Namespace Util

Classes

FloatImage

Multi-channel float raster image. Can compute mirrored borders.

RadianceHDRFormat

Radiance HDR (PIC) file-format. Can read/write RLE-encoded HDR files.

RadianceHDRFormat.HDRInstance

Class FloatImage

Namespace: Util

Assembly: rt004.dll

Multi-channel float raster image. Can compute mirrored borders.

```
public class FloatImage
```

Inheritance

object ← FloatImage

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

FloatImage(int, int, int, int)

Create a HDR image of required dimensions.

```
public FloatImage(int wid, int hei, int ch = 1, int bor = 0)
```

Parameters

`wid` int

Image width in pixels.

`hei` int

Image height in pixels.

`ch` int

Number of channels.

`bor` int

Border width in pixels.

FloatImage(FloatImage?)

```
public FloatImage(FloatImage? from)
```

Parameters

`from` FloatImage

Fields

BLUE_WEIGHT

```
public const float BLUE_WEIGHT = 0.1144
```

Field Value

float

GREEN_WEIGHT

```
public const float GREEN_WEIGHT = 0.5866
```

Field Value

float

RED_WEIGHT

```
public const float RED_WEIGHT = 0.2989
```

Field Value

float

border

Internal image border allocated around the whole image.

```
protected int border
```

Field Value

int

cdf

Cumulative distribution function in discrete form. Support data structure for efficient sampling. If allocated, it should have 'image resolution' + 1 values.

```
protected double[]? cdf
```

Field Value

double[]

channels

```
protected int channels
```

Field Value

int

data

Raw image data with optional border.

```
protected float[]? data
```

Field Value

float[]

height

```
protected int height
```

Field Value

int

origin

Origin of the image itself (skipping over the optional borders).

```
protected int origin
```

Field Value

int

stride

Array stride in floats (indices).

```
protected int stride
```

Field Value

int

widChannels

```
protected int widChannels
```

Field Value

int

width

```
protected int width
```

Field Value

int

Properties

Channels

Number of image channels (number of float numbers per pixel).

```
public int Channels { get; }
```

Property Value

int

Data

Raw data array (use Scan0 and Stride).

```
public float[]? Data { get; }
```

Property Value

float[]

Height

Image height in pixels.

```
public int Height { get; }
```

Property Value

int

Scan0

Offset of the upper-left image corner from the array origin (in array indices).

```
public int Scan0 { get; }
```

Property Value

int

Stride

Image stride in array indices.

```
public int Stride { get; }
```

Property Value

int

Width

Image width in pixels.

```
public int Width { get; }
```

Property Value

int

Methods

AssertMinBorder(int)

Asserts minimal border value. Computes actual border pixels.

```
public void AssertMinBorder(int minBorder)
```

Parameters

`minBorder` int

Minimal border size needed.

Blur(bool)

Image blur - Gaussian or uniform.

```
public void Blur(bool gauss = false)
```

Parameters

`gauss` bool

Use Gaussian filter? (3x3 window)

ComputeBorder(int)

Computes the image border.

```
protected void ComputeBorder(int type = 0)
```

Parameters

`type` int

Border type, ignored (only mirror border is implemented).

Contrast(out double, out double)

Computes image range (min and max pixel values).

```
public void Contrast(out double minY, out double maxY)
```

Parameters

`minY` double

Output (nonzero) min value.

`maxY` double

Output max value.

Exposure(byte[]?, double, double)

Computes simple exposure from HDR to LDR format. Optional gamma pre-compensation

```
public byte[]? Exposure(byte[]? result, double exp, double gamma = 0)
```

Parameters

`result` byte[]

Optional pre-allocated Bitmap.

`exp` double

Exposure coefficient.

`gamma` double

Optional target gammma (if zero, no pre-compensation is done).

Returns

byte[]

Output LDR Bitbap.

FromFile(string)

Read HDR image from the given disk file.

```
public static FloatImage? FromFile(string fileName)
```

Parameters

`fileName` string

Returns

FloatImage

Result or 'null' in case of failure.

GetGray(int, int)

Reads one grayscale pixel.

```
public float GetGray(int x, int y)
```

Parameters

`x` int

X-coordinate.

`y` int

Y-coordinate.

Returns

float

Gray value or -1.0f if out of range.

GetPixel(int, int, float[])

Reads one HDR color pixel.

```
public bool GetPixel(int x, int y, float[] pix)
```

Parameters

`x` int

X-coordinate.

`y` int

Y-coordinate.

`pix` float[]

Pre-allocated output array.

Returns

bool

True if Ok.

GetSample(out double, out double, double, Random?)

CDF-based sampling.

```
public void GetSample(out double x, out double y, double random, Random? rnd = null)
```

Parameters

`x` double

Output horizontal coordinate from the [0.0,width) range.

`y` double

Output vertical coordinate from the [0.0,height) range.

`random` double

[0,1] uniform random value.

`rnd` Random

Optional random generator instance. If provided, internal randomization is possible.

GrayImage(bool, double)

```
public FloatImage GrayImage(bool inv = false, double gamma = 0)
```

Parameters

`inv` bool

`gamma` double

Returns

FloatImage

MAD(FloatImage)

Mean Absolute Difference of two images.

```
public float MAD(FloatImage b)
```

Parameters

b FloatImage

The other image.

Returns

float

Sum of absolute pixel differences divided by number of pixels.

MAD(FloatImage, int)

Mean Absolute Difference of two images, one channel only.

```
public float MAD(FloatImage b, int ch)
```

Parameters

b FloatImage

The other image.

ch int

Channel number.

Returns

float

Sum of absolute pixel differences divided by number of pixels.

PrepareCdf(int)

Prepare support cdf array.

```
public void PrepareCdf(int ch = 0)
```

Parameters

ch int

Optional channel index.

PutPixel(int, int, float)

Sets the gray-value pixel.

```
public void PutPixel(int x, int y, float val)
```

Parameters

x int

X-coordinate.

y int

Y-coordinate.

val float

New pixel value.

PutPixel(int, int, float[]?)

Sets the required pixel to a new value.

```
public void PutPixel(int x, int y, float[]? pix)
```

Parameters

x int

X-coordinate.

y int

Y-coordinate.

pix float[]

New pixel value.

Resize(int)

Image resize by an integral subsample factor.

```
public void Resize(int factor)
```

Parameters

`factor` int

Subsample factor.

SaveHDR(string)

```
public void SaveHDR(string fileName)
```

Parameters

`fileName` string

SavePFM(string)

```
public void SavePFM(string fileName)
```

Parameters

`fileName` string

SetBorder(int)

Sets the required border size (exactly). Does not compute the border pixels yet!

```
public void SetBorder(int newBorder)
```

Parameters

`newBorder` int

Required border size in pixels. Zero value means 'no border'.

init(int, int, int, int, float[]?)

```
protected void init(int wid, int hei, int ch = 1, int bor = 0, float[]? d = null)
```

Parameters

`wid` int

`hei` int

`ch` int

`bor` int

`d` float[]

setAccelerators()

```
protected void setAccelerators()
```

Class RadianceHDRFormat

Namespace: Util

Assembly: rt004.dll

Radiance HDR (PIC) file-format. Can read/write RLE-encoded HDR files.

```
public class RadianceHDRFormat
```

Inheritance

object ← RadianceHDRFormat

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

RGBToRGBe(byte[], int, double, double, double)

Conversion from HDR floating-point RGB format to Radiance's RGBe 32-bit format.

```
public static void RGBToRGBe(byte[] rgbe, int startRgbe, double R, double G, double B)
```

Parameters

`rgbe` byte[]

`startRgbe` int

`R` double

`G` double

`B` double

RGBeToRGB(byte[], int, float[], int)

Conversion from Radiance's RGBe 32-bit format into HDR floating-point RGB format.

```
public static void RGBeToRGB(byte[] rgbe, int startRgbe, float[] rgb, int startRgb)
```

Parameters

`rgbe` byte[]

`startRgbe` int

`rgb` float[]

`startRgb` int

Class RadianceHDRFormat.HDRInstance

Namespace: Util

Assembly: rt004.dll

```
public class RadianceHDRFormat.HDRInstance
```

Inheritance

object ← RadianceHDRFormat.HDRInstance

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Fields

scanline

```
protected byte[]? scanline
```

Field Value

byte[]

Methods

assertScanline(int)

```
protected void assertScanline(int width)
```

Parameters

`width` int

commonLoad(Stream)

```
public FloatImage commonLoad(Stream stream)
```

Parameters

`stream` Stream

Returns

FloatImage

commonSave(Stream, FloatImage)

```
public void commonSave(Stream stream, FloatImage im)
```

Parameters

`stream` Stream

`im` FloatImage

oldReadScanline(int, int, Stream, int)

```
protected void oldReadScanline(int start, int len, Stream iss, int unget)
```

Parameters

`start` int

`len` int

`iss` Stream

`unget` int

readScanline(int, Stream)

```
protected void readScanline(int len, Stream iss)
```

Parameters

`len` int

`iss` Stream

readToken(Stream)

```
public static string readToken(Stream iss)
```

Parameters

`iss` Stream

Returns

string

writeScanline(int, Stream)

```
protected void writeScanline(int len, Stream oss)
```

Parameters

`len` int

`oss` Stream

Namespace rt004

Classes

Extensions

Scene

Represents a 3D scene containing cameras, solid objects, and light sources. Provides functionality for managing scene hierarchy, rendering, and ray-object intersection testing.

Class Extensions

Namespace: rt004

Assembly: rt004.dll

```
public static class Extensions
```

Inheritance

object ← Extensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

IsFloatEqual(double, double)

Compares two double values with +- RendererSettings.epsilon precision.

```
public static bool IsFloatEqual(this double value, double other)
```

Parameters

`value` double

Value to compare with

`other` double

Value to compare

Returns

bool

Returns true is other is in value +- epsilon range, else returns false

IsFloatEqual(float, float)

Compares two float values with +- RendererSettings.epsilon precision.

```
public static bool IsFloatEqual(this float value, float other)
```

Parameters

`value` float

Value to compare with

`other` float

Value to compare

Returns

bool

Returns true is other is in value +- epsilon range, else returns false

IsVectorEquals(Vector3d, Vector3d)

Compares two 3D vectors with +- RendererSettings.epsilon precision.

By calling Double.IsFloatEqual on each of its components comparing with apropriate value from other vector.

```
public static bool IsVectorEquals(this Vector3d vector, Vector3d other)
```

Parameters

`vector` Vector3d

`other` Vector3d

Vector to compare

Returns

bool

Returns true is other is in value +- epsilon range, else returns false

RotationMatrix(Vector3d)

Creates rotation matrix from euler angles in X,Y,Z order.

```
public static Matrix4d RotationMatrix(Vector3d eulerAngles)
```

Parameters

`eulerAngles` Vector3d

Vector defining euler angles in X,Y,Z order

Returns

Matrix4d

Returns rotation matrix

Class Scene

Namespace: rt004

Assembly: rt004.dll

Represents a 3D scene containing cameras, solid objects, and light sources. Provides functionality for managing scene hierarchy, rendering, and ray-object intersection testing.

```
public class Scene
```

Inheritance

object ← Scene

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Scene(out InnerSceneObject)

Initializes a new instance of the Scene class with default ambient light settings.

```
public Scene(out InnerSceneObject rootHierarchyNode)
```

Parameters

`rootHierarchyNode` InnerSceneObject

When this method returns, contains the root hierarchy node of the scene.

Scene(out InnerSceneObject, Color4, float)

Initializes a new instance of the Scene class with specified ambient light settings.

```
public Scene(out InnerSceneObject rootHierarchyNode, Color4 ambientLightColor,
float ambientLightIntensity)
```

Parameters

`rootHierarchyNode` InnerSceneObject

When this method returns, contains the root hierarchy node of the scene.

`ambientLightColor` Color4

The ambient light color for the scene.

`ambientLightIntensity` float

The ambient light intensity for the scene.

Fields

rootSceneObject

The root object of the scene hierarchy tree.

```
public readonly InnerSceneObject rootSceneObject
```

Field Value

InnerSceneObject

Properties

AmbientLightColor

Gets or sets the ambient light color that illuminates all objects in the scene uniformly.

```
public Color4 AmbientLightColor { get; set; }
```

Property Value

Color4

AmbientLightIntensity

Gets or sets the intensity factor for ambient lighting in the scene.

```
public float AmbientLightIntensity { get; set; }
```

Property Value

float

MainCamera

Gets or sets the main camera used for rendering by default.

```
public Camera? MainCamera { get; set; }
```

Property Value

Camera

Exceptions

KeyNotFoundException

Thrown when attempting to set a camera that is not in the scene hierarchy.

RootObject

Gets the root object of the scene hierarchy.

```
public InnerSceneObject RootObject { get; }
```

Property Value

InnerSceneObject

Methods

AddCamera(InnerSceneObject, Camera, bool)

Adds a camera to the scene and optionally sets it as the main camera.

```
public void AddCamera(InnerSceneObject parentSceneObject, Camera camera, bool setAsMain
= false)
```

Parameters

`parentSceneObject` InnerSceneObject

The parent object of this camera in the scene hierarchy.

`camera` Camera

The camera to add to the scene.

`setAsMain` bool

If true, sets this camera as the main camera. If false, sets as main only if no main camera exists.

AddLight(InnerSceneObject, LightSource)

Adds a light source to the scene.

```
public void AddLight(InnerSceneObject parentObject, LightSource light)
```

Parameters

`parentObject` InnerSceneObject

The parent object of this light source in the scene hierarchy.

`light` LightSource

The light source to add to the scene.

AddSceneObject(InnerSceneObject, SceneObject)

Adds a scene object to the scene, automatically determining its type and adding it to the appropriate collection.

```
public void AddSceneObject(InnerSceneObject parentSceneObject, SceneObject sceneObject)
```

Parameters

`parentSceneObject` InnerSceneObject

The parent object to add the scene object to as a child.

`sceneObject` SceneObject

The scene object to add to the scene.

Exceptions

NotImplementedException

Thrown when the scene object is not a recognized type (Camera, Solid, LightSource, or InnerSceneObject).

AddSolid(InnerSceneObject, Solid)

Adds a solid object to the scene.

```
public void AddSolid(InnerSceneObject parentSceneObject, Solid solid)
```

Parameters

`parentSceneObject` InnerSceneObject

The parent object of this solid in the scene hierarchy.

`solid` Solid

The solid object to add to the scene.

CastRay(Ray, out double)

Contains methods for testing ray intersections with objects in the scene. These methods are used for rendering, shadow calculation, and collision detection.

```
public bool CastRay(Ray ray, out double distance)
```

Parameters

`ray` Ray

The ray to cast in the scene.

`distance` double

When this method returns, contains the distance from the ray origin to the closest intersection point.

