

# Namespace Engine

## Classes

[Main](#)

# Class Main

Namespace: [Engine](#)

Assembly: Engine 0.1-a.1.dll

```
public class Main : Form, IDropTarget, ISynchronizeInvoke, IWin32Window, IBindableComponent,
IComponent, IDisposable,.IContainerControl
```

## Inheritance

```
object ↳ ← MarshalByRefObject ↳ ← Component ↳ ← Control ↳ ← ScrollableControl ↳ ←
ContainerControl ↳ ← Form ↳ ← Main
```

## Implements

```
IDropTarget ↳ , ISynchronizeInvoke ↳ , IWin32Window ↳ , IBindableComponent ↳ , IComponent ↳ ,
IDisposable ↳ , IContainerControl ↳
```

## Inherited Members

```
Form.SetVisibleCore(bool) ↳ , Form.Activate() ↳ , Form.ActivateMdiChild(Form) ↳ ,
Form.AddOwnedForm(Form) ↳ , Form.AdjustFormScrollbars(bool) ↳ , Form.Close() ↳ ,
Form.CreateControlsInstance() ↳ , Form.CreateHandle() ↳ , Form.DefWndProc(ref Message) ↳ ,
Form.ProcessMnemonic(char) ↳ , Form.CenterToParent() ↳ , Form.CenterToScreen() ↳ ,
Form.LayoutMdi(MdiLayout) ↳ , Form.OnActivated(EventArgs) ↳ ,
Form.OnBackgroundImageChanged(EventArgs) ↳ ,
Form.OnBackgroundImageLayoutChanged(EventArgs) ↳ , Form.OnClosing(CancelEventArgs) ↳ ,
Form.OnClosed(EventArgs) ↳ , Form.OnFormClosing(FormClosingEventArgs) ↳ ,
Form.OnFormClosed(FormClosedEventArgs) ↳ , Form.OnCreateControl() ↳ ,
Form.OnDeactivate(EventArgs) ↳ , Form.OnEnabledChanged(EventArgs) ↳ , Form.OnEnter(EventArgs) ↳ ,
Form.OnFontChanged(EventArgs) ↳ , Form.OnHandleCreated(EventArgs) ↳ ,
Form.OnHandleDestroyed(EventArgs) ↳ , Form.OnHelpButtonClicked(CancelEventArgs) ↳ ,
Form.OnLayout(LayoutEventArgs) ↳ , Form.OnLoad(EventArgs) ↳ ,
Form.OnMaximizedBoundsChanged(EventArgs) ↳ , Form.OnMaximumSizeChanged(EventArgs) ↳ ,
Form.OnMinimumSizeChanged(EventArgs) ↳ ,
Form.OnInputLanguageChanged(InputLanguageChangedEventArgs) ↳ ,
Form.OnInputLanguageChanging(InputLanguageChangingEventArgs) ↳ ,
Form.OnVisibleChanged(EventArgs) ↳ , Form.OnMdiChildActivate(EventArgs) ↳ ,
Form.OnMenuStart(EventArgs) ↳ , Form.OnMenuComplete(EventArgs) ↳ ,
Form.OnPaint(PaintEventArgs) ↳ , Form.OnResize(EventArgs) ↳ ,
Form.OnDpiChanged(DpiChangedEventArgs) ↳ , Form.OnGetDpiScaledSize(int, int, ref Size) ↳ ,
Form.OnRightToLeftLayoutChanged(EventArgs) ↳ , Form.OnShown(EventArgs) ↳ ,
```

[Form.OnTextChanged\(EventArgs\)](#) , [Form.ProcessCmdKey\(ref Message, Keys\)](#) ,  
[Form.ProcessDialogKey\(Keys\)](#) , [Form.ProcessDialogChar\(char\)](#) ,  
[Form.ProcessKeyPreview\(ref Message\)](#) , [Form.ProcessTabKey\(bool\)](#) ,  
[Form.RemoveOwnedForm\(Form\)](#) , [Form.Select\(bool, bool\)](#) ,  
[Form.GetScaledBounds\(Rectangle, SizeF, BoundsSpecified\)](#) ,  
[Form.ScaleControl\(SizeF, BoundsSpecified\)](#) , [Form.SetBoundsCore\(int, int, int, int, BoundsSpecified\)](#) ,  
[Form.SetClientSizeCore\(int, int\)](#) , [Form.SetDesktopBounds\(int, int, int, int\)](#) ,  
[Form.SetDesktopLocation\(int, int\)](#) , [Form.Show\(IWin32Window\)](#) , [Form.ShowDialog\(\)](#) ,  
[Form.ShowDialog\(IWin32Window\)](#) , [Form.ToString\(\)](#) , [Form.UpdateDefaultButton\(\)](#) ,  
[Form.OnResizeBegin\(EventArgs\)](#) , [Form.OnResizeEnd\(EventArgs\)](#) ,  
[Form.OnStyleChanged\(EventArgs\)](#) , [Form.ValidateChildren\(\)](#) ,  
[Form.ValidateChildren\(ValidationConstraints\)](#) , [Form.WndProc\(ref Message\)](#) , [Form.AcceptButton](#) ,  
[Form.ActiveForm](#) , [Form.ActiveMdiChild](#) , [Form.AllowTransparency](#) , [Form.AutoScroll](#) ,  
[Form.AutoSize](#) , [Form.AutoSizeMode](#) , [Form.AutoValidate](#) , [Form.BackColor](#) ,  
[Form.FormBorderStyle](#) , [Form.CancelButton](#) , [Form.ClientSize](#) , [Form.ControlBox](#) ,  
[Form.CreateParams](#) , [Form.DefaultImeMode](#) , [Form.DefaultSize](#) , [Form.DesktopBounds](#) ,  
[Form/DesktopLocation](#) , [Form/DialogResult](#) , [Form/HelpButton](#) , [Form/Icon](#) , [Form/IsMdiChild](#) ,  
[Form/IsMdiContainer](#) , [Form/IsRestrictedWindow](#) , [Form/KeyPreview](#) , [Form/Location](#) ,  
[Form/MaximizedBounds](#) , [Form/MaximumSize](#) , [Form/MainMenuStrip](#) , [Form/Menu](#) ,  
[Form/MinimumSize](#) , [Form/MaximizeBox](#) , [Form/MdiChildren](#) , [Form/MdiParent](#) ,  
[Form/MergedMenu](#) , [Form/MinimizeBox](#) , [Form/Modal](#) , [Form/Opacity](#) , [Form/OwnedForms](#) ,  
[Form/Owner](#) , [Form/RestoreBounds](#) , [Form/RightToLeftLayout](#) , [Form>ShowInTaskbar](#) ,  
[Form>ShowIcon](#) , [Form>ShowWithoutActivation](#) , [Form/Size](#) , [Form/SizeGripStyle](#) ,  
[Form/StartPosition](#) , [Form/Text](#) , [Form/TopLevel](#) , [Form/TopMost](#) , [Form/TransparencyKey](#) ,  
[Form/WindowState](#) , [Form/AutoSizeChanged](#) , [Form/AutoValidateChanged](#) ,  
[Form/HelpButtonClicked](#) , [Form/MaximizedBoundsChanged](#) , [Form/MaximumSizeChanged](#) ,  
[Form/MinimumSizeChanged](#) , [Form/Activated](#) , [Form/Deactivate](#) , [Form/FormClosing](#) ,  
[Form/FormClosed](#) , [Form/Load](#) , [Form/MdiChildActivate](#) , [Form/MenuComplete](#) ,  
[Form/MenuStart](#) , [Form/InputLanguageChanged](#) , [Form/InputLanguageChanging](#) ,  
[Form/RightToLeftLayoutChanged](#) , [Form/Shown](#) , [Form/DpiChanged](#) , [Form/ResizeBegin](#) ,  
[Form/ResizeEnd](#) , [ContainerControl.OnAutoValidateChanged\(EventArgs\)](#) ,  
[ContainerControl.OnParentChanged\(EventArgs\)](#) , [ContainerControl.PerformLayout\(\)](#) ,  
[ContainerControl.Validate\(\)](#) , [ContainerControl.Validate\(bool\)](#) ,  
[ContainerControl.AutoScaleDimensions](#) , [ContainerControl.AutoScaleFactor](#) ,  
[ContainerControl.AutoScaleMode](#) , [ContainerControl.BindingContext](#) ,  
[ContainerControl.CanEnableIme](#) , [ContainerControl.ActiveControl](#) ,  
[ContainerControl.CurrentAutoScaleDimensions](#) , [ContainerControl.ParentForm](#) ,  
[ScrollableControl.ScrollStateAutoScrolling](#) , [ScrollableControl.ScrollStateHScrollVisible](#) ,  
[ScrollableControl.ScrollStateVScrollVisible](#) , [ScrollableControl.ScrollStateUserHasScrolled](#) ,  
[ScrollableControl.ScrollStateFullDrag](#) , [ScrollableControl.GetScrollState\(int\)](#) ,

