtTerrain::SoarXDrawable, 67
 - dtTerrain::SoarXTerrainRenderer, 71
- GetDetailSize
 - dtTerrain::SoarXDrawable, 67
- GetDetailVerticalResolution
 - dtTerrain::SoarXDrawable, 67
- GetDTEDFilePath
 - dtTerrain::DTEDTerrainReader, 17
- GetDTEDLevel
 - dtTerrain::DTEDTerrainReader, 17
- GetGeoCoordinates
 - dtTerrain::PagedTerrainTile, 54
- GetHeight
 - dtTerrain::HeightField, 32
 - dtTerrain::SoarXDrawable, 67
 - dtTerrain::SoarXTerrainRenderer, 71
 - dtTerrain::Terrain, 77
 - dtTerrain::TerrainDataRenderer, 87
- GetHeightField
 - dtTerrain::PagedTerrainTile, 54
- GetHeightFieldData
 - dtTerrain::HeightField, 32
- GetImageSize
 - dtTerrain::LCCAnalyzer, 43
- GetIndex
 - dtTerrain::LCCType, 50
- GetInterpolatedHeight
 - dtTerrain::HeightField, 32
 - dtTerrain::TerrainDataReader, 82
- GetLatitude
 - dtTerrain::GeoCoordinates, 22
- GetLCCAnalyzer
 - dtTerrain::VegetationDecorator, 97
- GetLCCName
 - dtTerrain::LCCType, 50
- GetLineOfSightSpacing
 - dtTerrain::Terrain, 77
- GetLoadDistance
 - dtTerrain::Terrain, 77
- GetLongitude
 - dtTerrain::GeoCoordinates, 22
- GetMaxElevation
 - dtTerrain::LCCType, 50
- GetMaxSlope
 - dtTerrain::LCCType, 50
- GetMinElevation
 - dtTerrain::LCCType, 50
- GetMinSlope
 - dtTerrain::LCCType, 50
- GetModel
 - dtTerrain::LCCType, 50
- GetNormal
 - dtTerrain::SoarXTerrainRenderer, 71
 - dtTerrain::TerrainDataRenderer, 87
- GetNumberOfModels
 - dtTerrain::LCCType, 50
- GetNumColumns
 - dtTerrain::HeightField, 32

- GetNumDecorationLayers
 - dtTerrain::Terrain, 77
- GetNumeral
 - dtTerrain::DTEDTerrainReader::DTEDLevelEnum, 14
- GetNumLooks
 - dtTerrain::VegetationDecorator, 97
- GetNumRows
 - dtTerrain::HeightField, 32
- GetOrigin
 - dtTerrain::GeoCoordinates, 22
- GetOSGNode
 - dtTerrain::ColorMapDecorator, 11
 - dtTerrain::GeoTiffDecorator, 28
 - dtTerrain::TerrainDecorationLayer, 91
 - dtTerrain::VegetationDecorator, 97
- GetParentTerrain
 - dtTerrain::PagedTerrainTile, 54
 - dtTerrain::TerrainDataReader, 82, 83
 - dtTerrain::TerrainDataRenderer, 87
 - dtTerrain::TerrainDecorationLayer, 91
- GetRawCartesianPoint
 - dtTerrain::GeoCoordinates, 22
- GetRawData
 - dtTerrain::SoarXDrawable, 67
- GetResizePolicy
 - dtTerrain::TerrainDataReader, 83
- GetResourcePathList
 - dtTerrain::Terrain, 77
- GetRGB
 - dtTerrain::LCCType, 51
- GetRootDrawable
 - dtTerrain::SoarXTerrainRenderer, 71
 - dtTerrain::TerrainDataRenderer, 87
- GetTerrainFragmentShader
 - dtTerrain::SoarXTerrainRenderer, 72
- GetTerrainTileFactory
 - dtTerrain::Terrain, 78
- GetThreshold
 - dtTerrain::SoarXDrawable, 67
 - dtTerrain::SoarXTerrainRenderer, 72