Returns

bool

True if any object is intersected by the ray; otherwise, false.

CastRay(Ray, out double, double, double)

Casts a ray in the scene and checks for intersections with solid objects within specified distance limits.

```
public bool CastRay(Ray ray, out double distance, double maxDistance, double minDistance
= 0)
```

Parameters

`ray` Ray

The ray to cast in the scene.

`distance` double

When this method returns, contains the distance from the ray origin to the closest intersection point.

`maxDistance` double

The maximum distance to consider for intersections.

`minDistance` double

The minimum distance to consider for intersections.

Returns

bool

True if any object is intersected by the ray within the specified distance range; otherwise, false.

CastRay(Ray, out double, out Solid)

Casts a ray in the scene and checks for intersections with solid objects, returning the intersected object.

```
public bool CastRay(Ray ray, out double distance, out Solid intersectedBody)
```

Parameters

`ray` Ray

The ray to cast in the scene.

`distance` double

When this method returns, contains the distance from the ray origin to the closest intersection point.

`intersectedBody` Solid

When this method returns, contains the closest intersected solid object.

Returns

bool

True if any object is intersected by the ray; otherwise, false.

CastRay(Ray, out IntersectionProperties)

Casts a ray in the scene and returns detailed intersection properties for the closest intersection.

```
public bool CastRay(Ray ray, out IntersectionProperties properties)
```

Parameters

`ray` Ray

The ray to cast in the scene.

`properties` IntersectionProperties

When this method returns, contains the detailed properties of the closest intersection.

Returns

bool

True if an intersection is found; otherwise, false.

CastRay(Ray, out IntersectionProperties, double, double)

Casts a ray in the scene and returns detailed intersection properties for the closest intersection within distance limits.

```
public bool CastRay(Ray ray, out IntersectionProperties properties, double maxDistance,
double minDistance = 0)
```

Parameters

`ray` Ray

The ray to cast in the scene.

`properties` IntersectionProperties

When this method returns, contains the detailed properties of the closest intersection.

`maxDistance` double

The maximum distance from the ray origin to consider for intersections.

`minDistance` double

The minimum distance from the ray origin to consider for intersections.

Returns

bool

True if an intersection is found within the specified distance range; otherwise, false.

CastRay(Ray, out Point3D)

Casts a ray in the scene and returns the closest intersection point.

```
public bool CastRay(Ray ray, out Point3D closestIntersection)
```

Parameters

`ray` Ray

The ray to cast in the scene.

`closestIntersection` Point3D

When this method returns, contains the closest intersection point. Valid only when the method returns true.

Returns

bool

True if an intersection is found; otherwise, false.

CastRay(Ray, out Point3D, out Solid)

Casts a ray in the scene and returns the closest intersection point and the intersected object.

```
public bool CastRay(Ray ray, out Point3D closestIntersection, out Solid intersectedBody)
```

Parameters

`ray` Ray

The ray to cast in the scene.

`closestIntersection` Point3D

When this method returns, contains the closest intersection point. Valid only when the method returns true.

`intersectedBody` Solid

When this method returns, contains the closest intersected solid object. Valid only when the method returns true.

Returns

bool

True if an intersection is found; otherwise, false.

GetCameras()

Gets a shallow copy of an array containing all cameras in the scene.

```
public Camera[] GetCameras()
```

Returns

Camera[]

An array of all cameras in the scene.

GetLightSources()

Gets a shallow copy of an array containing all light sources in the scene.

```
public LightSource[] GetLightSources()
```

Returns

LightSource[]

An array of all light sources in the scene.

GetSolids()

Gets a shallow copy of an array containing all solid objects in the scene.

```
public Solid[] GetSolids()
```

Returns

Solid[]

An array of all solid objects in the scene.

RemoveCamera(Camera)

Removes a camera from the scene and updates the main camera if necessary.

```
public void RemoveCamera(Camera camera)
```

Parameters

`camera` Camera

The camera to remove from the scene.

RemoveLight(LightSource)

Removes a light source from the scene.

```
public void RemoveLight(LightSource light)
```

Parameters

`light` LightSource

The light source to remove from the scene.

RemoveSceneObject(SceneObject)

Removes a scene object from the scene, automatically determining its type and removing it from the appropriate collection.

```
public void RemoveSceneObject(SceneObject sceneObject)
```

Parameters

`sceneObject` SceneObject

The scene object to remove from the scene.

Exceptions

NotImplementedException

Thrown when the scene object is not a recognized type.

RemoveSolid(Solid)

Removes a solid object from the scene.

```
public void RemoveSolid(Solid solid)
```

Parameters

`solid` Solid

The solid object to remove from the scene.

RenderScene()

Renders an image from the main camera's perspective.

```
public FloatImage RenderScene()
```

Returns

FloatImage

A rendered image from the main camera.

Exceptions

ArgumentNullException

Thrown when no main camera exists and no cameras are available in the scene.

RenderSceneWithAllCameras()

Renders images from all cameras in the scene.

```
public FloatImage[] RenderSceneWithAllCameras()
```

Returns

FloatImage[]

An array of rendered images, one from each camera in the scene.

Namespace rt004.Loading

Classes

SceneLoader

Provides functionality for loading and creating scene instances from configuration data. Handles the creation of scene objects and their hierarchy from loaded data.

Class SceneLoader

Namespace: rt004.Loading

Assembly: rt004.dll

Provides functionality for loading and creating scene instances from configuration data. Handles the creation of scene objects and their hierarchy from loaded data.

```
public class SceneLoader
```

Inheritance

object ← SceneLoader

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Fields

ambientLightColor

Gets or sets the ambient light color for the scene.

```
public Color4 ambientLightColor
```

Field Value

Color4

ambientLightIntensity

Gets or sets the ambient light intensity for the scene.

```
public float ambientLightIntensity
```

Field Value

float

sceneObjectLoaders

Gets or sets the list of scene object loaders that define the objects to be created in the scene.

```
public List<SceneObjectLoader> sceneObjectLoaders
```

Field Value

List<SceneObjectLoader>

Methods

CreateInstance()

Creates a new Scene instance from the loaded configuration data.

```
public Scene CreateInstance()
```

Returns

Scene

A fully configured scene with all objects and hierarchy established.

ExtractChildren(InnerSceneObject, in List<SceneObject>)

Recursively extracts all children from a scene object hierarchy and adds them to a flat list.

```
public static void ExtractChildren(InnerSceneObject parentObject, in
List<SceneObject> allChildren)
```

Parameters

`parentObject` InnerSceneObject

The parent scene object to extract children from.

`allChildren` List<SceneObject>

The list to add all extracted children to.

Namespace rt004.MaterialLoading

Classes

PhongMaterialLoader

A loader class for creating instances of PhongMaterial.

Class PhongMaterialLoader

Namespace: rt004.MaterialLoading

Assembly: rt004.dll

A loader class for creating instances of PhongMaterial.

```
public class PhongMaterialLoader : MaterialLoader
```

Inheritance

object ← MaterialLoader ← PhongMaterialLoader

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PhongMaterialLoader()

Initializes a new instance of the PhongMaterialLoader class with default textures.

```
public PhongMaterialLoader()
```

PhongMaterialLoader(Color4, float, float, float, float, float)

Initializes a new instance of the PhongMaterialLoader class with specified parameters.

```
public PhongMaterialLoader(Color4 baseColor, float specular, float diffuse, float shininess,
float transparency, float indexOfRefraction)
```

Parameters

`baseColor` Color4

The base color texture.

`specular` float

The specular intensity.

`diffuse` float

The diffuse intensity.

`shininess` float

The shininess factor.

`transparency` float

The transparency factor.

`indexOfRefraction` float

The index of refraction.

Fields

baseColor

The loader for the base color texture.

```
public TextureLoader baseColor
```

Field Value

TextureLoader

diffuseTexture

The loader for the diffuse texture.

```
public MonochromeTextureLoader diffuseTexture
```

Field Value

MonochromeTextureLoader

indexOfRefractionTexture

The loader for the index of refraction texture.

```
public MonochromeTextureLoader indexOfRefractionTexture
```

Field Value

MonochromeTextureLoader

shininessTexture

The loader for the shininess texture.

```
public MonochromeTextureLoader shininessTexture
```

Field Value

MonochromeTextureLoader

specularTexture

The loader for the specular texture.

```
public MonochromeTextureLoader specularTexture
```

Field Value

MonochromeTextureLoader

transparencyTexture

The loader for the transparency texture.

```
public MonochromeTextureLoader transparencyTexture
```

Field Value

MonochromeTextureLoader

Methods

CreateInstance()

Creates an instance of PhongMaterial using the loaded textures and parameters.

```
public override Material CreateInstance()
```

Returns

Material

A new instance of PhongMaterial.

Namespace rt004.Materials

Classes

Material

Represents an abstract base class for different types of materials used in rendering.

MonochromeImageTexture

Represents grayscale image texture

MonochromeTexture

Represents texture with only one color chennel per pixel (gray-scale texture)

MonochromeUniformTexture

Represents a monochrome uniform texture with a single grayscale value across its surface.

Texture

Represents a base class for textures, providing common properties and methods for textures.

UniformTexture

Represents a uniform texture with a single, consistent color across its entire surface.

Class Material

Namespace: rt004.Materials

Assembly: rt004.dll

Represents an abstract base class for different types of materials used in rendering.

```
public abstract class Material
```

Inheritance

object ← Material

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

GetMaterialFor(LightModel)

Selects the appropriate material for a specified lighting model.

```
public static Material GetMaterialFor(LightModel lightModel)
```

Parameters

`lightModel` LightModel

The lighting model for which to select the material.

Returns

Material

A Material instance that is compatible with the specified lighting model.

Exceptions

NotImplementedException

Thrown if there is no defined material for the specified lighting model.

IsCorrectMaterialFor(LightModel)

Determines if the material is compatible with a specified lighting model.

```
public abstract bool IsCorrectMaterialFor(LightModel lightModel)
```

Parameters

`lightModel` LightModel

The lighting model to check for compatibility.

Returns

bool

`true` if the material is compatible with the specified lighting model; otherwise, `false`.

Class MonochromeImageTexture

Namespace: rt004.Materials

Assembly: rt004.dll

Represents grayscale image texture

```
public class MonochromeImageTexture : MonochromeTexture
```

Inheritance

object ← Texture ← MonochromeTexture ← MonochromeImageTexture

Inherited Members

Texture.width , Texture.height , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

MonochromeImageTexture(uint, uint)

```
public MonochromeImageTexture(uint u, uint v)
```

Parameters

u uint

v uint

Methods

GetColorAt(float, float)

returns value represented in color

```
public override Color4 GetColorAt(float u, float v)
```

Parameters

u [float](#)

horizontal coord

v [float](#)

vertical coord

Returns

Color4

returns factor in Color4 represented as (value, value, value, 1)

GetFactorAt(float, float)

Gets bilinaerly interpolated value from near image pixel values.

```
public override float GetFactorAt(float u, float v)
```

Parameters

u [float](#)

u coordinate

v [float](#)

v coordinate

Returns

[float](#)

bilinearly interpolated value from image

Class MonochromeTexture

Namespace: rt004.Materials

Assembly: rt004.dll

Represents texture with only one color chennel per pixel (gray-scale texture)

```
public abstract class MonochromeTexture : Texture
```

Inheritance

object ← Texture ← MonochromeTexture

Derived

MonochromeImageTexture, MonochromeUniformTexture

Inherited Members

Texture.width , Texture.height , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

MonochromeTexture(uint, uint)

```
public MonochromeTexture(uint u, uint v)
```

Parameters

u uint

v uint

Methods

GetColorAt(float, float)

Gets the color at a specified position in the texture.

```
public override abstract Color4 GetColorAt(float u, float v)
```

Parameters

u [float](#)

The horizontal coordinate, typically in normalized UV space.

v [float](#)

The vertical coordinate, typically in normalized UV space.

Returns

Color4

The color at the specified (u, v) coordinates.

GetFactorAt(float, float)

Gets the value at position

```
public abstract float GetFactorAt(float u, float v)
```

Parameters

u [float](#)

horizontal coord

v [float](#)

vertical coord

Returns

[float](#)

returns float value at the position

Class MonochromeUniformTexture

Namespace: rt004.Materials

Assembly: rt004.dll

Represents a monochrome uniform texture with a single grayscale value across its surface.

```
public class MonochromeUniformTexture : MonochromeTexture
```

Inheritance

object ← Texture ← MonochromeTexture ← MonochromeUniformTexture

Inherited Members

Texture.width , Texture.height , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

MonochromeUniformTexture(float)

Initializes a new instance of the MonochromeUniformTexture class with a specified grayscale value.

```
public MonochromeUniformTexture(float value)
```

Parameters

`value` float

The grayscale intensity value for the texture.

Fields

value

The grayscale value of the texture, consistent across the entire surface.

```
public float value
```

Field Value

float

Methods

GetColorAt(float, float)

Gets the color representation of the grayscale value at any UV coordinates. Since this is a uniform texture, the color is the same at all coordinates.

```
public override Color4 GetColorAt(float u, float v)
```

Parameters

u float

The horizontal coordinate (ignored in this class).

v float

The vertical coordinate (ignored in this class).

Returns

Color4

The color representation of the grayscale value, encoded as (value, value, value, 1).

GetFactorAt(float, float)

Gets the grayscale intensity factor of the texture at any UV coordinates.

```
public override float GetFactorAt(float u, float v)
```

Parameters

u float

The horizontal coordinate (ignored in this class).

v float

The vertical coordinate (ignored in this class).

Returns

float

The grayscale intensity factor.

Class Texture

Namespace: rt004.Materials

Assembly: rt004.dll

Represents a base class for textures, providing common properties and methods for textures.

```
public abstract class Texture
```

Inheritance

object ← Texture

Derived

MonochromeTexture, UniformTexture

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Texture(uint, uint)

Initializes a new instance of the Texture class with specified width and height.

```
public Texture(uint width, uint height)
```

Parameters

`width` uint

The width of the texture in pixels.

`height` uint

The height of the texture in pixels.

Fields

height

The height of the texture in pixels.

```
public readonly uint height
```

Field Value

uint

width

The width of the texture in pixels.

```
public readonly uint width
```

Field Value

uint

Methods

GetColorAt(float, float)

Gets the color at a specified position in the texture.

```
public abstract Color4 GetColorAt(float u, float v)
```

Parameters

u float

The horizontal coordinate, typically in normalized UV space.

v float

The vertical coordinate, typically in normalized UV space.

Returns

Color4

The color at the specified (u, v) coordinates.

Class UniformTexture

Namespace: rt004.Materials

Assembly: rt004.dll

Represents a uniform texture with a single, consistent color across its entire surface.

```
public class UniformTexture : Texture
```

Inheritance

object ← Texture ← UniformTexture

Inherited Members

Texture.width , Texture.height , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

UniformTexture(Color4)

Initializes a new instance of the UniformTexture class with a specified color.

```
public UniformTexture(Color4 color)
```

Parameters

`color` Color4

The uniform color to be applied across the texture.