[ScrollableControl.OnMouseWheel\(MouseEventArgs\)](#) ,  
[ScrollableControl.OnRightToLeftChanged\(EventArgs\)](#) ,  
[ScrollableControl.OnPaintBackground\(PaintEventArgs\)](#) ,  
[ScrollableControl.OnPaddingChanged\(EventArgs\)](#) , [ScrollableControl.SetDisplayRectLocation\(int, int\)](#) ,  
[ScrollableControl.ScrollControlIntoView\(Control\)](#) , [ScrollableControl.ScrollToControl\(Control\)](#) ,  
[ScrollableControl.OnScroll\(ScrollEventArgs\)](#) , [ScrollableControl.SetAutoScrollMargin\(int, int\)](#) ,  
[ScrollableControl.SetScrollState\(int, bool\)](#) , [ScrollableControl.AutoScrollMargin](#) ,  
[ScrollableControl.AutoScrollPosition](#) , [ScrollableControl.AutoScrollMinSize](#) ,  
[ScrollableControl.DisplayRectangle](#) , [ScrollableControl.HScroll](#) , [ScrollableControl.HorizontalScroll](#) ,  
[ScrollableControl.VScroll](#) , [ScrollableControl.VerticalScroll](#) , [ScrollableControl.Scroll](#) ,  
[Control.GetAccessibilityObjectById\(int\)](#) , [Control.SetAutoSizeMode\(AutoSizeMode\)](#) ,  
[Control.GetAutoSizeMode\(\)](#) , [Control.GetPreferredSize\(Size\)](#) ,  
[Control.AccessibilityNotifyClients\(AccessibleEvents, int\)](#) ,  
[Control.AccessibilityNotifyClients\(AccessibleEvents, int, int\)](#) , [Control.BeginInvoke\(Delegate\)](#) ,  
[Control.BeginInvoke\(Delegate, params object\[\]\)](#) , [Control.BringToFront\(\)](#) ,  
[Control.Contains\(Control\)](#) , [Control.CreateAccessibilityInstance\(\)](#) , [Control.CreateGraphics\(\)](#) ,  
[Control.CreateControl\(\)](#) , [Control.DestroyHandle\(\)](#) , [Control.DoDragDrop\(object, DragDropEffects\)](#) ,  
[Control.DrawToBitmap\(Bitmap, Rectangle\)](#) , [Control.EndInvoke\(IAsyncResult\)](#) , [Control.FindForm\(\)](#) ,  
[Control.GetTopLevel\(\)](#) , [Control.RaiseKeyEvent\(object, KeyEventArgs\)](#) ,  
[Control.RaiseMouseEvent\(object, MouseEventArgs\)](#) , [Control.Focus\(\)](#) ,  
[Control.FromChildHandle\(IntPtr\)](#) , [Control.FromHandle\(IntPtr\)](#) ,  
[Control.GetChildAtPoint\(Point, GetChildAtPointSkip\)](#) , [Control.GetChildAtPoint\(Point\)](#) ,  
[Control.GetContainerControl\(\)](#) , [Control.GetNextControl\(Control, bool\)](#) ,  
[Control.GetStyle\(ControlStyles\)](#) , [Control.Hide\(\)](#) , [Control.InitLayout\(\)](#) , [Control.Invalidate\(Region\)](#) ,  
[Control.Invalidate\(Region, bool\)](#) , [Control.Invalidate\(\)](#) , [Control.Invalidate\(bool\)](#) ,  
[Control.Invalidate\(Rectangle\)](#) , [Control.Invalidate\(Rectangle, bool\)](#) , [Control.Invoke\(Delegate\)](#) ,  
[Control.Invoke\(Delegate, params object\[\]\)](#) , [Control.InvokePaint\(Control, PaintEventArgs\)](#) ,  
[Control.InvokePaintBackground\(Control, PaintEventArgs\)](#) , [Control.IsKeyLocked\(Keys\)](#) ,  
[Control.IsInputChar\(char\)](#) , [Control.IsInputKey\(Keys\)](#) , [Control.IsMnemonic\(char, string\)](#) ,  
[Control.LogicalToDeviceUnits\(int\)](#) , [Control.ScaleBitmapLogicalToDevice\(ref Bitmap\)](#) ,  
[Control.NotifyInvalidate\(Rectangle\)](#) , [Control.InvokeOnClick\(Control, EventArgs\)](#) ,  
[Control.OnAutoSizeChanged\(EventArgs\)](#) , [Control.OnBackColorChanged\(EventArgs\)](#) ,  
[Control.OnBindingContextChanged\(EventArgs\)](#) , [Control.OnCausesValidationChanged\(EventArgs\)](#) ,  
[Control.OnContextMenuChanged\(EventArgs\)](#) , [Control.OnContextMenuStripChanged\(EventArgs\)](#) ,  
[Control.OnCursorChanged\(EventArgs\)](#) , [Control.OnDockChanged\(EventArgs\)](#) ,  
[Control.OnForeColorChanged\(EventArgs\)](#) , [Control.OnNotifyMessage\(Message\)](#) ,  
[Control.OnParentBackColorChanged\(EventArgs\)](#) ,  
[Control.OnParentBackgroundImageChanged\(EventArgs\)](#) ,  
[Control.OnParentBindingContextChanged\(EventArgs\)](#) , [Control.OnParentCursorChanged\(EventArgs\)](#) ,  
[Control.OnParentEnabledChanged\(EventArgs\)](#) , [Control.OnParentFontChanged\(EventArgs\)](#) ,