- GetUpdateCache
 - dtTerrain::PagedTerrainTile, 54
- GetVegetation
 - dtTerrain::VegetationDecorator, 97
- GetVegType
 - dtTerrain::VegetationDecorator, 98
- GetVertex
 - dtTerrain::SoarXDrawable, 67
- GetXInterval
 - dtTerrain::HeightField, 32
- GetYInterval
 - dtTerrain::HeightField, 32
- gOriginOffset
 - dtTerrain::GeoCoordinates, 23
- GRADIENT_SCALE
 - dtTerrain::SoarXTerrainRenderer, 73
- H -**
- height
 - dtTerrain::SoarXDrawable::RawData, 60
- HeightField
 - dtTerrain::HeightField, 31
- heightfield.cpp, 111
- heightfield.h, 112
- HEIGHTFIELD_FILENAME
 - dtTerrain::PagedTerrainTileResourceName, 59
- HeightFieldResizePolicy
 - dtTerrain::HeightFieldResizePolicy, 34
- HideDecorationLayer
 - dtTerrain::Terrain, 78
- I -**
- I
 - dtTerrain, 10
- I2F
 - dtTerrain::MathUtils, 52
- ILerp
 - dtTerrain::MathUtils, 52
- IMAGE_EXT
 - dtTerrain::LCCAnalyzerResourceName, 47
- ImageStats
 - dtTerrain::ImageUtils, 38
- ImageUtilException
 - dtTerrain::ImageUtilException, 35
- imageutils.cpp, 113
- imageutils.h, 114
- IMPLEMENT_ENUM
 - dtTerrain, 10
- IMPLEMENT_MANAGEMENT_LAYER
 - dtTerrain, 10
- IMul
 - dtTerrain::MathUtils, 52
- inc/ Directory Reference, 5
- inc/dtTerrain/ Directory Reference, 3
- Index
 - dtTerrain::SoarXDrawable::Index, 40, 41
- index
 - dtTerrain::SoarXDrawable::Vertex, 100
- InitializeRenderer
 - dtTerrain::SoarXTerrainRenderer, 72
- INVALID_CACHE
 - dtTerrain::LCCAnalyzerException, 46
- INVALID_DATA_READER
 - dtTerrain::TerrainException, 93
- INVALID_DATA_RENDERER
 - dtTerrain::TerrainException, 93
- INVALID_DECORATION_LAYER
 - dtTerrain::TerrainException, 93
- INVALID_HEIGHTFIELD
 - dtTerrain::HeightFieldException, 33
- INVALID_HEIGHTFIELD_DATA
 - dtTerrain::TerrainRendererException, 95
- INVALID_IMAGE_DIMENSIONS
 - dtTerrain::ImageUtilException, 35
- INVALID_IMAGE_FORMAT
 - dtTerrain::HeightFieldException, 33
- INVALID_LCC_TYPES
 - dtTerrain::VegetationException, 99
- INVALID_RASTER_FORMAT
 - dtTerrain::ImageUtilException, 35
- INVALID_RESOURCE_PATH
 - dtTerrain::TerrainException, 93
- INVALID_SLOPE_ASPECT_IMAGE
 - dtTerrain::VegetationException, 99
- IsCachingEnabled
 - dtTerrain::PagedTerrainTile, 54
- IsClearLineOfSight
 - dtTerrain::Terrain, 78
- IsGeoSpecificLCCImageLoaded
 - dtTerrain::LCCAnalyzer, 43
- IsTerrainTileResident
 - dtTerrain::Terrain, 78

- IsVisible
 - dtTerrain::TerrainDecorationLayer, 91
- L -**
- LCC_IMAGE_NAME
 - dtTerrain::LCCAnalyzerResourceName, 47
- LCCAnalyzer
 - dtTerrain::LCCAnalyzer, 43
- lccanalyzer.cpp, 115
- lccanalyzer.h, 116
- LCCAnalyzerException