Methods

GetColorAt(float, float)

Gets the color of the texture at any specified UV coordinates. Since this is a uniform texture, the color is the same across the entire texture.

```
public override Color4 GetColorAt(float u, float v)
```

Parameters

u [float](#)

The horizontal coordinate (ignored in this class).

v [float](#)

The vertical coordinate (ignored in this class).

Returns

Color4

The uniform color of the texture.

Namespace rt004.Materials.Loading

Classes

MaterialLoader

Abstract base class for loading and creating instances of Material. Supports XML serialization for different types of material loaders.

MonochromeImageTextureLoader

Loader of image in the texture. Currently not fully implemented!

MonochromeTextureLoader

Abstract support class used for loading a monochrome texture.

MonochromeUniformTextureLoader

A loader class for creating instances of MonochromeUniformTexture with a specified grayscale value.

TextureLoader

Represents a base class for loading textures, with support for XML serialization.

UniformTextureLoader

Represents a loader for creating UniformTexture instances with a specified color.

Class MaterialLoader

Namespace: rt004.Materials.Loading

Assembly: rt004.dll

Abstract base class for loading and creating instances of Material. Supports XML serialization for different types of material loaders.

```
public abstract class MaterialLoader
```

Inheritance

object ← MaterialLoader

Derived

PhongMaterialLoader

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

CreateInstance()

Creates a new instance of a material based on the loader's configuration.

```
public abstract Material CreateInstance()
```

Returns

Material

A new instance of a Material.

Class MonochromeImageTextureLoader

Namespace: rt004.Materials.Loading

Assembly: rt004.dll

Loader of image in the texture. Currently not fully implemented!

```
public class MonochromeImageTextureLoader : MonochromeTextureLoader
```

Inheritance

object ← TextureLoader ← MonochromeTextureLoader ← MonochromeImageTextureLoader

Inherited Members

TextureLoader.width , TextureLoader.height , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

MonochromeImageTextureLoader()

```
public MonochromeImageTextureLoader()
```

Methods

GetInstance()

Creates a new instance of MonochromeImageTexture with the specified color.

```
public override Texture GetInstance()
```

Returns

Texture

A new MonochromeImageTexture instance initialized with the specified color.

Class MonochromeTextureLoader

Namespace: rt004.Materials.Loading

Assembly: rt004.dll

Abstract support class used for loading a monochrome texture.

```
public abstract class MonochromeTextureLoader : TextureLoader
```

Inheritance

object ← TextureLoader ← MonochromeTextureLoader

Derived

MonochromeImageTextureLoader, MonochromeUniformTextureLoader

Inherited Members

TextureLoader.width , TextureLoader.height , TextureLoader.GetInstance() , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

MonochromeTextureLoader()

```
public MonochromeTextureLoader()
```

Class MonochromeUniformTextureLoader

Namespace: rt004.Materials.Loading

Assembly: rt004.dll

A loader class for creating instances of MonochromeUniformTexture with a specified grayscale value.

```
public class MonochromeUniformTextureLoader : MonochromeTextureLoader
```

Inheritance

object ← TextureLoader ← MonochromeTextureLoader ← MonochromeUniformTextureLoader

Inherited Members

TextureLoader.width , TextureLoader.height , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

MonochromeUniformTextureLoader()

Initializes a new instance of the MonochromeUniformTextureLoader class with a default value of 0.

```
public MonochromeUniformTextureLoader()
```

MonochromeUniformTextureLoader(float)

Initializes a new instance of the MonochromeUniformTextureLoader class with a specified grayscale value.

```
public MonochromeUniformTextureLoader(float value)
```

Parameters

`value` float

The grayscale value for the texture.

Fields

value

The grayscale value to be used for initializing the MonochromeUniformTexture.

```
public float value
```

Field Value

float

Methods

GetInstance()

Creates a new instance of MonochromeUniformTexture with the specified grayscale value.

```
public override Texture GetInstance()
```

Returns

Texture

A new MonochromeUniformTexture instance with the configured grayscale value.

Class TextureLoader

Namespace: rt004.Materials.Loading

Assembly: rt004.dll

Represents a base class for loading textures, with support for XML serialization.

```
public abstract class TextureLoader
```

Inheritance

object ← TextureLoader

Derived

MonochromeTextureLoader, UniformTextureLoader

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Fields

height

The height of the texture to load in pixels.

```
public uint height
```

Field Value

uint

width

The width of the texture to load in pixels.

```
public uint width
```

Field Value

uint

Methods

GetInstance()

Creates an instance of a Texture based on the current settings.

```
public abstract Texture GetInstance()
```

Returns

Texture

A new instance of Texture configured with specified width and height.

Class UniformTextureLoader

Namespace: rt004.Materials.Loading

Assembly: rt004.dll

Represents a loader for creating UniformTexture instances with a specified color.

```
public class UniformTextureLoader : TextureLoader
```

Inheritance

object ← TextureLoader ← UniformTextureLoader

Inherited Members

TextureLoader.width , TextureLoader.height , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

UniformTextureLoader()

Initializes a new instance of the UniformTextureLoader class with a default color (white).

```
public UniformTextureLoader()
```

UniformTextureLoader(Color4)

Initializes a new instance of the UniformTextureLoader class with a specified color.

```
public UniformTextureLoader(Color4 color)
```

Parameters

`color` Color4

The color to apply to the UniformTexture.

Fields

color

The color used to initialize the UniformTexture.

```
public Color4 color
```

Field Value

Color4

Methods

GetInstance()

Creates a new instance of UniformTexture with the specified color.

```
public override Texture GetInstance()
```

Returns

Texture

A new UniformTexture instance initialized with the specified color.

Namespace rt004.SceneObjects

Classes

Camera
: Represents an abstract base class for cameras within the scene, defining basic properties and methods.

InnerSceneObject

LightSource
: Represents an abstract base class for light sources in a scene, defining common properties and methods.

PerspectiveCamera
: Represents a perspective camera that renders a 3D scene with a specified field of view (FoV).

Plane
: Represents an infinite plane solid in 3D space for ray tracing and intersection tests.

PointLight

SceneObject

Solid
: Abstract base class for all solid objects that can be rendered in a scene. Provides common functionality for ray-solid intersection tests and material properties.

Sphere
: Represents a mathematically perfect sphere solid object in a 3D scene. Provides ray-sphere intersection calculations and surface normal computation.

Structs

IntersectionProperties

Class Camera

Namespace: rt004.SceneObjects

Assembly: rt004.dll

Represents an abstract base class for cameras within the scene, defining basic properties and methods.

```
public abstract class Camera : SceneObject
```

Inheritance

object ← SceneObject ← Camera

Derived

PerspectiveCamera

Inherited Members

SceneObject.parentScene , SceneObject.dirtyGlobalPosition , SceneObject.Position , SceneObject.GlobalPosition , SceneObject.Rotation , SceneObject.ParentScene , SceneObject.ParentObject , SceneObject.GetLocalHeding() , SceneObject.GetGlobalHeading() , SceneObject.GetGlobalPosition() , SceneObject.GetLocalToWorldTransformMatrix() , SceneObject.GetWorldToLocalTransformMatrix() , SceneObject.GetLocalTransformMatrix() , SceneObject.GetLocalInverseTransformMatrix() , SceneObject.ToWorldSpace(Vector3D) , SceneObject.ToWorldSpace(Point3D) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Camera(Scene, Point3D, Vector3, uint, uint)

Initializes a new instance of the Camera class with specified parameters.

```
public Camera(Scene parentScene, Point3D position, Vector3 rotation, uint width,
uint height)
```

Parameters

`parentScene` Scene

The parent scene to which this camera belongs.

`position` Point3D

The position of the camera within the scene.

`rotation` Vector3

The rotation of the camera within the scene.

`width` uint

The width of the camera resolution in pixels.

`height` uint

The height of the camera resolution in pixels.

Fields

height

The height of the camera in pixels.

```
protected uint height
```

Field Value

uint

maxRenderingDistanceSquered

The maximum rendering distance squared, defining how far the camera can render objects.

```
protected float maxRenderingDistanceSquered
```

Field Value

float

width

The width of the camera in pixels.

```
protected uint width
```

Field Value

uint

Properties

Height

Gets or sets the height of the camera's resolution. Setting this property updates the resolution.

```
public uint Height { get; set; }
```

Property Value

uint

MaxRenderingDistance

Gets or sets the maximum rendering distance of the camera.

```
public float MaxRenderingDistance { get; set; }
```

Property Value

float

Width

Gets or sets the width of the camera's resolution. Setting this property updates the resolution.

```
public uint Width { get; set; }
```

Property Value

uint

Methods

GetResolution()

Gets the current resolution of the camera.

```
public (uint, uint) GetResolution()
```

Returns

(uint, uint)

A tuple of `uint` representing the width and height, respectively.

RenderImage()

Renders an image of the scene from the camera's perspective.

```
public abstract FloatImage RenderImage()
```

Returns

FloatImage

A FloatImage representing the rendered scene image.

SetResolution(uint, uint)

Sets the resolution of the camera.

```
public abstract void SetResolution(uint width, uint height)
```

Parameters

`width` uint

The width of the resolution in pixels.

`height` uint

The height of the resolution in pixels.

Class InnerSceneObject

Namespace: rt004.SceneObjects

Assembly: rt004.dll

```
public class InnerSceneObject : SceneObject
```

Inheritance

object ← SceneObject ← InnerSceneObject

Inherited Members

SceneObject.parentScene , SceneObject.dirtyGlobalPosition , SceneObject.Position , SceneObject.GlobalPosition , SceneObject.Rotation , SceneObject.ParentObject , SceneObject.GetLocalHeding() , SceneObject.GetGlobalHeading() , SceneObject.GetGlobalPosition() , SceneObject.GetLocalToWorldTransformMatrix() , SceneObject.GetWorldToLocalTransformMatrix() , SceneObject.GetLocalTransformMatrix() , SceneObject.GetLocalInverseTransformMatrix() , SceneObject.ToWorldSpace(Vector3D) , SceneObject.ToWorldSpace(Point3D) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

InnerSceneObject(Scene, Point3D, Vector3)

```
public InnerSceneObject(Scene parentScene, Point3D position, Vector3 rotation)
```

Parameters

`parentScene` Scene

`position` Point3D

`rotation` Vector3

InnerSceneObject(Scene, Point3D, Vector3, HashSet<SceneObject>)

```
public InnerSceneObject(Scene parentScene, Point3D position, Vector3 rotation,
HashSet<SceneObject> children)
```

Parameters

`parentScene` Scene

`position` Point3D

`rotation` Vector3

`children` HashSet<SceneObject>

Properties

ParentScene

```
public override Scene ParentScene { get; set; }
```

Property Value

Scene

Methods

GetChildren()

Gets a shallow copy of an array with children of this InnerSceneObject.

```
public SceneObject[] GetChildren()
```

Returns

SceneObject[]

Returns a copy of an array with all children of this InnerSceneObject

IsChild(SceneObject)

Checks if an SceneObject is a child of this SceneObject.

```
public bool IsChild(SceneObject sceneObject)
```

Parameters

`sceneObject` SceneObject

A SceneObject to check if it is a child

Returns

bool

True if the sceneObject is a child

RegisterChild(SceneObject)

Adds SceneObject to list of children.

```
public bool RegisterChild(SceneObject child)
```

Parameters

`child` SceneObject

SceneObject to register as child

Returns

bool

Retruns True if child is succasfully added in and was not registerd before. Otherwise returns False

UnRegisterChild(SceneObject)

Removes SceneObject from list of children.

```
public bool UnRegisterChild(SceneObject child)
```

Parameters

`child` SceneObject

SceneObject to unregister as child

Returns

bool

Retruns True if child is succasfully removed and was not unregisterd before. Otherwise returns False

Struct IntersectionProperties

Namespace: rt004.SceneObjects

Assembly: rt004.dll

```
public record struct IntersectionProperties : IEquatable<IntersectionProperties>
```

Implements
IEquatable<IntersectionProperties>

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Fields

distance

The distance from the ray origin to the intersection point.

```
public double distance
```

Field Value

double

globalPosition

The global position of the intersection point in world coordinates.

```
public Point3D globalPosition
```

Field Value

Point3D

incomingRay

The incoming ray that caused this intersection.

```
public Ray incomingRay
```

Field Value

Ray

intersectedSolid

The solid object that was intersected by the ray.

```
public Solid intersectedSolid
```

Field Value

Solid

normal

The surface normal vector at the intersection point.

```
public Vector3D normal
```

Field Value

Vector3D

uvCoordinates

The UV texture coordinates at the intersection point.

```
public Point2D uvCoordinates
```

Field Value

Point2D

Class LightSource

Namespace: rt004.SceneObjects

Assembly: rt004.dll

Represents an abstract base class for light sources in a scene, defining common properties and methods.

```
public abstract class LightSource : SceneObject
```

Inheritance

object ← SceneObject ← LightSource

Derived

PointLight

Inherited Members

SceneObject.parentScene , SceneObject.dirtyGlobalPosition , SceneObject.Position , SceneObject.GlobalPosition , SceneObject.Rotation , SceneObject.ParentScene , SceneObject.ParentObject , SceneObject.GetLocalHeding() , SceneObject.GetGlobalHeading() , SceneObject.GetGlobalPosition() , SceneObject.GetLocalToWorldTransformMatrix() , SceneObject.GetWorldToLocalTransformMatrix() , SceneObject.GetLocalTransformMatrix() , SceneObject.GetLocalInverseTransformMatrix() , SceneObject.ToWorldSpace(Vector3D) , SceneObject.ToWorldSpace(Point3D) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

LightSource(Scene, Point3D, Vector3, Color4, float, float, float)

Initializes a new instance of the LightSource class with specified parameters.

```
public LightSource(Scene parentScene, Point3D position, Vector3 rotation, Color4 color,
float intensity, float diffuseFactor, float specularFactor)
```

Parameters

`parentScene` Scene

The parent scene to which this light source belongs.

`position` Point3D

The position of the light source in the scene.

`rotation` Vector3

The rotation of the light source in the scene.

`color` Color4

The color of the light emitted by this source.

`intensity` float

The intensity of the light source.

`diffuseFactor` float

The factor for controlling diffuse light.

`specularFactor` float

The factor for controlling specular light.