[Control.OnParentForeColorChanged\(EventArgs\)](#) , [Control.OnParentRightToLeftChanged\(EventArgs\)](#) ,  
[Control.OnParentVisibleChanged\(EventArgs\)](#) , [Control.OnPrint\(PaintEventArgs\)](#) ,  
[Control.OnTabIndexChanged\(EventArgs\)](#) , [Control.OnTabStopChanged\(EventArgs\)](#) ,  
[Control.OnClick\(EventArgs\)](#) , [Control.OnClientSizeChanged\(EventArgs\)](#) ,  
[Control.OnControlAdded\(ControlEventArgs\)](#) , [Control.OnControlRemoved\(ControlEventArgs\)](#) ,  
[Control.OnLocationChanged\(EventArgs\)](#) , [Control.OnDoubleClick\(EventArgs\)](#) ,  
[Control.OnDragEnter\(DragEventArgs\)](#) , [Control.OnDragOver\(DragEventArgs\)](#) ,  
[Control.OnDragLeave\(EventArgs\)](#) , [Control.OnDragDrop\(DragEventArgs\)](#) ,  
[Control.OnGiveFeedback\(GiveFeedbackEventArgs\)](#) , [Control.InvokeGotFocus\(Control, EventArgs\)](#) ,  
[Control.OnGotFocus\(EventArgs\)](#) , [Control.OnHelpRequested\(EventArgs\)](#) ,  
[Control.OnInvalidate\(EventArgs\)](#) , [Control.OnKeyDown\(KeyEventArgs\)](#) ,  
[Control.OnKeyPress\(KeyEventArgs\)](#) , [Control.OnKeyUp\(KeyEventArgs\)](#) ,  
[Control.OnLeave\(EventArgs\)](#) , [Control.InvokeLostFocus\(Control, EventArgs\)](#) ,  
[Control.OnLostFocus\(EventArgs\)](#) , [Control.OnMarginChanged\(EventArgs\)](#) ,  
[Control.OnMouseDoubleClick\(MouseEventArgs\)](#) , [Control.OnMouseClick\(MouseEventArgs\)](#) ,  
[Control.OnMouseCaptureChanged\(EventArgs\)](#) , [Control.OnMouseDown\(MouseEventArgs\)](#) ,  
[Control.OnMouseEnter\(EventArgs\)](#) , [Control.OnMouseLeave\(EventArgs\)](#) ,  
[Control.OnDpiChangedBeforeParent\(EventArgs\)](#) , [Control.OnDpiChangedAfterParent\(EventArgs\)](#) ,  
[Control.OnMouseHover\(EventArgs\)](#) , [Control.OnMouseMove\(MouseEventArgs\)](#) ,  
[Control.OnMouseUp\(MouseEventArgs\)](#) , [Control.OnMove\(EventArgs\)](#) ,  
[Control.OnQueryContinueDrag\(QueryContinueEventArgs\)](#) ,  
[Control.OnRegionChanged\(EventArgs\)](#) , [Control.OnPreviewKeyDown\(PreviewKeyDownEventArgs\)](#) ,  
[Control.OnSizeChanged\(EventArgs\)](#) , [Control.OnChangeUICues\(UICuesEventArgs\)](#) ,  
[Control.OnSystemColorsChanged\(EventArgs\)](#) , [Control.OnValidating\(CancelEventArgs\)](#) ,  
[Control.OnValidated\(EventArgs\)](#) , [Control.RescaleConstantsForDpi\(int, int\)](#) ,  
[Control.PerformLayout\(\)](#) , [Control.PerformLayout\(Control, string\)](#) , [Control.PointToClient\(Point\)](#) ,  
[Control.PointToScreen\(Point\)](#) , [Control.PreProcessMessage\(ref Message\)](#) ,  
[Control.PreProcessControlMessage\(ref Message\)](#) , [Control.ProcessKeyEventArgs\(ref Message\)](#) ,  
[Control.ProcessKeyMessage\(ref Message\)](#) , [Control.RaiseDragEvent\(object, DragEventArgs\)](#) ,  
[Control.RaisePaintEvent\(object, PaintEventArgs\)](#) , [Control.RecreateHandle\(\)](#) ,  
[Control.RectangleToClient\(Rectangle\)](#) , [Control.RectangleToScreen\(Rectangle\)](#) ,  
[Control.ReflectMessage\(IntPtr, ref Message\)](#) , [Control.Refresh\(\)](#) , [Control.ResetMouseEventArgs\(\)](#) ,  
[Control.ResetText\(\)](#) , [Control.ResumeLayout\(\)](#) , [Control.ResumeLayout\(bool\)](#) , [Control.Scale\(SizeF\)](#) ,  
[Control.Select\(\)](#) , [Control.SelectNextControl\(Control, bool, bool, bool, bool\)](#) , [Control.SendToBack\(\)](#) ,  
[Control.SetBounds\(int, int, int, int\)](#) , [Control.SetBounds\(int, int, int, int, BoundsSpecified\)](#) ,  
[Control.SizeFromClientSize\(Size\)](#) , [Control.SetStyle\(ControlStyles, bool\)](#) , [Control.SetTopLevel\(bool\)](#) ,  
[Control.RtlTranslateAlignment\(HorizontalAlignment\)](#) ,  
[Control.RtlTranslateAlignment\(LeftRightAlignment\)](#) ,  
[Control.RtlTranslateAlignment\(ContentAlignment\)](#) ,  
[Control.RtlTranslateHorizontal\(HorizontalAlignment\)](#) ,