 - dtTerrain::LCCAnalyzerException, 46
- LCCAnalyzerResourceName
 - dtTerrain::LCCAnalyzerResourceName, 47
- LCCHistogram
 - dtTerrain::LCCAnalyzer, 43
- LCCModel
 - dtTerrain::LCCType::LCCModel, 48
- LCCType
 - dtTerrain::LCCType, 50
- lcctype.cpp, 117
- lcctype.h, 118
- LeftRefine
 - dtTerrain::SoarXDrawable, 67
- Lerp
 - dtTerrain::MathUtils, 52
- LOAD_FAILED
 - dtTerrain::ImageUtilException, 35
- LoadAllGeoSpecificImages
 - dtTerrain::GeoTiffDecorator, 28
 - dtTerrain::LCCAnalyzer, 43
- LOADED
 - dtTerrain::PagedTerrainTileLoadState, 58
- LoadGeospecificLCCImage
 - dtTerrain::ImageUtils, 38
- LoadTerrainTile
 - dtTerrain::Terrain, 78
- M -**
- mainpage.h, 119
- MakeBaseColor
 - dtTerrain::ImageUtils, 38
- MakeBaseLCCColor
 - dtTerrain::LCCAnalyzer, 44
- MakeCombinedImage
 - dtTerrain::LCCAnalyzer, 44
- MakeFilteredImage
 - dtTerrain::ImageUtils, 38
- MakeLCCMask
 - dtTerrain::LCCAnalyzer, 44
- MakeRelativeElevationImage
 - dtTerrain::ImageUtils, 38
- MakeSlopeAspectImage
 - dtTerrain::ImageUtils, 38
- mAltitude
 - dtTerrain::GeoCoordinates, 23
- mathutils.cpp, 120
- mathutils.h, 121
- mCartesianPoint
 - dtTerrain::GeoCoordinates, 23
- mFileName
 - dtTerrain::ImageUtils::GeospecificImage, 26
- mGeoTransform
 - dtTerrain::ImageUtils::GeospecificImage, 26
- mImage
 - dtTerrain::ImageUtils::GeospecificImage, 26
- mInverseGeoTransform
 - dtTerrain::ImageUtils::GeospecificImage, 26
- mLatitude
 - dtTerrain::GeoCoordinates, 23
- mLongitude
 - dtTerrain::GeoCoordinates, 23
- mMaxLatitude
 - dtTerrain::ImageUtils::GeospecificImage, 26
- mMaxLongitude
 - dtTerrain::ImageUtils::GeospecificImage, 26
- mMinLatitude
 - dtTerrain::ImageUtils::GeospecificImage, 26
- mMinLongitude
 - dtTerrain::ImageUtils::GeospecificImage, 26
- mResidentTiles
 - dtTerrain::Terrain, 79
- mTilesToLoadQ
 - dtTerrain::Terrain, 79
- mTilesToUnloadQ
 - dtTerrain::Terrain, 79
- mUpdateCartesianPoint
 - dtTerrain::GeoCoordinates, 23
- N -**
- name
 - dtTerrain::LCCType::LCCModel, 48
- NEAREST_POWER_OF_TWO
 - dtTerrain::HeightFieldResizePolicy, 34
- NEAREST_POWER_OF_TWO_PLUS_ONE
 - dtTerrain::HeightFieldResizePolicy, 34
- NO_VALID_GEO_IMAGES
 - dtTerrain::LCCAnalyzerException, 46
- NONE
 - dtTerrain::HeightFieldResizePolicy, 34
- NOT_LOADED
 - dtTerrain::PagedTerrainTileLoadState, 58
- NULL_POINTER
 - dtTerrain::TerrainException, 93
- O -**
- ONE
 - dtTerrain::DTEDTerrainReader::DTEDLevelEnum, 14
- OnLoadTerrainTile
 - dtTerrain::ColorMapDecorator, 11
 - dtTerrain::DTEDTerrainReader, 17
 - dtTerrain::GeoTiffDecorator, 28
 - dtTerrain::SoarXTerrainRenderer, 72
 - dtTerrain::TerrainDataReader, 83