Fields

diffuseFactor

The factor controlling the diffuse light component for this light source.

```
public float diffuseFactor
```

Field Value

float

specularFactor

The factor controlling the specular light component for this light source.

```
public float specularFactor
```

Field Value

float

Properties

Color

Gets or sets the color of the light emitted by this light source.

```
public Color4 Color { get; set; }
```

Property Value

Color4

LightPower

Gets or sets the intensity of the light source, ensuring a non-negative value.

```
public float LightPower { get; set; }
```

Property Value

float

Methods

DiffuseLightIntensityAt(Point3D, bool)

Computes the diffuse light intensity at a specified position in the scene.

```
public abstract float DiffuseLightIntensityAt(Point3D point, bool areShadowsEnabled)
```

Parameters

`point` Point3D

The position where the intensity is calculated, in global coordinates.

`areShadowsEnabled` bool

Indicates if shadows are considered in the calculation.

Returns

float

The computed diffuse light intensity.

LightIntensityAt(Point3D, bool)

Computes both diffuse and specular light intensities at a specified position.

```
public virtual (float, float) LightIntensityAt(Point3D point, bool areShadowsEnabled)
```

Parameters

`point` Point3D

The position where the intensities are calculated, in global coordinates.

`areShadowsEnabled` bool

Indicates if shadows are considered in the calculation.

Returns

(float, float)

A tuple containing the diffuse and specular light intensities.

SpecularLightIntensityAt(Point3D, bool)

Computes the specular light intensity at a specified position in the scene.

```
public abstract float SpecularLightIntensityAt(Point3D point, bool areShadowsEnabled)
```

Parameters

`point` Point3D

The position where the intensity is calculated, in global coordinates.

`areShadowsEnabled` bool

Indicates if shadows are considered in the calculation.

Returns

float

The computed specular light intensity.

Class PerspectiveCamera

Namespace: rt004.SceneObjects

Assembly: rt004.dll

Represents a perspective camera that renders a 3D scene with a specified field of view (FoV).

```
public class PerspectiveCamera : Camera
```

Inheritance

object ← SceneObject ← Camera ← PerspectiveCamera

Inherited Members

Camera.width , Camera.height , Camera.maxRenderingDistanceSquered , Camera.MaxRenderingDistance , Camera.Width , Camera.Height , Camera.GetResolution() , SceneObject.parentScene , SceneObject.dirtyGlobalPosition , SceneObject.Position , SceneObject.GlobalPosition , SceneObject.Rotation , SceneObject.ParentScene , SceneObject.ParentObject , SceneObject.GetLocalHeding() , SceneObject.GetGlobalHeading() , SceneObject.GetGlobalPosition() , SceneObject.GetLocalToWorldTransformMatrix() , SceneObject.GetWorldToLocalTransformMatrix() , SceneObject.GetLocalTransformMatrix() , SceneObject.GetLocalInverseTransformMatrix() , SceneObject.ToWorldSpace(Vector3D) , SceneObject.ToWorldSpace(Point3D) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PerspectiveCamera(Scene, Point3D, Vector3, Color4, float, uint, uint)

Initializes a new instance of the PerspectiveCamera class with specified parameters.

```
public PerspectiveCamera(Scene parentScene, Point3D position, Vector3 rotation, Color4
backgroundColor, float fov, uint width, uint height)
```

Parameters

`parentScene` Scene

The parent scene that this camera will render.

`position` Point3D

The position of the camera in the scene.

`rotation` Vector3

The rotation of the camera in the scene.

`backgroundColor` Color4

The background color of the camera.

`fov` float

The field of view in radians.

`width` uint

The horizontal resolution of the camera.

`height` uint

The vertical resolution of the camera.

Fields

backgroundColor

The background color of the camera view when no object is rendered.

```
public Color4 backgroundColor
```

Field Value

Color4

Properties

FoV

Gets or sets the field of view in radians. Throws an exception if set outside the range (0, PI).

```
public float FoV { get; set; }
```

Property Value

float

Methods

RenderImage()

Renders an image of the scene from the perspective of this camera in a perspective view.

```
public override FloatImage RenderImage()
```

Returns

FloatImage

A FloatImage containing the rendered image of the scene.

SetResolution(uint, uint)

Sets the resolution of the camera in pixels. Updates the pixel density based on the field of view and resolution.

```
public override void SetResolution(uint width, uint height)
```

Parameters

`width` uint

Number of pixels horizontally.

`height` uint

Number of pixels vertically.

Class Plane

Namespace: rt004.SceneObjects

Assembly: rt004.dll

Represents an infinite plane solid in 3D space for ray tracing and intersection tests.

```
public class Plane : Solid
```

Inheritance

object ← SceneObject ← Solid ← Plane

Inherited Members

Solid.material , Solid.TryGetRayIntersection(Ray, out Point3D) , Solid.TryGetRayIntersection(Ray, out Point3D, out Point2D) , Solid.TryGetRayIntersection(Ray, out IntersectionProperties) , SceneObject.parentScene , SceneObject.dirtyGlobalPosition , SceneObject.Position , SceneObject.GlobalPosition , SceneObject.Rotation , SceneObject.ParentScene , SceneObject.ParentObject , SceneObject.GetLocalHeding() , SceneObject.GetGlobalHeading() , SceneObject.GetGlobalPosition() , SceneObject.GetLocalToWorldTransformMatrix() , SceneObject.GetWorldToLocalTransformMatrix() , SceneObject.GetLocalTransformMatrix() , SceneObject.GetLocalInverseTransformMatrix() , SceneObject.ToWorldSpace(Vector3D) , SceneObject.ToWorldSpace(Point3D) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Plane(Scene, Point3D, Vector3)

Initializes a new instance of the Plane class with the default material for the current lighting model.

```
public Plane(Scene parentScene, Point3D position, Vector3 rotation)
```

Parameters

`parentScene` Scene

The parent scene to which this plane belongs.

`position` Point3D

The position of the plane in world coordinates.

`rotation` Vector3

The rotation of the plane in Euler angles (degrees or radians as per convention).

Plane(Scene, Point3D, Vector3, Material)

Initializes a new instance of the Plane class with a specified material.

```
public Plane(Scene parentScene, Point3D position, Vector3 rotation, Material material)
```

Parameters

`parentScene` Scene

The parent scene to which this plane belongs.

`position` Point3D

The position of the plane in world coordinates.

`rotation` Vector3

The rotation of the plane in Euler angles (degrees or radians as per convention).

`material` Material

The material to use for rendering the plane.

Exceptions

InvalidCastException

Thrown if the provided material is not compatible with the current renderer settings.

Methods

GetNormalAt(Point3D)

Gets the normal vector of the plane at the specified global position.

```
public override Vector3D GetNormalAt(Point3D globalPosition)
```

Parameters

`globalPosition` Point3D

The position on the plane (unused, as the normal is constant).

Returns

Vector3D

The normal vector of the plane.

TryGetRayIntersection(Ray, out double)

Attempts to compute the intersection between a ray and this plane.

```
public override bool TryGetRayIntersection(Ray line, out double parameter)
```

Parameters

`line` Ray

The ray to test for intersection.

`parameter` double

If intersection occurs, contains the distance along the ray to the intersection point.

Returns

bool

True if the ray intersects the plane; otherwise, false.

TryGetRayIntersection(Ray, out double, out Point2D)

Attempts to compute the intersection between a ray and this plane, and calculates the UV coordinates at the intersection point.

```
public override bool TryGetRayIntersection(Ray ray, out double distance, out
Point2D uvCoord)
```

Parameters

`ray` Ray

The ray to test for intersection.

`distance` double

If intersection occurs, contains the distance along the ray to the intersection point.

`uvCoord` Point2D

If intersection occurs, contains the UV coordinates at the intersection point on the plane.

Returns

bool

True if the ray intersects the plane; otherwise, false.

Class PointLight

Namespace: rt004.SceneObjects

Assembly: rt004.dll

```
public class PointLight : LightSource
```

Inheritance

object ← SceneObject ← LightSource ← PointLight

Inherited Members

LightSource.diffuseFactor , LightSource.specularFactor , LightSource.LightPower , LightSource.Color , SceneObject.parentScene , SceneObject.dirtyGlobalPosition , SceneObject.Position , SceneObject.GlobalPosition , SceneObject.Rotation , SceneObject.ParentScene , SceneObject.ParentObject , SceneObject.GetLocalHeding() , SceneObject.GetGlobalHeading() , SceneObject.GetGlobalPosition() , SceneObject.GetLocalToWorldTransformMatrix() , SceneObject.GetWorldToLocalTransformMatrix() , SceneObject.GetLocalTransformMatrix() , SceneObject.GetLocalInverseTransformMatrix() , SceneObject.ToWorldSpace(Vector3D) , SceneObject.ToWorldSpace(Point3D) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PointLight(Scene, Point3D, Vector3, Color4, float, float, float)

```
public PointLight(Scene parentScene, Point3D position, Vector3 rotation, Color4 color, float
intensity, float diffuseFactor, float specularFactor)
```

Parameters

`parentScene` Scene

`position` Point3D

`rotation` Vector3

`color` Color4

`intensity` float

`diffuseFactor` float

`specularFactor` float

Methods

DiffuseLightIntensityAt(Point3D, bool)

Computes diffuse intensity of light from this light at specified point

```
public override float DiffuseLightIntensityAt(Point3D point, bool areShadowsEnabled)
```

Parameters

`point` Point3D

point to compute intensity at, in global coord

`areShadowsEnabled` bool

Returns

float

Returns the intensity

LightIntensityAt(Point3D, bool)

Computes Diffuse and Specular light insnsity at specific point.

```
public override (float, float) LightIntensityAt(Point3D point, bool areShadowsEnabled)
```

Parameters

`point` Point3D

Position where to compute intensity. In global coordinates.

`areShadowsEnabled` bool

Returns

(float, float)

Returns two light intensities (Diffuse, Specular)

SpecularLightIntensityAt(Point3D, bool)

Computes specular intensity of light from this light at specified point

```
public override float SpecularLightIntensityAt(Point3D point, bool areShadowsEnabled)
```

Parameters

`point` Point3D

point to compute intensity at

`areShadowsEnabled` bool

Returns

float

Returns the intensity

Class SceneObject

Namespace: rt004.SceneObjects

Assembly: rt004.dll

```
public abstract class SceneObject
```

Inheritance

object ← SceneObject

Derived

Camera, InnerSceneObject, LightSource, Solid

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

SceneObject(Scene, Point3D, Vector3)

```
public SceneObject(Scene parentScene, Point3D position, Vector3 rotation)
```

Parameters

`parentScene` Scene

`position` Point3D

`rotation` Vector3

Fields

dirtyGlobalPosition

```
protected bool dirtyGlobalPosition
```

Field Value

bool

parentScene

```
protected Scene parentScene
```

Field Value

Scene

Properties

GlobalPosition

```
public Point3D GlobalPosition { get; set; }
```

Property Value

Point3D

ParentObject

```
public InnerSceneObject ParentObject { get; set; }
```

Property Value

InnerSceneObject

ParentScene

```
public virtual Scene ParentScene { get; set; }
```

Property Value

Scene

Position

```
public Point3D Position { get; set; }
```

Property Value

Point3D

Rotation

```
public Vector3 Rotation { get; set; }
```

Property Value

Vector3

Methods

GetGlobalHeading()

```
public Vector3d GetGlobalHeading()
```

Returns

Vector3d

GetGlobalPosition()

```
public Point3D GetGlobalPosition()
```

Returns

Point3D

GetLocalHeding()

```
public Vector3d GetLocalHeding()
```

Returns

Vector3d

GetLocalInverseTransformMatrix()

```
public Matrix4d GetLocalInverseTransformMatrix()
```

Returns

Matrix4d

GetLocalToWorldTransformMatrix()

Gets transform matrix from local to world space

```
public virtual Matrix4d GetLocalToWorldTransformMatrix()
```

Returns

Matrix4d

Matrix containing the transformation

GetLocalTransformMatrix()

```
public Matrix4d GetLocalTransformMatrix()
```

Returns

Matrix4d

GetWorldToLocalTransformMatrix()

Gets transform matrix from world to local space

```
public virtual Matrix4d GetWorldToLocalTransformMatrix()
```

Returns

Matrix4d

Matrix containing the transformation

ToWorldSpace(Point3D)

```
public Point3D ToWorldSpace(Point3D pointInObjectSpace)
```

Parameters

`pointInObjectSpace` Point3D

Returns

Point3D

ToWorldSpace(Vector3D)

```
public Vector3D ToWorldSpace(Vector3D vectorInObjectSpace)
```

Parameters

`vectorInObjectSpace` Vector3D

Returns

Vector3D

Class Solid

Namespace: rt004.SceneObjects

Assembly: rt004.dll

Abstract base class for all solid objects that can be rendered in a scene. Provides common functionality for ray-solid intersection tests and material properties.

```
public abstract class Solid : SceneObject
```

Inheritance

object ← SceneObject ← Solid

Derived

Plane, Sphere

Inherited Members

SceneObject.parentScene , SceneObject.dirtyGlobalPosition , SceneObject.Position , SceneObject.GlobalPosition , SceneObject.Rotation , SceneObject.ParentScene , SceneObject.ParentObject , SceneObject.GetLocalHeding() , SceneObject.GetGlobalHeading() , SceneObject.GetGlobalPosition() , SceneObject.GetLocalToWorldTransformMatrix() , SceneObject.GetWorldToLocalTransformMatrix() , SceneObject.GetLocalTransformMatrix() , SceneObject.GetLocalInverseTransformMatrix() , SceneObject.ToWorldSpace(Vector3D) , SceneObject.ToWorldSpace(Point3D) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Solid(Scene, Point3D, Vector3)

Initializes a new instance of the Solid class with default material.

```
public Solid(Scene parentScene, Point3D position, Vector3 rotation)
```

Parameters

`parentScene` Scene

The scene that contains this solid.

`position` Point3D

The position of the solid in 3D space.

`rotation` Vector3

The rotation of the solid as Euler angles.

Solid(Scene, Point3D, Vector3, Material)

Initializes a new instance of the Solid class with a specified material.

```
public Solid(Scene parentScene, Point3D position, Vector3 rotation, Material material)
```

Parameters

`parentScene` Scene

The scene that contains this solid.

`position` Point3D

The position of the solid in 3D space.

`rotation` Vector3

The rotation of the solid as Euler angles.

`material` Material

The material to assign to this solid.

Exceptions

InvalidCastException

Thrown when the material is not compatible with the current light model.

Properties

material

Gets or sets the material that defines the visual properties of this solid.

```
public Material material { get; set; }
```

Property Value

Material

Methods

GetNormalAt(Point3D)

Gets normal of this object at specified position.

```
public abstract Vector3D GetNormalAt(Point3D globalPosition)
```

Parameters

`globalPosition` Point3D

Position, where the normal should be start on the solid side or edge. In world coordinates

Returns

Vector3D

Returns normal of the object at specified position

TryGetRayIntersection(Ray, out double)

Tests whether a ray intersects with this solid and returns the distance to the intersection point.

```
public abstract bool TryGetRayIntersection(Ray ray, out double distance)
```

Parameters

`ray` Ray

The ray to test for intersection.