[Control.RtlTranslateLeftRight\(LeftRightAlignment\)](#) , [Control.RtlTranslateContent\(ContentAlignment\)](#) ,  
[Control.Show\(\)](#) , [Control.SuspendLayout\(\)](#) , [Control.Update\(\)](#) , [Control.UpdateBounds\(\)](#) ,  
[Control.UpdateBounds\(int, int, int, int\)](#) , [Control.UpdateBounds\(int, int, int, int, int, int\)](#) ,  
[Control.UpdateZOrder\(\)](#) , [Control.UpdateStyles\(\)](#) , [Control.OnImeModeChanged\(EventArgs\)](#) ,  
[Control.AccessibilityObject](#) , [Control.AccessibleDefaultActionDescription](#) ,  
[Control.AccessibleDescription](#) , [Control.AccessibleName](#) , [Control.AccessibleRole](#) ,  
[Control.AllowDrop](#) , [Control.Anchor](#) , [Control.AutoScrollOffset](#) , [Control.LayoutEngine](#) ,  
[Control.BackgroundImage](#) , [Control.BackgroundImageLayout](#) , [Control.Bottom](#) , [Control.Bounds](#) ,  
[Control.CanFocus](#) , [Control.CanRaiseEvents](#) , [Control.CanSelect](#) , [Control.Capture](#) ,  
[Control.CausesValidation](#) , [Control.CheckForIllegalCrossThreadCalls](#) , [Control.ClientRectangle](#) ,  
[Control.CompanyName](#) , [Control.ContainsFocus](#) , [Control.ContextMenu](#) ,  
[Control.ContextMenuStrip](#) , [Control.Controls](#) , [Control.Created](#) , [Control.Cursor](#) ,  
[Control.DataBindings](#) , [Control.DefaultBackColor](#) , [Control.DefaultCursor](#) , [Control.DefaultFont](#) ,  
[Control.DefaultForeColor](#) , [Control.DefaultMargin](#) , [Control.DefaultMaximumSize](#) ,  
[Control.DefaultMinimumSize](#) , [Control.DefaultPadding](#) , [Control.DeviceDpi](#) , [Control.IsDisposed](#) ,  
[Control.Disposing](#) , [Control.Dock](#) , [Control.DoubleBuffered](#) , [Control.Enabled](#) , [Control.Focused](#) ,  
[Control.Font](#) , [Control.FontHeight](#) , [Control.ForeColor](#) , [Control.Handle](#) , [Control.HasChildren](#) ,  
[Control.Height](#) , [Control.IsHandleCreated](#) , [Control.InvokeRequired](#) , [Control.Accessible](#) ,  
[Control.IsMirrored](#) , [Control.Left](#) , [Control.Margin](#) , [Control.ModifierKeys](#) ,  
[Control.MouseButtons](#) , [Control.mousePosition](#) , [Control.Name](#) , [Control.Parent](#) ,  
[Control.ProductName](#) , [Control.ProductVersion](#) , [Control.RecreatingHandle](#) , [Control.Region](#) ,  
[Control.RenderRightToLeft](#) , [Control.ResizeRedraw](#) , [Control.Right](#) , [Control.RightToLeft](#) ,  
[Control.ScaleChildren](#) , [Control.Site](#) , [Control.TabIndex](#) , [Control.TabStop](#) , [Control.Tag](#) ,  
[Control.Top](#) , [Control.TopLevelControl](#) , [Control.ShowKeyboardCues](#) , [Control.ShowFocusCues](#) ,  
[Control.UseWaitCursor](#) , [Control.Visible](#) , [Control.Width](#) , [Control.PreferredSize](#) ,  
[Control.Padding](#) , [Control.ImeMode](#) , [Control.ImeModeBase](#) , [Control.PropagatingImeMode](#) ,  
[Control.BackColorChanged](#) , [Control.BackgroundImageChanged](#) ,  
[Control.BackgroundImageLayoutChanged](#) , [Control.BindingContextChanged](#) ,  
[Control.CausesValidationChanged](#) , [Control.ClientSizeChanged](#) , [Control.ContextMenuChanged](#) ,  
[Control.ContextMenuStripChanged](#) , [Control.CursorChanged](#) , [Control.DockChanged](#) ,  
[Control.EnabledChanged](#) , [Control.FontChanged](#) , [Control.ForeColorChanged](#) ,  
[Control.LocationChanged](#) , [Control.MarginChanged](#) , [Control.RegionChanged](#) ,  
[Control.RightToLeftChanged](#) , [Control.SizeChanged](#) , [Control.TabIndexChanged](#) ,  
[Control.TabStopChanged](#) , [Control.TextChanged](#) , [Control.VisibleChanged](#) , [Control.Click](#) ,  
[Control.ControlAdded](#) , [Control.ControlRemoved](#) , [Control.DragDrop](#) , [Control.DragEnter](#) ,  
[Control.DragOver](#) , [Control.DragLeave](#) , [Control.GiveFeedback](#) , [Control.HandleCreated](#) ,  
[Control.HandleDestroyed](#) , [Control.HelpRequested](#) , [Control.Invalidate](#) ,  
[Control.PaddingChanged](#) , [Control.Paint](#) , [Control.QueryContinueDrag](#) ,  
[Control.QueryAccessibilityHelp](#) , [Control.DoubleClick](#) , [Control.Enter](#) , [Control.GotFocus](#) ,  
[Control.KeyDown](#) , [Control.KeyPress](#) , [Control.KeyUp](#) , [Control.Layout](#) , [Control.Leave](#) ,

[Control.LostFocus](#) , [Control.MouseClick](#) , [Control.MouseDoubleClick](#) ,  
[Control.MouseCaptureChanged](#) , [Control.MouseDown](#) , [Control.MouseEnter](#) ,  
[Control.MouseLeave](#) , [Control.DpiChangedBeforeParent](#) , [Control.DpiChangedAfterParent](#) ,  
[Control.MouseHover](#) , [Control.MouseMove](#) , [Control.MouseUp](#) , [Control.MouseWheel](#) ,  
[Control.Move](#) , [Control.PreviewKeyDown](#) , [Control.Resize](#) , [Control.ChangeUICues](#) ,  
[Control.StyleChanged](#) , [Control.SystemColorsChanged](#) , [Control.Validating](#) , [Control.Validated](#) ,  
[Control.ParentChanged](#) , [Control.ImeModeChanged](#) , [Component.Dispose\(\)](#) ,  
[Component.GetService\(Type\)](#) , [Component.Events](#) , [Component.Container](#) ,  
[Component.DesignMode](#) , [Component.Disposed](#) , [MarshalByRefObject.MemberwiseClone\(bool\)](#) ,  
[MarshalByRefObject.GetLifetimeService\(\)](#) , [MarshalByRefObject.InitializeLifetimeService\(\)](#) ,  
[MarshalByRefObject.CreateObjRef\(Type\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### Main()

```
public Main()
```

## Methods

### Dispose(bool)

Verwendete Ressourcen bereinigen.

```
protected override void Dispose(bool disposing)
```

#### Parameters

**disposing** [bool](#)

True, wenn verwaltete Ressourcen gelöscht werden sollen; andernfalls False.

# Namespace Engine.Core

## Classes

[Component](#)

[LCS3Element](#)

[LCS3ElementGroup](#)

[Layer](#)

[Object3D](#)

[Particle](#)

[Ray](#)

[Tag](#)

## Interfaces

[IParticleHandler](#)

[IRay](#)

# Class Component

Namespace: [Engine.Core](#)

Assembly: Engine.dll

```
public class Component
```

## Inheritance

[object](#) ← Component

## Derived

[Camera](#), [Edge3D](#), [Face3D](#), [LCS3](#), [Mesh3D](#), [Vertex3D](#)

## Inherited Members

[object.ToString\(\)](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),  
[object.MemberwiseClone\(\)](#)

## Constructors

Component(Object3D)

```
public Component(Object3D dependency)
```

## Parameters

dependency [Object3D](#)

## Properties

Dependency

```
public Object3D Dependency { get; set; }
```

Property Value

## Object3D

# Interface IParticleHandler

Namespace: [Engine.Core](#)

Assembly: Engine.dll

```
public interface IParticleHandler
```

## Methods

### EliminateAllParticles()

```
void EliminateAllParticles()
```

### EliminateParticle(ref Particle)

```
void EliminateParticle(ref Particle particle)
```

## Parameters

particle [Particle](#)

# Interface IRay

Namespace: [Engine.Core](#)

Assembly: Engine.dll

```
public interface IRay
```

## Methods

OnRayCollision(List<ParticleCollision3D>)

```
void OnRayCollision(List<ParticleCollision3D> collisions)
```

Parameters

collisions [List](#)<ParticleCollision3D>

# Class LCS3Element

Namespace: [Engine.Core](#)

Assembly: Engine.dll

```
public class LCS3Element
```

## Inheritance

[object](#) ← LCS3Element

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

LCS3Element(int, LCS3, LCS3, Guid)

```
public LCS3Element(int depth, LCS3 lcs, LCS3 parent, Guid groupID)
```

## Parameters

depth [int](#)

lcs [LCS3](#)

parent [LCS3](#)

groupID [Guid](#)

## Fields

GroupID

```
public Guid GroupID
```

Field Value

[Guid ↗](#)

## Properties

Depth

```
public int Depth { get; }
```

Property Value

[int ↗](#)

Lcs

```
public LCS3 Lcs { get; }
```

Property Value

[LCS3](#)

Parent

```
public LCS3 Parent { get; }
```

Property Value

[LCS3](#)

## Operators

implicit operator LCS3(LCS3Element)

```
public static implicit operator LCS3(LCS3Element element)
```

Parameters

element [LCS3Element](#)

Returns

[LCS3](#)

# Class LCS3ElementGroup

Namespace: [Engine.Core](#)