 - dtTerrain::TerrainDataRenderer, 87
 - dtTerrain::TerrainDecorationLayer, 91
 - dtTerrain::VegetationDecorator, 98
- OnMessage
 - dtTerrain::Terrain, 78
- OnTerrainTileResident
 - dtTerrain::TerrainDecorationLayer, 91
 - dtTerrain::VegetationDecorator, 98
- OnUnloadTerrainTile
 - dtTerrain::SoarXTerrainRenderer, 72
 - dtTerrain::TerrainDataReader, 83
 - dtTerrain::TerrainDataRenderer, 87
 - dtTerrain::TerrainDecorationLayer, 91
 - dtTerrain::VegetationDecorator, 98
- operator<
 - dtTerrain::GeoCoordinates, 22

dtTerrain::SoarXDrawable::Index, 41
 operator<<=
 dtTerrain::SoarXDrawable::Index, 41
 operator>
 dtTerrain::SoarXDrawable::Index, 41
 operator>>=
 dtTerrain::SoarXDrawable::Index, 41
 operator()
 dtTerrain::FixedPointNoise, 19
 dtTerrain::TerrainCullCallback, 80
 operator+=
 dtTerrain::SoarXDrawable::Index, 41
 operator-=
 dtTerrain::SoarXDrawable::Index, 41
 operator==
 dtTerrain::GeoCoordinates, 22
 operator&=
 dtTerrain::SoarXDrawable::Index, 41
 OUT_OF_BOUNDS
 dtTerrain::GeoCoordinatesException, 25
 dtTerrain::HeightFieldException, 33
 OutputDebugImages
 dtTerrain::LCCAnalyzer, 44
- P -
 PagedTerrainTile
 dtTerrain::PagedTerrainTile, 54
 pagedterraintile.cpp, 122
 pagedterraintile.h, 123
 PagedTerrainTileFactory
 dtTerrain::PagedTerrainTile, 55
 dtTerrain::PagedTerrainTileFactory, 57
 pagedterraintilefactory.h, 124
 PagedTerrainTileLoadState
 dtTerrain::PagedTerrainTileLoadState, 58
 PagedTerrainTileResourceName
 dtTerrain::PagedTerrainTileResourceName, 59
 PAGING
 dtTerrain::PagedTerrainTileLoadState, 58
 POLAR_RADIUS
 dtTerrain::GeoCoordinates, 24
 position
 dtTerrain::SoarXDrawable::Vertex, 100
 PostFrame
 dtTerrain::Terrain, 78
 PreFrame
 dtTerrain::Terrain, 78
 ProcessLCCData
 dtTerrain::LCCAnalyzer, 44
- Q -
 q
 dtTerrain::SoarXDrawable::Index, 41
- R -
 radius
 dtTerrain::SoarXDrawable::RawData, 60
 dtTerrain::SoarXDrawable::Vertex, 100
 READER_PLUGIN_NOT_FOUND
 dtTerrain::TerrainDataReaderException, 85
 ReadFromCache
 dtTerrain::PagedTerrainTile, 55
 Refine
 dtTerrain::SoarXDrawable, 67
 REL_ELEV_IMAGE
 dtTerrain::LCCAnalyzerResourceName, 47
 RemoveDecorationLayer
 dtTerrain::Terrain, 78
 RemoveModel
 dtTerrain::LCCType, 51
 RemoveResourcePath
 dtTerrain::Terrain, 78
 Render
 dtTerrain::SoarXDrawable, 67
 Repair
 dtTerrain::SoarXDrawable, 67
 RepairBoundingSphereHierarchy
 dtTerrain::SoarXDrawable, 67
 RestoreDataFromCache
 dtTerrain::SoarXDrawable, 67
 RightRefine
 dtTerrain::SoarXDrawable, 67
- S -
 scale
 dtTerrain::LCCType::LCCModel, 48
 dtTerrain::SoarXDrawable::RawData, 60
 ScaleOSGHeightField