`distance` double

When this method returns, contains the distance from the ray origin to the intersection point if an intersection exists.

Returns

bool

`true` if the ray intersects with this solid; otherwise, `false`.

TryGetRayIntersection(Ray, out double, out Point2D)

Tests whether a ray intersects with this solid and returns the distance and UV coordinates of the intersection.

```
public abstract bool TryGetRayIntersection(Ray ray, out double distance, out
Point2D uvCoord)
```

Parameters

`ray` Ray

The ray to test for intersection.

`distance` double

When this method returns, contains the distance from the ray origin to the intersection point if an intersection exists.

`uvCoord` Point2D

When this method returns, contains the UV texture coordinates at the intersection point if an intersection exists.

Returns

bool

`true` if the ray intersects with this solid; otherwise, `false`.

TryGetRayIntersection(Ray, out IntersectionProperties)

Tests whether a ray intersects with this solid and returns complete intersection properties.

```
public bool TryGetRayIntersection(Ray ray, out IntersectionProperties intersection)
```

Parameters

`ray` Ray

The ray to test for intersection.

`intersection` IntersectionProperties

When this method returns, contains the complete intersection properties if an intersection exists.

Returns

bool

`true` if the ray intersects with this solid; otherwise, `false`.

TryGetRayIntersection(Ray, out Point3D)

Computes closest intersection from line.Position

```
public bool TryGetRayIntersection(Ray ray, out Point3D intersection)
```

Parameters

`ray` Ray

Ray to intersect with

`intersection` Point3D

closest intersection point

Returns

bool

true if ray intersects with the object, otherwise false

TryGetRayIntersection(Ray, out Point3D, out Point2D)

Tests whether a ray intersects with this solid and returns the intersection point and UV coordinates.

```
public bool TryGetRayIntersection(Ray ray, out Point3D intersection, out Point2D uvCoord)
```

Parameters

`ray` Ray

The ray to test for intersection.

`intersection` Point3D

When this method returns, contains the 3D intersection point if an intersection exists.

`uvCoord` Point2D

When this method returns, contains the UV texture coordinates at the intersection point if an intersection exists.

Returns

bool

`true` if the ray intersects with this solid; otherwise, `false`.

Class Sphere

Namespace: rt004.SceneObjects

Assembly: rt004.dll

Represents a mathematically perfect sphere solid object in a 3D scene. Provides ray-sphere intersection calculations and surface normal computation.

```
public class Sphere : Solid
```

Inheritance

object ← SceneObject ← Solid ← Sphere

Inherited Members

Solid.material , Solid.TryGetRayIntersection(Ray, out Point3D) , Solid.TryGetRayIntersection(Ray, out Point3D, out Point2D) , Solid.TryGetRayIntersection(Ray, out IntersectionProperties) , SceneObject.parentScene , SceneObject.dirtyGlobalPosition , SceneObject.Position , SceneObject.GlobalPosition , SceneObject.Rotation , SceneObject.ParentScene , SceneObject.ParentObject , SceneObject.GetLocalHeding() , SceneObject.GetGlobalHeading() , SceneObject.GetGlobalPosition() , SceneObject.GetLocalToWorldTransformMatrix() , SceneObject.GetWorldToLocalTransformMatrix() , SceneObject.GetLocalTransformMatrix() , SceneObject.GetLocalInverseTransformMatrix() , SceneObject.ToWorldSpace(Vector3D) , SceneObject.ToWorldSpace(Point3D) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Sphere(Scene, Point3D, Vector3, Material, float)

Initializes a new instance of the Sphere class.

```
public Sphere(Scene parentScene, Point3D position, Vector3 rotation, Material material,
float radius)
```

Parameters

`parentScene` Scene

The scene that contains this sphere.

`position` Point3D

The position of the sphere's center in 3D space.

`rotation` Vector3

The rotation of the sphere as Euler angles.

`material` Material

The material that defines the visual properties of the sphere.

`radius` float

The radius of the sphere.

Fields

radius

The radius of the sphere.

```
public readonly float radius
```

Field Value

float

Methods

GetNormalAt(Point3D)

Gets the surface normal vector at the specified position on the sphere.

```
public override Vector3D GetNormalAt(Point3D position)
```

Parameters

`position` Point3D

The position on the sphere surface in world coordinates.

Returns

Vector3D

The normalized surface normal vector pointing outward from the sphere at the specified position.

TryGetRayIntersection(Ray, out double)

Tests whether a ray intersects with this sphere and returns the distance to the closest intersection point. Uses the quadratic formula to solve for ray-sphere intersection.

```
public override bool TryGetRayIntersection(Ray ray, out double parameter)
```

Parameters

`ray` Ray

The ray to test for intersection.

`parameter` double

When this method returns, contains the distance from the ray origin to the closest intersection point if an intersection exists.

Returns

bool

`true` if the ray intersects with the sphere and the intersection is in front of the ray origin; otherwise, `false`.

TryGetRayIntersection(Ray, out double, out Point2D)

Tests whether a ray intersects with this sphere and returns both the distance and UV texture coordinates. Calculates UV coordinates based on spherical coordinate mapping.

```
public override bool TryGetRayIntersection(Ray ray, out double distance, out
Point2D uvCoord)
```

Parameters

`ray` Ray

The ray to test for intersection.

`distance` double

When this method returns, contains the distance from the ray origin to the intersection point if an intersection exists.

`uvCoord` Point2D

When this method returns, contains the UV texture coordinates at the intersection point if an intersection exists.

Returns

bool

`true` if the ray intersects with the sphere; otherwise, `false`.

Namespace rt004.SceneObjects.Loading

Classes

CameraLoader
: Abstract base class for loading camera configurations from XML or other serialized formats.

InnerSceneObjectLoader

LightSourceLoader
: Abstract base class for loading light sources from serialized data, with common properties for light configuration.

PerspectiveCameraLoader
: Responsible for loading a PerspectiveCamera from specified parameters.

PlaneLoader
: Loader class for deserializing and instantiating Plane objects from scene data.

PointLightLoader

SceneObjectLoader
: Abstract class used for loading of SceneObject. SceneObjectLoader inheratance structure must be parallel to the SceneObject one.

SolidLoader
: Abstract base class for loading solid objects from configuration data. Supports XML serialization for different types of solid loaders.

SphereLoader
: Provides functionality for loading sphere objects from configuration data. Supports XML serialization and deserialization of sphere properties.

Class CameraLoader

Namespace: rt004.SceneObjects.Loading

Assembly: rt004.dll

Abstract base class for loading camera configurations from XML or other serialized formats.

```
public abstract class CameraLoader : SceneObjectLoader
```

Inheritance

object ← SceneObjectLoader ← CameraLoader

Derived

PerspectiveCameraLoader

Inherited Members

SceneObjectLoader.position , SceneObjectLoader.rotation , SceneObjectLoader.CreateInstance(Scene) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

CameraLoader()

Initializes a new instance of the CameraLoader class with default settings.

```
public CameraLoader()
```

CameraLoader(Point3D, Vector3, Color4, uint, uint)

Initializes a new instance of the CameraLoader class with specified parameters.

```
public CameraLoader(Point3D position, Vector3 rotation, Color4 backgroundColor, uint width,
uint height)
```

Parameters

`position` Point3D

The position of the camera in the scene.

`rotation` Vector3

The rotation of the camera in the scene.

`backgroundColor` Color4

The background color for the camera.

`width` uint

The width of the camera's resolution in pixels.

`height` uint

The height of the camera's resolution in pixels.

Fields

backgroundColor

The background color of the camera view.

```
public Color4 backgroundColor
```

Field Value

Color4

height

The height of the camera, defaulting to the settings defined in RendererSettings.

```
public uint height
```

Field Value

uint

width

The width of the camera, defaulting to the settings defined in RendererSettings.

```
public uint width
```

Field Value

uint

Class InnerSceneObjectLoader

Namespace: rt004.SceneObjects.Loading

Assembly: rt004.dll

```
public class InnerSceneObjectLoader : SceneObjectLoader
```

Inheritance

object ← SceneObjectLoader ← InnerSceneObjectLoader

Inherited Members

SceneObjectLoader.position , SceneObjectLoader.rotation , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

InnerSceneObjectLoader()

```
public InnerSceneObjectLoader()
```

InnerSceneObjectLoader(Point3D, Vector3)

```
public InnerSceneObjectLoader(Point3D position, Vector3 rotation)
```

Parameters

`position` Point3D

`rotation` Vector3

Fields

children

```
public SceneObjectLoader[] children
```

Field Value

SceneObjectLoader[]

Methods

CreateInstance(Scene)

Creates instance of corresponding SceneObject type

```
public override SceneObject CreateInstance(Scene parentScene)
```

Parameters

`parentScene` Scene

Returns

SceneObject

instanciated object

Class LightSourceLoader

Namespace: rt004.SceneObjects.Loading

Assembly: rt004.dll

Abstract base class for loading light sources from serialized data, with common properties for light configuration.

```
public abstract class LightSourceLoader : SceneObjectLoader
```

Inheritance

object ← SceneObjectLoader ← LightSourceLoader

Derived

PointLightLoader

Inherited Members

SceneObjectLoader.position , SceneObjectLoader.rotation , SceneObjectLoader.CreateInstance(Scene) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

LightSourceLoader()

Initializes a new instance of the LightSourceLoader class with default settings.

```
public LightSourceLoader()
```

LightSourceLoader(Point3D, Vector3, Color4, float, float, float)

Initializes a new instance of the LightSourceLoader class with specified parameters.

```
public LightSourceLoader(Point3D position, Vector3 rotation, Color4 lightColor, float
intensity, float diffuseFactor, float specularFactor)
```

Parameters

`position` Point3D

The position of the light source in the scene.

`rotation` Vector3

The rotation of the light source in the scene.

`lightColor` Color4

The color of the light emitted by this source.

`intensity` float

The intensity of the light source.

`diffuseFactor` float

The factor for controlling diffuse light.

`specularFactor` float

The factor for controlling specular light.

Fields

diffuseFactor

The diffuse factor for the light source, defaulting to 0.5.

```
public float diffuseFactor
```

Field Value

float

intensity

The intensity of the light source, defaulting to 1.

```
public float intensity
```

Field Value

float

lightColor

The color of the light emitted by this source, defaulting to white.

```
public Color4 lightColor
```

Field Value

Color4

specularFactor

The specular factor for the light source, defaulting to 0.5.

```
public float specularFactor
```

Field Value

float

Class PerspectiveCameraLoader

Namespace: rt004.SceneObjects.Loading

Assembly: rt004.dll

Responsible for loading a PerspectiveCamera from specified parameters.

```
public class PerspectiveCameraLoader : CameraLoader
```

Inheritance

object ← SceneObjectLoader ← CameraLoader ← PerspectiveCameraLoader

Inherited Members

CameraLoader.width , CameraLoader.height , CameraLoader.backgroundColor , SceneObjectLoader.position , SceneObjectLoader.rotation , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PerspectiveCameraLoader()

Initializes a new instance of the PerspectiveCameraLoader class.

```
public PerspectiveCameraLoader()
```

PerspectiveCameraLoader(Point3D, Vector3, Color4, float, uint, uint)

Initializes a new instance of the PerspectiveCameraLoader class with specified parameters.

```
public PerspectiveCameraLoader(Point3D position, Vector3 rotation, Color4 backgroundColor,
float fov, uint width, uint height)
```

Parameters

`position` Point3D

The position of the camera in the scene.

`rotation` Vector3

The rotation of the camera in the scene.

`backgroundColor` Color4

The background color for the camera.

`fov` float

The field of view in degrees.

`width` uint

The horizontal resolution of the camera.

`height` uint

The vertical resolution of the camera.

Fields

fov

The field of view to be loaded in degrees.

```
public float fov
```

Field Value

float

lightModelName

The name of the light model used in this camera setup.

```
public string lightModelName
```

Field Value

string

Methods

CreateInstance(Scene)

Creates an instance of PerspectiveCamera within the specified parent scene. Converts field of view from degrees to radians if it has not yet been converted.

```
public override SceneObject CreateInstance(Scene parentScene)
```

Parameters

`parentScene` Scene

The parent scene in which this camera instance will be created.

Returns

SceneObject

A new instance of PerspectiveCamera configured with the loader's settings.

Class PlaneLoader

Namespace: rt004.SceneObjects.Loading

Assembly: rt004.dll

Loader class for deserializing and instantiating Plane objects from scene data.

```
public class PlaneLoader : SolidLoader
```

Inheritance

object ← SceneObjectLoader ← SolidLoader ← PlaneLoader

Inherited Members

SolidLoader.material , SceneObjectLoader.position , SceneObjectLoader.rotation , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PlaneLoader()

Initializes a new instance of the PlaneLoader class.

```
public PlaneLoader()
```

PlaneLoader(Point3D, Vector3, MaterialLoader)

Initializes a new instance of the PlaneLoader class with specified parameters.

```
public PlaneLoader(Point3D position, Vector3 rotation, MaterialLoader material)
```

Parameters

`position` Point3D

The position of the plane.

`rotation` Vector3

The rotation of the plane.

`material` MaterialLoader

The material loader for the plane.

Methods

CreateInstance(Scene)

Creates an instance of a Plane using the loader's configuration.

```
public override SceneObject CreateInstance(Scene parentScene)
```

Parameters

`parentScene` Scene

The parent scene to which the plane will be added.

Returns

SceneObject

A new Plane instance.