Assembly: Engine.dll

```
public class LCS3ElementGroup
```

## Inheritance

[object](#) ← LCS3ElementGroup

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### LCS3ElementGroup(Guid, GCS3)

```
public LCS3ElementGroup(Guid groupID, GCS3 dependency)
```

#### Parameters

groupID [Guid](#)

dependency [GCS3](#)

### LCS3ElementGroup(Guid, List<LCS3Element>, GCS3)

```
public LCS3ElementGroup(Guid groupID, List<LCS3Element> elements, GCS3 dependency)
```

#### Parameters

groupID [Guid](#)

elements [List](#)<[LCS3Element](#)>

dependency [GCS3](#)

## Properties

### Bounding

```
public BoundingBox3D Bounding { get; }
```

#### Property Value

[BoundingBox3D](#)

### GroupID

```
public Guid GroupID { get; }
```

#### Property Value

[Guid](#)

### MaxDepth

Amount of layers the parent tree diagram has.

```
public int MaxDepth { get; }
```

#### Property Value

[int](#)

## Methods

### Add(LCS3Element)

```
public void Add(LCS3Element element)
```

Parameters

element [LCS3Element](#)

## Contains(LCS3Element)

```
public bool Contains(LCS3Element element)
```

Parameters

element [LCS3Element](#)

Returns

[bool](#)

## ContainsMesh(int, out List<Mesh3D>)

```
public bool ContainsMesh(int depth, out List<Mesh3D> meshes)
```

Parameters

depth [int](#)

meshes [List](#)<[Mesh3D](#)>

Returns

[bool](#)

## GetAllByDepth(int)

```
public List<LCS3Element> GetAllByDepth(int depth)
```

Parameters

depth [int](#)

Returns

[List](#) <[LCS3Element](#)>

## GetAllLCS3sByDepth(int)

```
public List<LCS3> GetAllLCS3sByDepth(int depth)
```

Parameters

depth [int](#)

Returns

[List](#) <[LCS3](#)>

## GetBounding(int)

```
public BoundingBox3D GetBounding(int depth)
```

Parameters

depth [int](#)

Returns

[BoundingBox3D](#)

## InBoundsMesh(int, Vector3, out Mesh3D[])

The InBoundsMesh method checks if a given global position is within a hierarchy of bounding volumes at different depths, and returns an array of all of those bounding volumes through its out parameter, containing only bounding volumes that contain a mesh

```
public bool InBoundsMesh(int depth, Vector3 globalPosition, out Mesh3D[] hits)
```

## Parameters

depth [int](#)

globalPosition [Vector3](#)

hits [Mesh3D](#)[]

## Returns

[bool](#)

A bool indicating whether or not any bounding volumes were found.

## Exceptions

[ArgumentOutOfRangeException](#)

## Remove(LCS3Element)

```
public void Remove(LCS3Element element)
```

## Parameters

element [LCS3Element](#)

# Class Layer

Namespace: [Engine.Core](#)

Assembly: Engine.dll

```
public class Layer
```

## Inheritance

[object](#) ← Layer

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### Layer(string, int)

```
public Layer(string name, int index)
```

#### Parameters

name [string](#)

index [int](#)

### Layer(string, int, int)

```
public Layer(string name, int id, int index)
```

#### Parameters

name [string](#)

id [int](#)

## Fields

### All

```
public static readonly List<Layer> All
```

#### Field Value

[List↗ <Layer>](#)

## Properties

### ID

The layer's unique id for the layer listing

```
public int ID { get; }
```

#### Property Value

[int↗](#)

### Index

The layer's unique index indicating its position (that's neither its identification nor its position in the list).

```
public int Index { get; }
```

#### Property Value

[int↗](#)

## this[int]

```
public Layer this[int id] { get; set; }
```

### Parameters

id [int](#)

### Property Value

[Layer](#)

## Name

The layer's unique name

```
public string Name { get; }
```

### Property Value

[string](#)

## Methods

### ChangeLayer(int, string, int?)

```
public static void ChangeLayer(int Id, string name, int? index)
```

### Parameters

Id [int](#)

name [string](#)

index [int](#)?

# Class Object3D

Namespace: [Engine.Core](#)

Assembly: Engine.dll

```
public class Object3D
```

## Inheritance

[object](#) ← Object3D

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### Object3D(Vector3, RotationVector3, Vector3, bool)

```
public Object3D(Vector3 position, RotationVector3 rotation, Vector3 scale, bool global  
= false)
```

#### Parameters

position [Vector3](#)

rotation [RotationVector3](#)

scale [Vector3](#)

global [bool](#)

### Object3D(Vector3, RotationVector3, bool)

```
public Object3D(Vector3 position, RotationVector3 rotation, bool global = false)
```

Parameters

position [Vector3](#)

rotation [RotationVector3](#)

global [bool](#) ↴

## Object3D(Vector3, bool)

```
public Object3D(Vector3 position, bool global = false)
```

Parameters

position [Vector3](#)

global [bool](#) ↴

## Object3D(bool)

```
public Object3D(bool global = false)
```

Parameters

global [bool](#) ↴

## Fields

### Components

```
public List<Component> Components
```

Field Value

[List](#) ↴ <[Component](#)>

## Name

`public string Name`

### Field Value

[string](#)

## customTags

`public List<Tag> customTags`

### Field Value

[List](#) <[Tag](#)>

## Properties

### InternalTag

`public Tag InternalTag { get; }`

### Property Value

[Tag](#)

### Local

local coordinate system of the object

`public LCS3 Local { get; }`

### Property Value

[LCS3](#)

## Parent

The set-accessor resets the local transformations the new Parent ones. To keep this Object3D's transformation, use the method instead.

```
public Object3D Parent { get; set; }
```

## Property Value

[Object3D](#)

## Methods

### AddComponent<T>(T)

```
public T AddComponent<T>(T component) where T : Component
```

#### Parameters

component T

#### Returns

T

#### Type Parameters

T

### FindChild(string)

Finds and returns the first child found with the inputted name.

```
public Object3D FindChild(string name)
```

Parameters

`name` [string](#)

Returns

[Object3D](#)

## GetComponent<T>()

```
public T GetComponent<T>() where T : Component
```

Returns

T

Type Parameters

T

## GetComponent<T>(T)

```
public T GetComponent<T>(T component) where T : Component
```

Parameters

`component` T

Returns

T

Type Parameters

T

## GetComponents<T>()

```
public List<T> GetComponents<T>() where T : Component
```

Returns

[List](#)<T>

Type Parameters

T

## GetComponents<T>(T)

```
public List<T> GetComponents<T>(T component) where T : Component
```

Parameters

component T

Returns

[List](#)<T>

Type Parameters

T

## RemoveComponent<T>(T)

```
public void RemoveComponent<T>(T component) where T : Component
```

Parameters

component T

## Type Parameters

T

### SetParent(Object3D, bool)

```
public void SetParent(Object3D parent, bool keepGlobalTransformation = false)
```

#### Parameters

parent [Object3D](#)

keepGlobalTransformation [bool](#) ↗

## Operators

### implicit operator LCS3(Object3D)

```
public static implicit operator LCS3(Object3D o)
```

#### Parameters

o [Object3D](#)

#### Returns

[LCS3](#)

# Class Particle

Namespace: [Engine.Core](#)

Assembly: Engine.dll

```
public class Particle
```

## Inheritance

[object](#) ← Particle

## Derived

[Photon](#)