 dtTerrain::TerrainDataReader, 83
 SCENE_GRAPH
 dtTerrain::LCCAnalyzerResourceName, 47
 sceneNode
 dtTerrain::LCCType::LCCModel, 48
 dtTerrain::SoarXTerrainRenderer::DrawableEntry, 13
 SetAltitude
 dtTerrain::GeoCoordinates, 22
 SetAspect
 dtTerrain::LCCType, 51
 SetBaseTextureImage
 dtTerrain::PagedTerrainTile, 55
 SetCachePath
 dtTerrain::PagedTerrainTile, 55
 dtTerrain::Terrain, 78
 SetCartesianPoint
 dtTerrain::GeoCoordinates, 22
 SetDataReader
 dtTerrain::Terrain, 79
 SetDataRenderer
 dtTerrain::Terrain, 79
 SetDetailMultiplier
 dtTerrain::SoarXDrawable, 67
 dtTerrain::SoarXTerrainRenderer, 72
 SetDetailNoise
 dtTerrain::SoarXDrawable, 68
 SetDTEDLevel
 dtTerrain::DTEDTerrainReader, 17
 SetElevation
 dtTerrain::LCCType, 51
 SetEnableFog
 dtTerrain::SoarXTerrainRenderer, 72
 SetEyePoint
 dtTerrain::SoarXDrawable, 68
 SetGeoCoordinates
 dtTerrain::PagedTerrainTile, 55
 SetGeospecificImage
 dtTerrain::VegetationDecorator, 98
 SetHeight
 dtTerrain::HeightField, 32
 SetHeightField
 dtTerrain::PagedTerrainTile, 55
 SetLatitude

- dtTerrain::GeoCoordinates, 22, 23
 - SetLCCTypes
 - dtTerrain::VegetationDecorator, 98
 - SetLineOfSightSpacing
 - dtTerrain::Terrain, 79
 - SetLoadDistance
 - dtTerrain::Terrain, 79
 - SetLongitude
 - dtTerrain::GeoCoordinates, 23
 - SetMaxImageSize
 - dtTerrain::LCCAnalyzer, 44
 - SetMaxVegetationPerCell
 - dtTerrain::VegetationDecorator, 98
 - SetOrigin
 - dtTerrain::GeoCoordinates, 23
 - SetOutputDebugImages
 - dtTerrain::LCCAnalyzer, 44
 - SetRandomSeed
 - dtTerrain::VegetationDecorator, 98
 - SetRelativeElevation
 - dtTerrain::LCCType, 51
 - SetResizePolicy
 - dtTerrain::TerrainDataReader, 83
 - SetResultingImageDimensions
 - dtTerrain::GeoTiffDecorator, 28
 - SetRGB
 - dtTerrain::LCCType, 51
 - SetSlope
 - dtTerrain::LCCType, 51
 - SetTerrainFragmentShader
 - dtTerrain::SoarXTerrainRenderer, 72
 - SetTerrainTileFactory
 - dtTerrain::Terrain, 79
 - SetThreshold
 - dtTerrain::SoarXDrawable, 68
 - dtTerrain::SoarXTerrainRenderer, 72
 - SetUpdateCache
 - dtTerrain::PagedTerrainTile, 55
 - SetupRenderState
 - dtTerrain::SoarXTerrainRenderer, 72
 - SetVegetationDistance
 - dtTerrain::VegetationDecorator, 98
 - SetVisible
 - dtTerrain::TerrainDecorationLayer, 92
 - SetXInterval
 - dtTerrain::HeightField, 32
 - SetYInterval
 - dtTerrain::HeightField, 32
 - ShowDecorationLayer
 - dtTerrain::Terrain, 79
 - SIDE
 - dtTerrain, 10
 - SimpleLineOfSight
 - dtTerrain, 10
 - SLOPE_IMAGE
 - dtTerrain::LCCAnalyzerResourceName, 47
 - SoarXCacheResourceName
 - dtTerrain::SoarXCacheResourceName, 61
 - SoarXCullCallback