Class PointLightLoader

Namespace: rt004.SceneObjects.Loading

Assembly: rt004.dll

```
public class PointLightLoader : LightSourceLoader
```

Inheritance

object ← SceneObjectLoader ← LightSourceLoader ← PointLightLoader

Inherited Members

LightSourceLoader.intensity , LightSourceLoader.lightColor , LightSourceLoader.diffuseFactor , LightSourceLoader.specularFactor , SceneObjectLoader.position , SceneObjectLoader.rotation , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PointLightLoader()

```
public PointLightLoader()
```

PointLightLoader(Point3D, Vector3, Color4, float, float, float)

```
public PointLightLoader(Point3D position, Vector3 rotation, Color4 color, float intensity,
float diffuseFactor, float specularFactor)
```

Parameters

`position` Point3D

`rotation` Vector3

`color` Color4

`intensity` float

`diffuseFactor` float

`specularFactor` float

Methods

CreateInstance(Scene)

Creates instance of corresponding SceneObject type

```
public override SceneObject CreateInstance(Scene parentScene)
```

Parameters

`parentScene` Scene

Returns

SceneObject

instanciated object

Class SceneObjectLoader

Namespace: rt004.SceneObjects.Loading

Assembly: rt004.dll

Abstract class used for loading of SceneObject. SceneObjectLoader inheratance structure must be parallel to the SceneObject one.

```
public abstract class SceneObjectLoader
```

Inheritance

object ← SceneObjectLoader

Derived

CameraLoader, InnerSceneObjectLoader, LightSourceLoader, SolidLoader

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

SceneObjectLoader()

```
public SceneObjectLoader()
```

SceneObjectLoader(Point3D, Vector3)

```
public SceneObjectLoader(Point3D position, Vector3 rotation)
```

Parameters

`position` Point3D

`rotation` Vector3

Fields

position

```
public Vector3 position
```

Field Value

Vector3

rotation

```
public Vector3 rotation
```

Field Value

Vector3

Methods

CreateInstance(Scene)

Creates instance of corresponding SceneObject type

```
public abstract SceneObject CreateInstance(Scene parentScene)
```

Parameters

`parentScene` Scene

Returns

SceneObject

instanciated object

Class SolidLoader

Namespace: rt004.SceneObjects.Loading

Assembly: rt004.dll

Abstract base class for loading solid objects from configuration data. Supports XML serialization for different types of solid loaders.

```
public abstract class SolidLoader : SceneObjectLoader
```

Inheritance

object ← SceneObjectLoader ← SolidLoader

Derived

PlaneLoader, SphereLoader

Inherited Members

SceneObjectLoader.position , SceneObjectLoader.rotation , SceneObjectLoader.CreateInstance(Scene) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

SolidLoader()

Initializes a new instance of the SolidLoader class.

```
public SolidLoader()
```

SolidLoader(Point3D, Vector3, MaterialLoader)

Initializes a new instance of the SolidLoader class with specified parameters.

```
public SolidLoader(Point3D position, Vector3 rotation, MaterialLoader material)
```

Parameters

`position` Point3D

The position of the solid in 3D space.

`rotation` Vector3

The rotation of the solid as Euler angles.

`material` MaterialLoader

The material loader for creating the solid's material.

Fields

material

Gets or sets the material loader that defines how to create the material for the solid.

```
public MaterialLoader material
```

Field Value

MaterialLoader

Class SphereLoader

Namespace: rt004.SceneObjects.Loading

Assembly: rt004.dll

Provides functionality for loading sphere objects from configuration data. Supports XML serialization and deserialization of sphere properties.

```
public class SphereLoader : SolidLoader
```

Inheritance

object ← SceneObjectLoader ← SolidLoader ← SphereLoader

Inherited Members

SolidLoader.material , SceneObjectLoader.position , SceneObjectLoader.rotation , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

SphereLoader()

Initializes a new instance of the SphereLoader class.

```
public SphereLoader()
```

SphereLoader(Point3D, Vector3, MaterialLoader, float)

Initializes a new instance of the SphereLoader class with specified parameters.

```
public SphereLoader(Point3D position, Vector3 rotation, MaterialLoader material,
float diameter)
```

Parameters

`position` Point3D

The position of the sphere in 3D space.

`rotation` Vector3

The rotation of the sphere as Euler angles.

`material` MaterialLoader

The material loader for creating the sphere's material.

`diameter` float

The diameter of the sphere.

Fields

diameter

Gets or sets the diameter of the sphere to be created.

```
public float diameter
```

Field Value

float

Methods

CreateInstance(Scene)

Creates a new Sphere instance from the loader's configuration.

```
public override SceneObject CreateInstance(Scene parentScene)
```

Parameters

`parentScene` Scene

The scene that will contain the created sphere.

Returns

SceneObject

A new Sphere instance configured with the loader's properties.

Namespace rt004.Util

Classes

ArgumentParser

Provides static methods for parsing command line arguments and XML configuration files. Handles variable substitution and creates scene objects from configuration data.

LightModelExtensions

Structs

ArgumentParser.DataLoader

Data structure to facilitate loading from XML file. Contains all necessary information for scene creation and rendering configuration.

BasicPlane

represents Plane in 3D

Point2D

Represents a 2D point with double precision, defined by X and Y coordinates.

Point3D

Represents a 3D point with double precision, defined by X, Y, and Z coordinates.

Ray

Represents a line parametricly in 3D using Point and Vector

Vector2D

Represents an immutable 2-dimensional vector.

Vector3D

Represents an immutable 3-dimensional vector.

Enums

LightModel

Class ArgumentParser

Namespace: rt004.Util

Assembly: rt004.dll

Provides static methods for parsing command line arguments and XML configuration files. Handles variable substitution and creates scene objects from configuration data.

```
public static class ArgumentParser
```

Inheritance

object ← ArgumentParser

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

ParseScene(string[], string)

Parses config file to scene, output path and saves rendererSettings.

```
public static (Scene, string) ParseScene(string[] args, string configFile = "config.xml")
```

Parameters

`args` string[]

Command line parameters

`configFile` string

Default path to config file

Returns

(Scene, string)

Returns tuple (initialized Scene, output path where to save rendered image)

replaceVariables(string, Dictionary<string, string>)

Replaces all variable instances in input with variable contents

```
public static string replaceVariables(string input, Dictionary<string,
string> variableDefinitions)
```

Parameters

`input` string

The whole input to replace variables with

`variableDefinitions` Dictionary<string, string>

definitions of variables and their contents

Returns

string

Returns input with replaced variables

Struct ArgumentParser.DataLoader

Namespace: rt004.Util

Assembly: rt004.dll

Data structure to facilitate loading from XML file. Contains all necessary information for scene creation and rendering configuration.

```
public struct ArgumentParser.DataLoader
```

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Fields

output

Gets or sets the output file path where the rendered image will be saved.

```
public string output
```

Field Value

string

rendererSettings

Gets or sets the renderer settings loader that contains rendering configuration options.

```
public RendererSettingsLoader rendererSettings
```

Field Value

RendererSettingsLoader

sceneLoader

Gets or sets the scene loader that contains scene object definitions and hierarchy.

```
public SceneLoader sceneLoader
```

Field Value

SceneLoader

Struct BasicPlane

Namespace: rt004.Util

Assembly: rt004.dll

represents Plane in 3D

```
public record struct BasicPlane : IEquatable<BasicPlane>
```

Implements
IEquatable<BasicPlane>

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

BasicPlane(Point3D, Vector3D)

Creates BasicPlane based on point on the plane and normal of the plane.

```
public BasicPlane(Point3D pointOnPlane, Vector3D normal)
```

Parameters

`pointOnPlane` Point3D

Point on the plane as center of rotation

`normal` Vector3D

Normal of the plane

Fields

Normal

Normal vector represented as vector with lenght of 1

```
public readonly Vector3D Normal
```

Field Value

Vector3D

PointOnPlane

```
public readonly Point3D PointOnPlane
```

Field Value

Point3D

Methods

GetAxisOnPlane()

Gets to axies on the plane

```
public (Vector3D axisA, Vector3D axisB) GetAxisOnPlane()
```

Returns

(Vector3D axisA, Vector3D axisB)

pair of vectors at the plane penpendicular to each other

GetD()

computes constant d from general equation of a plane (aX + bY + cZ + d = 0)

```
public double GetD()
```

Returns

double

Returns constant d

Enum LightModel

Namespace: rt004.Util

Assembly: rt004.dll

```
public enum LightModel
```

Extension Methods

LightModelExtensions.GetLightModelComputation(LightModel)

Fields

`PhongModel = 0`

Class LightModelExtensions

Namespace: rt004.Util

Assembly: rt004.dll

```
public static class LightModelExtensions
```

Inheritance

object ← LightModelExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

GetLightModelComputation(LightModel)

```
public static LightModelComputation GetLightModelComputation(this LightModel lightModel)
```

Parameters

`lightModel` LightModel

Returns

LightModelComputation

Struct Point2D

Namespace: rt004.Util

Assembly: rt004.dll

Represents a 2D point with double precision, defined by X and Y coordinates.

```
public readonly struct Point2D
```

Inherited Members
object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

Point2D(Vector2)

Initializes a new instance of the Point2D struct from a OpenTK.Mathematics.Vector2.

```
public Point2D(Vector2 position)
```

Parameters

`position` Vector2

The position of the point.

Point2D(Vector2d)

Initializes a new instance of the Point2D struct from a OpenTK.Mathematics.Vector2d.

```
public Point2D(Vector2d position)
```

Parameters

`position` Vector2d

The position of the point.

Point2D(double, double)

Initializes a new instance of the Point2D struct from the specified X and Y coordinates.

```
public Point2D(double x, double y)
```

Parameters

x double

The X coordinate of the point.

y double

The Y coordinate of the point.

Properties

AsVector

Gets the position of the point as a OpenTK.Mathematics.Vector2d.

```
public Vector2d AsVector { get; }
```

Property Value

Vector2d

DistanceFromOrigin

Gets the distance of the point from the origin.

```
public double DistanceFromOrigin { get; }
```

Property Value

double

DistanceFromOriginSquared

Gets the square of the distance of the point from the origin.

```
public double DistanceFromOriginSquared { get; }
```

Property Value

double

X

Gets the X coordinate of the point.

```
public double X { get; }
```

Property Value

double

Y

Gets the Y coordinate of the point.

```
public double Y { get; }
```

Property Value

double

Zero

Gets a Point2D instance representing the point (0, 0).

```
public static Point2D Zero { get; }
```

Property Value

Point2D

Methods

Equals(object?)

Determines whether two Point2D instances are equal by comparing their X and Y coordinates.

```
public override bool Equals(object? obj)
```

Parameters

`obj` object

The object to compare to.

Returns

bool

True if the objects are equal, false otherwise.

Equals(Point2D)

Determines whether this Point2D is equal to another Point2D.

```
public bool Equals(Point2D other)
```

Parameters

`other` Point2D

The Point2D to compare with the current instance.

Returns

bool

`true` if the specified Point2D has the same X and Y values as the current instance; otherwise, `false`.

GetHashCode()

Returns the hash code for this instance.

```
public override int GetHashCode()
```

Returns

int

An integer representing the hash code.

RoundDown()

Rounds the X and Y coordinates of the point down to the nearest integer.

```
public (int, int) RoundDown()
```

Returns

(int, int)

A tuple containing the rounded X and Y coordinates.

RoundDown(Point2D)

Rounds the X and Y coordinates of a given point down.

```
public static Point2D RoundDown(Point2D point)
```

Parameters

`point` Point2D

The point to round down.

Returns

Point2D

A new Point2D instance with the rounded coordinates.

RoundUp()

Rounds the X and Y coordinates of the point up to the nearest integer.

```
public (int, int) RoundUp()
```

Returns

(int, int)

A tuple containing the rounded X and Y coordinates.

RoundUp(Point2D)

Rounds the X and Y coordinates of a given point up.

```
public static Point2D RoundUp(Point2D point)
```

Parameters

`point` Point2D

The point to round up.

Returns

Point2D

A new Point2D instance with the rounded coordinates.

ToString()

Returns a string that represents the current point.

```
public override string ToString()
```

Returns

string

A string in the format "[X, Y]".

Operators

operator +(Point2D, Vector2D)

Adds a Vector2D to a Point2D.

```
public static Point2D operator +(Point2D point, Vector2D vector)
```

Parameters

`point` Point2D

The Point2D instance.

`vector` Vector2D

The Vector2D instance.

Returns

Point2D

A new Point2D resulting from the addition.

operator /(Point2D, double)

Divides a Point2D by a scalar value.

```
public static Point2D operator /(Point2D point, double scale)
```

Parameters

`point` Point2D

The Point2D instance.

`scale` double

The scalar value to divide by.

Returns

Point2D

A new Point2D resulting from the division.

operator /(Point2D, Vector2D)

Divides a Point2D by a OpenTK.Mathematics.Vector2d.

```
public static Point2D operator /(Point2D p, Vector2D scale)
```

Parameters

`p` Point2D

The Point2D instance.

`scale` Vector2D

The OpenTK.Mathematics.Vector2d to divide by.

Returns

Point2D

A new Point2D resulting from the division.

operator ==(Point2D, Point2D)

Compares two Point2D instances for equality.

```
public static bool operator ==(Point2D value, Point2D other)
```

Parameters

`value` Point2D

The first Point2D instance.

`other` Point2D

The second Point2D instance.

Returns

bool

True if the X and Y coordinates are equal, false otherwise.

explicit operator Vector2(Point2D)

Explicitly converts a Point2D to a OpenTK.Mathematics.Vector2.

```
public static explicit operator Vector2(Point2D dir)
```

Parameters

`dir` Point2D

The Point2D instance to convert.

Returns

Vector2

explicit operator Vector2d(Point2D)

Explicitly converts a Point2D to a OpenTK.Mathematics.Vector2d.

```
public static explicit operator Vector2d(Point2D dir)
```

Parameters

`dir` Point2D

The Point2D instance to convert.

Returns

Vector2d

operator !=(Point2D, Point2D)

Compares two Point2D instances for inequality.

```
public static bool operator !=(Point2D value, Point2D other)
```

Parameters

`value` Point2D

The first Point2D instance.

`other` Point2D

The second Point2D instance.

Returns

bool

True if the X and Y coordinates are not equal, false otherwise.

operator *(Point2D, double)

Multiplies a Point2D by a scalar value.

```
public static Point2D operator *(Point2D point, double scale)
```

Parameters

`point` Point2D

The Point2D instance.

`scale` double

The scalar value to multiply by.

Returns

Point2D

A new Point2D resulting from the multiplication.

operator *(Point2D, Vector2D)

Multiplies a Point2D by a OpenTK.Mathematics.Vector2d.

```
public static Point2D operator *(Point2D point, Vector2D scale)
```

Parameters

`point` Point2D

The Point2D instance.

`scale` Vector2D

The OpenTK.Mathematics.Vector2d to multiply by.

Returns

Point2D

A new Point2D resulting from the multiplication.

operator -(Point2D, Point2D)

Subtracts one Point2D from another, returning the result as a Vector2D.

```
public static Vector2D operator -(Point2D point1, Point2D point2)
```

Parameters

`point1` Point2D

The first Point2D.

`point2` Point2D

The second Point2D.

Returns

Vector2D

A Vector2D representing the difference.

operator -(Point2D, Vector2D)

Subtracts a Vector2D from a Point2D.

```
public static Point2D operator -(Point2D position, Vector2D direction)
```

Parameters

`position` Point2D

The Point2D position.

`direction` Vector2D

The Vector2D direction.

Returns

Point2D

A new Point2D resulting from the subtraction.

operator -(Point2D)

Negates the coordinates of this Point2D, returning a new Point2D with the opposite values.

```
public static Point2D operator -(Point2D point)
```

Parameters

`point` Point2D

The Point2D instance to negate.

Returns

Point2D

A new Point2D with X and Y values negated.