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## Particle(Vector3, bool)

```
public Particle(Vector3 position, bool eliminateOnHit = false)
```

## Parameters

position [Vector3](#)

eliminateOnHit [bool](#)

# Properties

## Position

```
public Vector3 Position { get; }
```

Property Value

[Vector3](#)

## Methods

~Particle()

```
protected ~Particle()
```

GetCollisions()

```
public List<ParticleCollision3D> GetCollisions()
```

Returns

[List](#) <[ParticleCollision3D](#)>

SetPosition(Vector3)

```
public void SetPosition(Vector3 newPosition)
```

Parameters

`newPosition` [Vector3](#)

Update()

```
public void Update()
```

# Class Ray

Namespace: [Engine.Core](#)

Assembly: Engine.dll

```
public class Ray : IParticleHandler, IRay
```

Inheritance

[object](#) ← Ray

Implements

[IParticleHandler](#), [IRay](#)

Derived

[LightRay](#)

Inherited Members

[object.ToString\(\)](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),  
[object.MemberwiseClone\(\)](#)

## Constructors

Ray(Straight3D, float)

```
public Ray(Straight3D ray, float rayStart)
```

Parameters

ray [Straight3D](#)

rayStart [float](#)

Ray(Straight3D, float, float)

```
public Ray(Straight3D ray, float rayStart, float samplingRate)
```

Parameters

ray [Straight3D](#)

rayStart [float](#)

samplingRate [float](#)

## Properties

### Particle

```
public Particle Particle { get; }
```

Property Value

[Particle](#)

## Methods

### EliminateAllParticles()

```
public virtual void EliminateAllParticles()
```

### EliminateParticle(ref Particle)

```
public virtual void EliminateParticle(ref Particle particle)
```

Parameters

particle [Particle](#)

### Emit(float)

```
public Task<List<ParticleCollision3D>> Emit(float distance)
```

Parameters

**distance** [float](#)

Returns

[Task](#) <[List](#) <ParticleCollision3D>>

## OnRayCollision(List<ParticleCollision3D>)

```
public void OnRayCollision(List<ParticleCollision3D> collisions)
```

Parameters

**collisions** [List](#) <ParticleCollision3D>

# Class Tag

Namespace: [Engine.Core](#)

Assembly: Engine.dll

```
public class Tag
```

## Inheritance

[object](#) ← Tag

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## Tag(string)

```
public Tag(string name)
```

## Parameters

name [string](#)

# Fields

## Tags

```
public static List<Tag> Tags
```

## Field Value

[List](#) <[Tag](#)>

# Properties

## Name

```
public string Name { get; set; }
```

## Property Value

[string ↗](#)

# Methods

## RemoveTag(Tag)

```
public static void RemoveTag(Tag tag)
```

## Parameters

tag [Tag](#)

## SetName(string)

```
public void SetName(string name)
```

## Parameters

name [string ↗](#)

## SetName(string, int)

```
public static void SetName(string name, int index)
```

## Parameters

name [string](#)

index [int](#)

## Operators

### implicit operator int(Tag)

```
public static implicit operator int(Tag t)
```

Parameters

t [Tag](#)

Returns

[int](#)

### implicit operator string(Tag)

```
public static implicit operator string(Tag t)
```

Parameters

t [Tag](#)

Returns

[string](#)

### implicit operator Tag(int)

```
public static implicit operator Tag(int i)
```

Parameters

i [int](#)

Returns

[Tag](#)

implicit operator Tag(string)

```
public static implicit operator Tag(string s)
```

Parameters

s [string](#)

Returns

[Tag](#)

# Namespace Engine.Core.Components

## Classes

[Camera](#)

[Camera.CameraPlane](#)

[Edge3D](#)

[Face3D](#)

[LCS3](#)

[Mesh3D](#)

[Vertex3D](#)

# Class Camera

Namespace: [Engine.Core.Components](#)

Assembly: Engine.dll

```
public class Camera : Component
```

## Inheritance

[object](#) ← [Component](#) ← Camera

## Inherited Members

[Component.Dependency](#) , [object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

Camera(Object3D, float, float)

```
public Camera(Object3D dependency, float distanceToNearPlane = 0.25, float  
distanceToFarPlane = 1000)
```

## Parameters

dependency [Object3D](#)

distanceToNearPlane [float](#)

distanceToFarPlane [float](#)

## Properties

FarPlane

```
public Camera.CameraPlane FarPlane { get; }
```

Property Value

[Camera.CameraPlane](#)

## FarPlaneDistance

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

Property Value

[float](#)

## ImagePlane

```
public Camera.CameraPlane ImagePlane { get; }
```

Property Value

[Camera.CameraPlane](#)

## NearPlane

```
public Camera.CameraPlane NearPlane { get; }
```

Property Value

[Camera.CameraPlane](#)

## NearPlaneDistance

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

Property Value

[float](#)

## Methods

### AnyLightSource([out List<LightSource>](#))

True if any light source affects the scene part that is rendered. Outputs all light sources that affect the scene.

```
[Obsolete("Not implemented")]
public bool AnyLightSource(out List<LightSource> lightSources)
```

#### Parameters

[lightSources](#) [List<LightSource>](#)

#### Returns

[bool](#)

### RenderPoint([float](#), [float](#))

```
public Pixel RenderPoint(float ds, float dt)
```

#### Parameters

[ds](#) [float](#)

[dt](#) [float](#)

#### Returns

[Pixel](#)

# Class Camera.CameraPlane

Namespace: [Engine.Core.Components](#)

Assembly: Engine.dll

```
public class Camera.CameraPlane
```

## Inheritance

[object](#) ← Camera.CameraPlane

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### CameraPlane(float, Object3D)

```
public CameraPlane(float distance, Object3D dependency)
```

## Parameters

distance [float](#)

dependency [Object3D](#)

## Properties

### BoundaryS

```
public float[] BoundaryS { get; }
```

## Property Value

[float](#)[]

## BoundaryT

```
public float[] BoundaryT { get; }
```

Property Value

[float](#)[]

## GlobalHorizontal

```
public Vector3 GlobalHorizontal { get; }
```

Property Value

[Vector3](#)

## GlobalVertical

```
public Vector3 GlobalVertical { get; }
```

Property Value

[Vector3](#)

## Normal

```
public Vector3 Normal { get; }
```

Property Value

[Vector3](#)

## plane

```
public Plane plane { get; }
```

Property Value

[Plane](#)

## Methods

GetPoint(float, float)

```
public Vector3 GetPoint(float s, float t)
```

Parameters

s [float](#)

t [float](#)

Returns

[Vector3](#)

# Class Edge3D

Namespace: [Engine.Core.Components](#)

Assembly: Engine.dll

```
public class Edge3D : Component
```

## Inheritance

[object](#) ← [Component](#) ← Edge3D

## Inherited Members

[Component.Dependency](#) , [object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### Edge3D(Vertex3D, Vertex3D, Object3D)

```
public Edge3D(Vertex3D A, Vertex3D B, Object3D dependency)
```

## Parameters

A [Vertex3D](#)

B [Vertex3D](#)

dependency [Object3D](#)

## Fields

### A

```
public Vertex3D A
```

Field Value

[Vertex3D](#)

B

```
public Vertex3D B
```

Field Value

[Vertex3D](#)

## Methods

### GetVertexOnEdge(float)

Gets a position vector on the edge t needs to be a value between 0 and 1

```
public Vertex3D GetVertexOnEdge(float t)
```

Parameters

t [float](#)

Returns

[Vertex3D](#)

Returns a [Vertex3D](#).