 - dtTerrain::SoarXDrawable, 68
 - SoarXDrawable
 - dtTerrain::SoarXDrawable, 65
 - soarxdrawable.cpp, 125
 - soarxdrawable.h, 126
 - SoarXTerrainRenderer
 - dtTerrain::SoarXTerrainRenderer, 71
 - soarxterrainrenderer.cpp, 127
 - soarxterrainrenderer.h, 128
 - src/ Directory Reference, 6
 - src/dtTerrain/ Directory Reference, 4
 - supports
 - dtTerrain::SoarXDrawable, 68
- T -**
- Terrain
 - dtTerrain::PagedTerrainTile, 55
 - dtTerrain::Terrain, 76
 - dtTerrain::TerrainDataReader, 84
 - dtTerrain::TerrainDataRenderer, 88
 - dtTerrain::TerrainDecorationLayer, 92
 - terrain.cpp, 129
 - terrain.h, 130
 - terrain_export.h, 131
 - DT_TERRAIN_EXPORT, 131
 - TerrainCullCallback
 - dtTerrain::TerrainCullCallback, 80
 - TerrainDataReader
 - dtTerrain::TerrainDataReader, 82
 - terraindatareader.cpp, 132
 - terraindatareader.h, 133
 - TerrainDataReaderException
 - dtTerrain::TerrainDataReaderException, 85
 - TerrainDataRenderer
 - dtTerrain::TerrainDataRenderer, 87
 - terraindatarenderer.cpp, 134
 - terraindatarenderer.h, 135
 - TerrainDataType
 - dtTerrain::TerrainDataType, 89
 - terraindatatype.cpp, 136
 - terraindatatype.h, 137
 - TerrainDecorationLayer
 - dtTerrain::TerrainDecorationLayer, 91
 - terraindecorationlayer.cpp, 138
 - terraindecorationlayer.h, 139
 - TerrainException
 - dtTerrain::TerrainException, 93
 - TerrainRendererException
 - dtTerrain::TerrainRendererException, 95
 - THREE
 - dtTerrain::DTEDTerrainReader::DTEDLevelEnum, 14
 - ToString
 - dtTerrain::GeoCoordinates, 23
 - ToStringAll
 - dtTerrain::GeoCoordinates, 23
 - TurnCorner
 - dtTerrain::SoarXDrawable, 68
 - TWO
 - dtTerrain::DTEDTerrainReader::DTEDLevelEnum, 14
- U -**
- UnloadAllTerrainTiles
 - dtTerrain::Terrain, 79
 - UnloadTerrainTile
 - dtTerrain::Terrain, 79
 - UNSUPPORTED_DATA_FORMAT
 - dtTerrain::TerrainException, 93
- V -**
- vec3dToString
 - dtTerrain, 10
 - vec3ToString

- dtTerrain, 10
- VegetationDecorator
 - dtTerrain::VegetationDecorator, 97
- vegetationdecorator.cpp, 140
- vegetationdecorator.h, 141
- VegetationException
 - dtTerrain::VegetationException, 99
- VegetationMap
 - dtTerrain::VegetationDecorator, 97
- Vertex
 - dtTerrain::SoarXDrawable::Vertex, 100
- VERTEX_DATA
 - dtTerrain::SoarXCacheResourceName, 61
- W -**
- WATER_MASK
 - dtTerrain::LCCAnalyzerResourceName, 47
- WriteDataToCache
 - dtTerrain::SoarXDrawable, 68
- WriteToCache
 - dtTerrain::PagedTerrainTile, 55
- X -**
- x
 - dtTerrain::SoarXDrawable::Index, 41
- Y -**
- y
 - dtTerrain::SoarXDrawable::Index, 41
- Z -**
- ZERO
 - dtTerrain::DTEDTerrainReader::DTEDLevelEnum, 14