Struct Point3D

Namespace: rt004.Util

Assembly: rt004.dll

Represents a 3D point with double precision, defined by X, Y, and Z coordinates.

```
public struct Point3D
```

Inherited Members
object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

Point3D(Vector3d)

Initializes a new instance of the Point3D struct from a OpenTK.Mathematics.Vector3d.

```
public Point3D(Vector3d position)
```

Parameters

`position` Vector3d

The position of the point.

Point3D(double, double, double)

Initializes a new instance of the Point3D struct from the specified X, Y, and Z coordinates.

```
public Point3D(double x, double y, double z)
```

Parameters

`x` double

The X coordinate of the point.

y double

The Y coordinate of the point.

z double

The Z coordinate of the point.

Properties

AsVector

Gets the position of the point as a OpenTK.Mathematics.Vector3d.

```
public readonly Vector3d AsVector { get; }
```

Property Value

Vector3d

DistanceFromOrigin

Gets the distance of the point from the origin.

```
public readonly double DistanceFromOrigin { get; }
```

Property Value

double

DistanceFromOriginSquared

Gets the square of the distance of the point from the origin.

```
public readonly double DistanceFromOriginSquared { get; }
```

Property Value

double

X

Gets the X coordinate of the point.

```
public double X { get; }
```

Property Value

double

Y

Gets the Y coordinate of the point.

```
public double Y { get; }
```

Property Value

double

Z

Gets the Z coordinate of the point.

```
public double Z { get; }
```

Property Value

double

Zero

Gets a Point3D instance representing the point (0, 0, 0).

```
public static Point3D Zero { get; }
```

Property Value

Point3D

Methods

Equals(object?)

Determines whether two Point3D instances are equal by comparing their X, Y, and Z coordinates.

```
public override bool Equals(object? obj)
```

Parameters

`obj` object

The object to compare to.

Returns

bool

True if the objects are equal, false otherwise.

Equals(Point3D)

Determines whether this Point3D is equal to another Point3D.

```
public bool Equals(Point3D other)
```

Parameters

`other` Point3D

The Point3D to compare with the current instance.

Returns

bool

`true` if the specified Point3D has the same X, Y, and Z values as the current instance; otherwise, `false`.

GetHashCode()

Returns the hash code for this instance.

```
public override int GetHashCode()
```

Returns

int

An integer representing the hash code.

RoundDown()

Rounds the X, Y, and Z coordinates of the point down to the nearest integer.

```
public (int, int, int) RoundDown()
```

Returns

(int, int, int)

A tuple containing the rounded X, Y, and Z coordinates.

RoundDown(Point3D)

Rounds the X, Y, and Z coordinates of a given point down.

```
public static Point3D RoundDown(Point3D point)
```

Parameters

`point` Point3D

The point to round down.

Returns

Point3D

A new Point3D instance with the rounded coordinates.

RoundUp()

Rounds the X, Y, and Z coordinates of the point up to the nearest integer.

```
public (int, int, int) RoundUp()
```

Returns

(int, int, int)

A tuple containing the rounded X, Y, and Z coordinates.

RoundUp(Point3D)

Rounds the X, Y, and Z coordinates of a given point up.

```
public static Point3D RoundUp(Point3D point)
```

Parameters

`point` Point3D

The point to round up.

Returns

Point3D

A new Point3D instance with the rounded coordinates.

ToString()

Returns a string that represents the current point.

```
public override string ToString()
```

Returns

string

A string in the format "[X, Y, Z]".

Transform(Point3D, Matrix4d)

Transforms a Point3D by applying the specified transformation matrix.

```
public static Point3D Transform(Point3D point, Matrix4d transformation)
```

Parameters

`point` Point3D

The Point3D to transform.

`transformation` Matrix4d

The transformation matrix to apply.

Returns

Point3D

A new Point3D resulting from the transformation.

Operators

operator +(Point3D, Vector3D)

Adds a Vector3D to a Point3D.

```
public static Point3D operator +(Point3D point, Vector3D vector)
```

Parameters

`point` Point3D

The Point3D instance.

`vector` Vector3D

The Vector3D instance.

Returns

Point3D

A new Point3D resulting from the addition.

operator /(Point3D, double)

Divides a Point3D by a scalar value.

```
public static Point3D operator /(Point3D point, double scale)
```

Parameters

`point` Point3D

The Point3D instance.

`scale` double

The scalar value to divide by.

Returns

Point3D

A new Point3D resulting from the division.

operator /(Point3D, Vector3D)

Divides a Point3D by a OpenTK.Mathematics.Vector3d.

```
public static Point3D operator /(Point3D point, Vector3D scale)
```

Parameters

`point` Point3D

The Point3D instance.

`scale` Vector3D

The OpenTK.Mathematics.Vector3d to divide by.

Returns

Point3D

A new Point3D resulting from the division.

operator ==(Point3D, Point3D)

Compares two Point3D instances for equality.

```
public static bool operator ==(Point3D value, Point3D other)
```

Parameters

`value` Point3D

The first Point3D instance.

`other` Point3D

The second Point3D instance.

Returns

bool

True if the X, Y, and Z coordinates are equal, false otherwise.

explicit operator Vector3(Point3D)

Explicitly converts a Point3D to a OpenTK.Mathematics.Vector3.

```
public static explicit operator Vector3(Point3D point)
```

Parameters

`point` Point3D

The Point3D instance to convert.

Returns

Vector3

explicit operator Vector3d(Point3D)

Explicitly converts a Point3D to a OpenTK.Mathematics.Vector3d.

```
public static explicit operator Vector3d(Point3D point)
```

Parameters

`point` Point3D

The Point3D instance to convert.

Returns

Vector3d

operator !=(Point3D, Point3D)

Compares two Point3D instances for inequality.

```
public static bool operator !=(Point3D value, Point3D other)
```

Parameters

`value` Point3D

The first Point3D instance.

`other` Point3D

The second Point3D instance.

Returns

bool

True if the X, Y, and Z coordinates are not equal, false otherwise.

operator *(Point3D, Vector3d)

Multiplies a Point3D by a OpenTK.Mathematics.Vector3.

```
public static Point3D operator *(Point3D point, Vector3d scale)
```

Parameters

`point` Point3D

The Point3D instance.

`scale` Vector3d

The OpenTK.Mathematics.Vector3d to multiply by.

Returns

Point3D

A new Point3D resulting from the multiplication.

operator *(Point3D, float)

Multiplies a Point3D by a scalar value.

```
public static Point3D operator *(Point3D point, float scale)
```

Parameters

`point` Point3D

The Point3D instance.

`scale` float

The scalar value to multiply by.

Returns

Point3D

A new Point3D resulting from the multiplication.

operator -(Point3D, Point3D)

Subtracts one Point3D from another, returning the result as a Vector3D.

```
public static Vector3D operator -(Point3D point1, Point3D point2)
```

Parameters

`point1` Point3D

The first Point3D.

`point2` Point3D

The second Point3D.

Returns

Vector3D

A Vector3D representing the difference.

operator -(Point3D, Vector3D)

Subtracts a Vector3D from a Point3D.

```
public static Point3D operator -(Point3D point, Vector3D direction)
```

Parameters

`point` Point3D

The Point3D position.

`direction` Vector3D

The Vector3D direction.

Returns

Point3D

A new Point3D resulting from the subtraction.

operator -(Point3D)

Negates the coordinates of this Point3D, returning a new Point3D with the opposite values.

```
public static Point3D operator -(Point3D point)
```

Parameters

`point` Point3D

The Point3D instance to negate.

Returns

Point3D

A new Point3D with X, Y, and Z values negated.

Struct Ray

Namespace: rt004.Util

Assembly: rt004.dll

Represents a line parametricly in 3D using Point and Vector

```
public record struct Ray : IEquatable<Ray>
```

Implements
IEquatable<Ray>

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

Ray(Point3D, Vector3D)

```
public Ray(Point3D position, Vector3D direction)
```

Parameters

`position` Point3D

`direction` Vector3D

Fields

Direction

Direction of line represented as vector of length 1

```
public readonly Vector3D Direction
```

Field Value

Vector3D

Origin

origin point of the ray

```
public readonly Point3D Origin
```

Field Value

Point3D

Methods

GetPointOnRay(double)

Computes point on the line by multipling the parameter with normalized directional vector.

```
public Point3D GetPointOnRay(double distance)
```

Parameters

`distance` double

Distance, how far from position in direction of Direction is the point

Returns

Point3D

Point on the line

Struct Vector2D

Namespace: rt004.Util

Assembly: rt004.dll

Represents an immutable 2-dimensional vector.

```
public struct Vector2D
```

Inherited Members
object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

Vector2D(Vector2d)

Initializes a new instance of the Vector2D struct from a OpenTK.Mathematics.Vector2d.

```
public Vector2D(Vector2d vector)
```

Parameters

`vector` Vector2d

The vector to initialize from.

Vector2D(double, double)

Initializes a new instance of the Vector2D struct from the specified X and Y coordinates.

```
public Vector2D(double x, double y)
```

Parameters

`x` double

The X coordinate of the vector.

y double

The Y coordinate of the vector.

Properties

AsVector

Returns the vector's internal representation as a OpenTK.Mathematics.Vector2d.

```
public Vector2d AsVector { get; }
```

Property Value

Vector2d

Length

Gets the length (magnitude) of the vector.

```
public double Length { get; }
```

Property Value

double

LengthSquared

Gets the square of the vector's length (*X**X* + *Y**Y*), avoiding the cost of square root calculation.

```
public double LengthSquared { get; }
```

Property Value

double

X

Gets the X coordinate of the vector.

```
public double X { get; }
```

Property Value

double

Y

Gets the Y coordinate of the vector.

```
public double Y { get; }
```

Property Value

double

Zero

Returns a new instance of a zero vector (0, 0).

```
public static Vector2D Zero { get; }
```

Property Value

Vector2D

Methods

Dot(Vector2D, Vector2D)

Calculates the dot product of two vectors.

```
public static double Dot(Vector2D vector1, Vector2D vector2)
```

Parameters

`vector1` Vector2D

The first vector.

`vector2` Vector2D

The second vector.

Returns

double

The dot product result as a double.

Equals(object?)

Overrides the default Equals method to compare the current instance with another object.

```
public override bool Equals(object? obj)
```

Parameters

`obj` object

The object to compare with the current instance.

Returns

bool

`true` if the object is a Vector2D and its components are equal to those of the current instance; otherwise, `false`.

Equals(Vector2D)

Determines whether this Vector2D is equal to another Vector2D.

```
public bool Equals(Vector2D other)
```

Parameters

`other` Vector2D

The Vector2D to compare with the current instance.

Returns

bool

`true` if the specified Vector2D is equal to the current instance; otherwise, `false`.

GetHashCode()

Returns the hash code for this instance.

```
public override int GetHashCode()
```

Returns

int

An integer representing the hash code.

Normalize()

Normalizes the vector in place, updating its direction to unit length.

```
public void Normalize()
```

Normalized()

Returns a normalized (unit length) version of this vector.

```
public Vector2D Normalized()
```

Returns

Vector2D

A new Vector2D with unit length.

RoundDown()

Rounds down the X and Y coordinates of the vector to the nearest integer.

```
public (int X, int Y) RoundDown()
```

Returns

(int, int)

A tuple containing the rounded X and Y coordinates.

RoundDown(Vector2D)

Returns a new vector with the X and Y coordinates rounded down.

```
public static Vector2D RoundDown(Vector2D point)
```

Parameters

`point` Vector2D

The vector to round down.

Returns

Vector2D

A new Vector2D instance with rounded coordinates.

RoundUp()

Rounds up the X and Y coordinates of the vector to the nearest integer.

```
public (int X, int Y) RoundUp()
```

Returns

(int, int)

A tuple containing the rounded X and Y coordinates.

RoundUp(Vector2D)

Returns a new vector with the X and Y coordinates rounded up.

```
public static Vector2D RoundUp(Vector2D point)
```

Parameters

`point` Vector2D

The vector to round up.

Returns

Vector2D

A new Vector2D instance with rounded coordinates.

ToString()

Returns a string that represents the current vector.

```
public override string ToString()
```

Returns

string

A string in the format "[X, Y]".

Operators

operator +(Vector2D, Point2D)

Adds a Vector2D to a Point2D, resulting in a new Point2D.

```
public static Point2D operator +(Vector2D vector, Point2D point)
```

Parameters

`vector` Vector2D

The Vector2D to add.

`point` Point2D

The Point2D to which the vector is added.

Returns

Point2D

A new Point2D representing the sum of the point and vector.

operator +(Vector2D, Vector2D)

Adds two Vector2D instances.

```
public static Vector2D operator +(Vector2D vector1, Vector2D vector2)
```

Parameters

`vector1` Vector2D

The first vector.

`vector2` Vector2D

The second vector.

Returns

Vector2D

A new Vector2D representing the sum of the two vectors.

operator /(Vector2D, double)

Divides a Vector2D by a scalar value, resulting in a scaled Vector2D.

```
public static Vector2D operator /(Vector2D vector, double scale)
```

Parameters

`vector` Vector2D

The Vector2D to scale down.

`scale` double

The scalar divisor.

Returns

Vector2D

A new Vector2D representing the vector divided by the scalar.

operator /(Vector2D, Vector2D)

Divides a vector by another vector component-wise.

```
public static Vector2D operator /(Vector2D dir, Vector2D scale)
```

Parameters

`dir` Vector2D

The vector to divide.

`scale` Vector2D

The vector to divide by.

Returns

Vector2D

A new Vector2D representing the divided vector.

operator ==(Vector2D, Vector2D)

Determines whether two Vector2D instances are equal by comparing their components.

```
public static bool operator ==(Vector2D left, Vector2D right)
```

Parameters

`left` Vector2D

The first Vector2D instance.

`right` Vector2D

The second Vector2D instance.

Returns

bool

`true` if the components of both vectors are equal; otherwise, `false`.

explicit operator Vector2(Vector2D)

Explicitly converts a Vector2D to a OpenTK.Mathematics.Vector2.

```
public static explicit operator Vector2(Vector2D dir)
```

Parameters

`dir` Vector2D

The Vector2D instance to convert.

Returns

Vector2

explicit operator Vector2d(Vector2D)

Explicitly converts a Vector2D to a OpenTK.Mathematics.Vector2d.

```
public static explicit operator Vector2d(Vector2D dir)
```

Parameters

`dir` Vector2D

The Vector2D instance to convert.

Returns

Vector2d

operator !=(Vector2D, Vector2D)

Determines whether two Vector2D instances are not equal by comparing their components.

```
public static bool operator !=(Vector2D left, Vector2D right)
```

Parameters

`left` Vector2D

The first Vector2D instance.

`right` Vector2D

The second Vector2D instance.

Returns

bool

`true` if the components of both vectors are not equal; otherwise, `false`.

operator *(Vector2D, double)

Multiplies a vector by a scalar.

```
public static Vector2D operator *(Vector2D direction, double scale)
```

Parameters

`direction` Vector2D

The vector to scale.