Exceptions

[ArgumentOutOfRangeException](#)

# Class Face3D

Namespace: [Engine.Core.Components](#)

Assembly: Engine.dll

```
public class Face3D : Component
```

## Inheritance

[object](#) ← [Component](#) ← Face3D

## Inherited Members

[Component.Dependency](#) , [object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

Face3D(Vertex3D[], Edge3D[], Object3D)

```
public Face3D(Vertex3D[] vertices, Edge3D[] edges, Object3D dependency)
```

## Parameters

vertices [Vertex3D](#)[]

edges [Edge3D](#)[]

dependency [Object3D](#)

## Properties

NormalVector

```
public Vector3 NormalVector { get; }
```

Property Value

[Vector3](#)

## Methods

### Contains(Vector3)

```
[Obsolete("Not Implemented, needs LCS3.ToLocalPosition(gp) first")]
public bool Contains(Vector3 globalPosition)
```

Parameters

`globalPosition` [Vector3](#)

Returns

`bool` ↗

### GetVertexOnFace(float, float)

```
public Vertex3D GetVertexOnFace(float s, float t)
```

Parameters

`s` [float](#) ↗

`t` [float](#) ↗

Returns

[Vertex3D](#)

### SetFace(Vertex3D[], Edge3D[])

Setups the face and returns if the operation was successfull

```
public bool SetFace(Vertex3D[] vertices, Edge3D[] edges)
```

Parameters

vertices [Vertex3D](#)[]

edges [Edge3D](#)[]

Returns

bool ↗

Returns true if the face is set correctly

# Class LCS3

Namespace: [Engine.Core.Components](#)

Assembly: Engine.dll

```
public class LCS3 : Component, IEnumerable<LCS3>, IEnumerable, IEnumerator<LCS3>,
IEnumerator, IDisposable
```

## Inheritance

[object](#) ← [Component](#) ← LCS3

## Implements

[IEnumerable](#), [IEnumerator](#), [IEnumerator](#)<LCS3>, [IEnumerator](#), [IDisposable](#)

## Inherited Members

[Component.Dependency](#), [object.ToString\(\)](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),  
[object.MemberwiseClone\(\)](#)

# Constructors

## LCS3(Object3D)

```
public LCS3(Object3D dependency)
```

## Parameters

dependency [Object3D](#)

# Fields

## axis

```
public readonly Straight3D[] axis
```

Field Value

[Straight3D\[\]](#)

children

`protected List<LCS3> children`

Field Value

[List<LCS3>](#)

## Properties

ChildCount

`public int ChildCount { get; }`

Property Value

[int](#)

Current

`public LCS3 Current { get; }`

Property Value

[LCS3](#)

GlobalPosition

`public Vector3 GlobalPosition { get; set; }`

Property Value

[Vector3](#)

## GlobalRotation

```
public RotationVector3 GlobalRotation { get; set; }
```

Property Value

[RotationVector3](#)

## GlobalScale

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

Property Value

[Vector3](#)

## HasChild

Is true if the LCS has at least one child Lcs

```
public bool HasChild { get; }
```

Property Value

[bool](#)

## LocalPosition

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

Property Value

[Vector3](#)

## LocalRotation

```
public RotationVector3 LocalRotation { get; set; }
```

Property Value

[RotationVector3](#)

## LocalScale

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

Property Value

[Vector3](#)

## Parent

The higher the value the more parents it has, starting from 0: no parents

```
public LCS3 Parent { get; }
```

Property Value

[LCS3](#)

## Methods

### AxisToGlobalUnitVectors()

```
public Vector3[] AxisToGlobalUnitVectors()
```

Returns

[Vector3\[\]](#)

Dispose()

```
public void Dispose()
```

Dispose(bool)

```
protected virtual void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

~LCS3()

```
protected ~LCS3()
```

GetChilds()

```
public List<LCS3> GetChilds()
```

Returns

[List](#) <[LCS3](#)>

GetEnumerator()

```
public IEnumarator<LCS3> GetEnumarator()
```

Returns

[IEnumarator](#) <LCS3>

## MoveNext()

```
public bool MoveNext()
```

Returns

[bool](#)

## Reset()

```
public void Reset()
```

## ToLocalPositionVector(Vector3)

```
[Obsolete("Not Implemented")]
public Vector3 ToLocalPositionVector(Vector3 globalPosition)
```

Parameters

globalPosition [Vector3](#)

Returns

[Vector3](#)

# Class Mesh3D

Namespace: [Engine.Core.Components](#)

Assembly: Engine.dll

```
public class Mesh3D : Component
```

## Inheritance

[object](#) ← [Component](#) ← Mesh3D

## Inherited Members

[Component.Dependency](#) , [object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### Mesh3D(Object3D)

```
public Mesh3D(Object3D dependency)
```

#### Parameters

dependency [Object3D](#)

### Mesh3D(List<Vertex3D>, List<Edge3D>, List<Face3D>, Object3D)

```
public Mesh3D(List<Vertex3D> vertices, List<Edge3D> edges, List<Face3D> faces,  
Object3D dependency)
```

#### Parameters

vertices [List](#)<[Vertex3D](#)>

edges [List](#)<Edge3D>

faces [List](#)<Face3D>

dependency [Object3D](#)

## Properties

### EdgesCount

`public int EdgesCount { get; }`

Property Value

[int](#)

### FacesCount

`public int FacesCount { get; }`

Property Value

[int](#)

### VerticesCount

`public int VerticesCount { get; }`

Property Value

[int](#)

## Methods

## AddEdge(Edge3D)

```
public void AddEdge(Edge3D edge)
```

Parameters

edge [Edge3D](#)

## AddEdge(Vertex3D, Vertex3D)

```
public void AddEdge(Vertex3D v1, Vertex3D v2)
```

Parameters

v1 [Vertex3D](#)

v2 [Vertex3D](#)

## AddEdge(Vertex3D[])

```
public void AddEdge(Vertex3D[] vertices)
```

Parameters

vertices [Vertex3D\[\]](#)

## AddFace(Face3D)

```
public void AddFace(Face3D face)
```

Parameters

face [Face3D](#)

## AddFace(Vertex3D[])

```
public void AddFace(Vertex3D[] vertices)
```

### Parameters

`vertices` [Vertex3D\[\]](#)

## AddFace(Vertex3D[], Edge3D[])

```
public void AddFace(Vertex3D[] vertices, Edge3D[] edges)
```

### Parameters

`vertices` [Vertex3D\[\]](#)

`edges` [Edge3D\[\]](#)

## AddVertex(Vertex3D)

```
public void AddVertex(Vertex3D vertex)
```

### Parameters

`vertex` [Vertex3D](#)

## FindEdge(Edge3D)

```
public Edge3D FindEdge(Edge3D edge)
```

### Parameters

`edge` [Edge3D](#)

Returns

[Edge3D](#)

## FindFace(Face3D)

```
public Face3D FindFace(Face3D face)
```

Parameters

face [Face3D](#)

Returns

[Face3D](#)

## FindIndex(Edge3D)

```
public int FindIndex(Edge3D edge)
```

Parameters

edge [Edge3D](#)

Returns

[int↗](#)

## FindIndex(Face3D)

```
public int FindIndex(Face3D face)
```

Parameters

face [Face3D](#)

Returns

[int ↗](#)

## FindIndex(Vertex3D)