`scale` double

The scalar multiplier.

Returns

Vector2D

A new Vector2D representing the scaled vector.

operator *(Vector2D, Vector2D)

Multiplies two vectors component-wise.

```
public static Vector2D operator *(Vector2D vector, Vector2D scale)
```

Parameters

`vector` Vector2D

The vector to scale.

`scale` Vector2D

The vector to scale by.

Returns

Vector2D

A new Vector2D representing the scaled vector.

operator -(Vector2D, Point2D)

Subtracts a Point2D from a Vector2D, resulting in a new Point2D.

```
public static Point2D operator -(Vector2D vector, Point2D point)
```

Parameters

`vector` Vector2D

The Vector2D to subtract from.

`point` Point2D

The Point2D to be subtracted.

Returns

Point2D

A new Point2D representing the difference.

operator -(Vector2D, Vector2D)

Subtracts one Vector2D from another.

```
public static Vector2D operator -(Vector2D vector1, Vector2D vector2)
```

Parameters

`vector1` Vector2D

The first vector.

`vector2` Vector2D

The second vector to subtract from the first.

Returns

Vector2D

A new Vector2D representing the difference.

operator -(Vector2D)

Negates the vector, flipping its direction.

```
public static Vector2D operator -(Vector2D vector)
```

Parameters

`vector` Vector2D

The Vector2D instance to negate.

Returns

Vector2D

A new Vector2D pointing in the opposite direction.

Struct Vector3D

Namespace: rt004.Util

Assembly: rt004.dll

Represents an immutable 3-dimensional vector.

```
public struct Vector3D
```

Inherited Members
object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

Vector3D(Vector3d)

Initializes a new instance of the Vector3D struct from a OpenTK.Mathematics.Vector3d.

```
public Vector3D(Vector3d vector)
```

Parameters

`vector` Vector3d

The vector to initialize from.

Vector3D(double, double, double)

Initializes a new instance of the Vector3D struct from the specified X, Y, and Z coordinates.

```
public Vector3D(double x, double y, double z)
```

Parameters

`x` double

The X coordinate of the vector.

y double

The Y coordinate of the vector.

z double

The Z coordinate of the vector.

Properties

AsVector

Returns the vector's internal representation as a OpenTK.Mathematics.Vector3d.

```
public Vector3d AsVector { get; }
```

Property Value

Vector3d

Length

Gets the length (magnitude) of the vector.

```
public double Length { get; }
```

Property Value

double

LengthSquared

Gets the square of the vector's length (X*X* + *YY* + Z*Z), avoiding the cost of a square root calculation.

```
public double LengthSquared { get; }
```

Property Value

double

OneX

Returns a unit vector along the X axis.

```
public static Vector3D OneX { get; }
```

Property Value

Vector3D

OneY

Returns a unit vector along the Y axis.

```
public static Vector3D OneY { get; }
```

Property Value

Vector3D

OneZ

Returns a unit vector along the Z axis.

```
public static Vector3D OneZ { get; }
```

Property Value

Vector3D

X

Gets the X coordinate of the vector.

```
public double X { get; }
```

Property Value

double

Y

Gets the Y coordinate of the vector.

```
public double Y { get; }
```

Property Value

double

Z

Gets the Z coordinate of the vector.

```
public double Z { get; }
```

Property Value

double

Zero

Returns a new instance of a zero vector (0, 0, 0).

```
public static Vector3D Zero { get; }
```

Property Value

Vector3D

Methods

Cross(Vector3D, Vector3D)

Computes the cross product of two vectors.

```
public static Vector3D Cross(Vector3D vector1, Vector3D vector2)
```

Parameters

`vector1` Vector3D

The first vector.

`vector2` Vector3D

The second vector.

Returns

Vector3D

A new Vector3D representing the cross product.

Dot(Vector3D, Vector3D)

Calculates the dot product of two vectors.

```
public static double Dot(Vector3D vector1, Vector3D vector2)
```

Parameters

`vector1` Vector3D

The first vector.

`vector2` Vector3D

The second vector.

Returns

double

The dot product result as a double.

Equals(object?)

Determines whether this instance of Vector3D is equal to a specified object.

```
public override bool Equals(object? obj)
```

Parameters

`obj` object

The object to compare with the current Vector3D.

Returns

bool

`true` if `obj` is a Vector3D and has the same X, Y, and Z values as the current instance; otherwise, `false`.

Equals(Vector3D)

Determines whether this Vector3D is equal to another Vector3D.

```
public bool Equals(Vector3D other)
```

Parameters

`other` Vector3D

The Vector3D to compare with the current instance.

Returns

bool

`true` if the specified Vector3D is equal to the current instance; otherwise, `false`.

GetHashCode()

Returns the hash code for this instance.

```
public override int GetHashCode()
```

Returns

int

An integer representing the hash code.

Normalize()

Normalizes the vector in place, updating its direction to unit length.

```
public void Normalize()
```

Normalized()

Returns a normalized (unit length) version of this vector.

```
public Vector3D Normalized()
```

Returns

Vector3D

A new Vector3D with unit length.

RoundDown()

Rounds down the X, Y, and Z coordinates of the vector to the nearest integer.

```
public (int X, int Y, int Z) RoundDown()
```

Returns

(int, int, int)

A tuple containing the rounded X, Y, and Z coordinates.

RoundDown(Vector3D)

Returns a new vector with the X, Y, and Z coordinates rounded down.

```
public static Vector3D RoundDown(Vector3D point)
```

Parameters

`point` Vector3D

The vector to round down.

Returns

Vector3D

A new Vector3D instance with rounded coordinates.

RoundUp()

Rounds up the X, Y, and Z coordinates of the vector to the nearest integer.

```
public (int X, int Y, int Z) RoundUp()
```

Returns

(int, int, int)

A tuple containing the rounded X, Y, and Z coordinates.

RoundUp(Vector3D)

Returns a new vector with the X, Y, and Z coordinates rounded up.

```
public static Vector3D RoundUp(Vector3D point)
```

Parameters

`point` Vector3D

The vector to round up.

Returns

Vector3D

A new Vector3D instance with rounded coordinates.

ToString()

Returns a string that represents the current vector.

```
public override string ToString()
```

Returns

string

A string in the format "[X, Y, Z]".

Transform(Vector3D, Matrix4d)

Transforms a direction vector by the given matrix, ignoring any translation.

```
public static Vector3D Transform(Vector3D vector, Matrix4d transform)
```

Parameters

`vector` Vector3D

The vector to transform.

`transform` Matrix4d

The transformation matrix.

Returns

Vector3D

A new Vector3D resulting from the transformation.

Transform(Vector3D, Quaternion)

Transforms a vector by applying the specified quaternion rotation.

```
public static Vector3D Transform(Vector3D vector, Quaternion rotation)
```

Parameters

`vector` Vector3D

The vector to rotate.

`rotation` Quaternion

The rotation quaternion.

Returns

Vector3D

A new Vector3D resulting from the rotation.

Operators

operator +(Vector3D, Point3D)

Adds a Vector3D and a Point3D.

```
public static Point3D operator +(Vector3D vector, Point3D point)
```

Parameters

`vector` Vector3D

The vector to add.

`point` Point3D

The point to add to the vector.

Returns

Point3D

A new Point3D representing the result of the addition.

operator +(Vector3D, Vector3D)

Adds two Vector3D instances.

```
public static Vector3D operator +(Vector3D vector1, Vector3D vector2)
```

Parameters

`vector1` Vector3D

The first vector.

`vector2` Vector3D

The second vector.

Returns

Vector3D

A new Vector3D representing the sum of the two vectors.

operator /(Vector3D, Vector3d)

Divides a vector by another vector component-wise.

```
public static Vector3D operator /(Vector3D vector, Vector3d scale)
```

Parameters

`vector` Vector3D

The vector to divide.

`scale` Vector3d

The vector to divide by.

Returns

Vector3D

A new Vector3D representing the divided vector.

operator /(Vector3D, double)

Divides a vector by a scalar.

```
public static Vector3D operator /(Vector3D dir, double scale)
```

Parameters

`dir` Vector3D

The vector to divide.

`scale` double

The scalar to divide by.

Returns

Vector3D

A new Vector3D representing the scaled vector.

operator ==(Vector3D, Vector3D)

Determines whether two Vector3D instances are equal by comparing their components.

```
public static bool operator ==(Vector3D left, Vector3D right)
```

Parameters

`left` Vector3D

The first Vector3D instance.

`right` Vector3D

The second Vector3D instance.

Returns

bool

`true` if the components of both vectors are equal; otherwise, `false`.

explicit operator Vector3(Vector3D)

Explicitly converts a Vector3D to a OpenTK.Mathematics.Vector3.

```
public static explicit operator Vector3(Vector3D dir)
```

Parameters

`dir` Vector3D

The Vector3D instance to convert.

Returns

Vector3

explicit operator Vector3d(Vector3D)

Explicitly converts a Vector3D to a OpenTK.Mathematics.Vector3d.

```
public static explicit operator Vector3d(Vector3D dir)
```

Parameters

`dir` Vector3D

The Vector3D instance to convert.

Returns

Vector3d

operator !=(Vector3D, Vector3D)

Determines whether two Vector3D instances are not equal by comparing their components.

```
public static bool operator !=(Vector3D left, Vector3D right)
```

Parameters

`left` Vector3D

The first Vector3D instance.

`right` Vector3D

The second Vector3D instance.

Returns

bool

`true` if the components of both vectors are not equal; otherwise, `false`.

operator *(Vector3D, Vector3d)

Multiplies a Vector3D by a OpenTK.Mathematics.Vector3d component-wise.

```
public static Vector3D operator *(Vector3D vector, Vector3d scale)
```

Parameters

`vector` Vector3D

The Vector3D to multiply.

`scale` Vector3d

The OpenTK.Mathematics.Vector3d to multiply by.

Returns

Vector3D

A new Vector3D representing the result of the component-wise multiplication.

operator *(Vector3D, double)

Multiplies a vector by a scalar.

```
public static Vector3D operator *(Vector3D vector, double scale)
```

Parameters

`vector` Vector3D

The vector to scale.

`scale` double

The scalar multiplier.

Returns

Vector3D

A new Vector3D representing the scaled vector.

operator *(Vector3D, Vector3D)

Multiplies two vectors component-wise.

```
public static Vector3D operator *(Vector3D vector, Vector3D scale)
```

Parameters

`vector` Vector3D

The vector to scale.

`scale` Vector3D

The vector to scale by.

Returns

Vector3D

A new Vector3D representing the scaled vector.

operator -(Vector3D, Point3D)

Subtracts a Vector3D from a Point3D.

```
public static Point3D operator -(Vector3D vector, Point3D point)
```

Parameters

`vector` Vector3D

The vector to subtract.

`point` Point3D

The point from which to subtract the vector.

Returns

Point3D

A new Point3D representing the result of the subtraction.

operator -(Vector3D, Vector3D)

Subtracts one Vector3D from another.

```
public static Vector3D operator -(Vector3D vector1, Vector3D vector2)
```

Parameters

`vector1` Vector3D

The first vector.

`vector2` Vector3D

The second vector to subtract from the first.

Returns

Vector3D

A new Vector3D representing the difference.

operator -(Vector3D)

Negates the vector, flipping its direction.

```
public static Vector3D operator -(Vector3D vector)
```

Parameters

`vector` Vector3D

The Vector3D instance to negate.

Returns

Vector3D

A new Vector3D pointing in the opposite direction.

Namespace rt004.Util.LightModels

Classes

LightModelComputation

Abstract class for light model computation.

Class LightModelComputation

Namespace: rt004.Util.LightModels

Assembly: rt004.dll

Abstract class for light model computation.

```
public abstract class LightModelComputation
```

Inheritance

object ← LightModelComputation

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

ComputeLightColor(IntersectionProperties, Color4, LightSource[])

Computes color at the intersection

```
public abstract Color4 ComputeLightColor(IntersectionProperties intersection, Color4
backgroundColor, LightSource[] lights)
```

Parameters

`intersection` IntersectionProperties

Intersection properties of ray - scene intersection

`backgroundColor` Color4

background color of the scene

`lights` LightSource[]

light sources to compute lighting with

Returns

Color4

Returns color at the intersection position

GetCorrectLightModel(LightModel)

Gets appropriate light computation model based on enum

```
public static LightModelComputation GetCorrectLightModel(LightModel lightModel)
```

Parameters

`lightModel` LightModel

Returns

LightModelComputation

Returns LightModelComputation appropriate for lightModel

Namespace rt004.UtilLoading

Classes

RendererSettingsLoader

Class RendererSettingsLoader

Namespace: rt004.UtilLoading

Assembly: rt004.dll

```
public class RendererSettingsLoader
```

Inheritance

object ← RendererSettingsLoader

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Fields

defaultAmbientLightColor

```
public Color4 defaultAmbientLightColor
```

Field Value

Color4

defaultAmbientLightFactor

```
public float defaultAmbientLightFactor
```

Field Value

float

defaultBackgroundColor

```
public Color4 defaultBackgroundColor
```

Field Value

Color4

defaultCameraHeight

```
public static uint defaultCameraHeight
```

Field Value

uint

defaultCameraWidth

```
public static uint defaultCameraWidth
```

Field Value

uint

defaultDiffuseFactor

```
public float defaultDiffuseFactor
```

Field Value

float

defaultIndexOfRefraction

```
public float defaultIndexOfRefraction
```

Field Value

float

defaultRoughnessFactor

```
public float defaultRoughnessFactor
```

Field Value

float

defaultShininessFactor

```
public float defaultShininessFactor
```

Field Value

float

defaultSolidColor

```
public Color4 defaultSolidColor
```

Field Value

Color4

defaultSpecularFactor

```
public float defaultSpecularFactor
```

Field Value

float

defaultTransparencyFactor

```
public float defaultTransparencyFactor
```

Field Value

float

epsilon

```
public float epsilon
```

Field Value

float

lightModel

```
public string lightModel
```

Field Value

string

maxReflectionDepth

```
public uint maxReflectionDepth
```

Field Value

uint

minRayContribution

```
public float minRayContribution
```

Field Value

float

reflections

```
public bool reflections
```

Field Value

bool

refractions

```
public bool refractions
```

Field Value

bool

shadows

```
public bool shadows
```

Field Value

bool

Methods

SaveLoadedSettings()

```
public void SaveLoadedSettings()
```

Namespace rt004_tests

Classes

Tests

Class Tests

Namespace: rt004_tests

Assembly: rt004_tests.dll

```
[TestFixture]
public class Tests
```

Inheritance

object ← Tests

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

GlobalPositionTest()

```
[Test]
[Description("tests Global position computation")]
public void GlobalPositionTest()
```

Setup()

```
[SetUp]
public void Setup()
```