```
public int FindIndex(Vertex3D vertex)
```

Parameters

**vertex** [Vertex3D](#)

Returns

[int ↗](#)

## FindVertex(Vertex3D)

```
public Vertex3D FindVertex(Vertex3D vertex)
```

Parameters

**vertex** [Vertex3D](#)

Returns

[Vertex3D](#)

## GetAllFaces()

```
public List<Face3D> GetAllFaces()
```

Returns

[List ↗ <Face3D>](#)

## GetAllVertices()

```
public List<Vertex3D> GetAllVertices()
```

Returns

[List](#) <[Vertex3D](#)>

## GetBoundingBox()

```
public BoundingBox3D GetBoundingBox()
```

Returns

[BoundingBox3D](#)

## GetEdge(int)

```
public Edge3D GetEdge(int index)
```

Parameters

**index** [int](#)

Returns

[Edge3D](#)

## GetFace(int)

```
public Face3D GetFace(int index)
```

Parameters

**index** [int](#)

Returns

[Face3D](#)

**GetVertex(int)**

```
public Vertex3D GetVertex(int index)
```

Parameters

index [int](#)

Returns

[Vertex3D](#)

# Class Vertex3D

Namespace: [Engine.Core.Components](#)

Assembly: Engine.dll

```
public class Vertex3D : Component
```

## Inheritance

[object](#) ← [Component](#) ← Vertex3D

## Inherited Members

[Component.Dependency](#) , [object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### Vertex3D(Vector3, Object3D)

```
public Vertex3D(Vector3 localPosition, Object3D dependency)
```

## Parameters

localPosition [Vector3](#)

dependency [Object3D](#)

## Properties

### GlobalPosition

```
public Vector3 GlobalPosition { get; }
```

## Property Value

[Vector3](#)

# LocalPosition

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

Property Value

[Vector3](#)

## Methods

SetPosition(Vector3)

```
[Obsolete("Global Position is got wrongly.")]  
public void SetPosition(Vector3 localPosition)
```

Parameters

localPosition [Vector3](#)

# Namespace Engine.Core.Events

## Classes

[Event](#)

[Event<T>](#)

[Event<T0, T1>](#)

# Class Event

Namespace: [Engine.Core.Events](#)

Assembly: Engine.dll

```
public class Event
```

## Inheritance

[object](#) ← Event

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Methods

## AddListener(Action)

```
public void AddListener(Action method)
```

### Parameters

method [Action](#)

## Invoke()

```
public void Invoke()
```

## RemoveAllListeners()

```
public void RemoveAllListeners()
```

## RemoveListener(Action)

```
public void RemoveListener(Action method)
```

Parameters

method [Action](#)

## Operators

### operator +(Event, Action)

```
public static Event operator +(Event e, Action method)
```

Parameters

e [Event](#)

method [Action](#)

Returns

[Event](#)

### operator -(Event, Action)

```
public static Event operator -(Event e, Action method)
```

Parameters

e [Event](#)

method [Action](#)

Returns

[Event](#)



# Class Event<T>

Namespace: [Engine.Core.Events](#)

Assembly: Engine.dll

```
public class Event<T>
```

## Type Parameters

T

### Inheritance

[object](#) ← Event<T>

### Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Methods

### AddListener(Action<T>)

```
public void AddListener(Action<T> method)
```

#### Parameters

method [Action](#)<T>

### Invoke(T)

```
public void Invoke(T t)
```

#### Parameters

## RemoveAllListeners()

```
public void RemoveAllListeners()
```

## RemoveListener(Action<T>)

```
public void RemoveListener(Action<T> method)
```

### Parameters

method [Action](#)<T>

## Operators

### operator +(Event<T>, Action<T>)

```
public static Event<T> operator +(Event<T> e, Action<T> method)
```

### Parameters

e [Event](#)<T>

method [Action](#)<T>

### Returns

[Event](#)<T>

### operator -(Event<T>, Action<T>)

```
public static Event<T> operator -(Event<T> e, Action<T> method)
```

Parameters

e [Event](#)<T>

method [Action](#) <T>

Returns

[Event](#)<T>

# Class Event<T0, T1>

Namespace: [Engine.Core.Events](#)

Assembly: Engine.dll

```
public class Event<T0, T1>
```

## Type Parameters

T0

T1

## Inheritance

[object](#) ← Event<T0, T1>

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Methods

## AddListener(Action<T0, T1>)

```
public void AddListener(Action<T0, T1> method)
```

## Parameters

method [Action](#)<T0, T1>

## Invoke(T0, T1)

```
public void Invoke(T0 t0, T1 t1)
```

Parameters

t0 T0

t1 T1

## RemoveAllListeners()

```
public void RemoveAllListeners()
```

## RemoveListener(Action<T0, T1>)

```
public void RemoveListener(Action<T0, T1> method)
```

Parameters

method [Action](#)<T0, T1>

## Operators

### operator +(Event<T0, T1>, Action<T0, T1>)

```
public static Event<T0, T1> operator +(Event<T0, T1> e, Action<T0, T1> method)
```

Parameters

e [Event](#)<T0, T1>

method [Action](#)<T0, T1>

Returns

[Event](#)<T0, T1>

## operator -(Event<T0, T1>, Action<T0, T1>)

```
public static Event<T0, T1> operator -(Event<T0, T1> e, Action<T0, T1> method)
```

### Parameters

e [Event](#)<T0, T1>

method [Action](#)<T0, T1>

### Returns

[Event](#)<T0, T1>

# Namespace Engine.Core.Maths

## Classes

[AffineMatrix3x3](#)

[AffineMatrix4x4](#)

[AugmentedMatrix](#)

[DilationMatrix2x2](#)

[DilationMatrix3x3](#)

[FloatEnumerator](#)

[GCS3](#)

[Gauss](#)

[Gauss.EliminationResult](#)

[HomogeneousObject3D](#)

[Mathf](#)

[Matrix](#)

[Matrix3x3](#)

[MatrixMxN](#)

[MirrorMatrix3x3](#)

[Plane](#)

[Quaternion](#)

[RotationMatrix2x2](#)

[RotationMatrix3x3](#)

[RotationVector3](#)

Saves a globalRotation in RAD, allows input in degree or rad measure.

[SheerMatrix2x2](#)

[SheerMatrix3D](#)

## [Straight3D](#)

This code defines a class called Straight3D that represents a straight line in 3D space. It has properties such as OA, OB, and Dir, which represent the endpoints of the line, and the direction of the line, respectively. The class also contains methods such as GetPointAt, Intersection, Distance, and Distance with 2 output parameters, which return the point on the line at a given position, determine if the line intersects with another line, find the distance between the line and a point in 3D space, and find the distance between the line and another line in 3D space along with the closest points on both lines to the point of intersection, respectively. The class also has some static fields, such as x1\_Axis, x2\_Axis, and x3\_Axis, which represent the x, y, and z axes in 3D space, respectively.

## [Vector](#)

A vector with n dimensions

## [Vector2](#)

## [Vector3](#)

# Structs

## [Limits](#)

# Interfaces

## [IMatrix](#)

# Enums

## [Axis](#)

## [BaseDirections2D](#)

## [BasePlane](#)

## [HomogeneousType](#)

## [IntegrationApproximation](#)

## [MatrixOperation](#)

## [Plane.PlaneSetupType](#)

## [RotationConvention](#)

## [RotationMeasure](#)

## [RotationType](#)

[ShearConvention](#)

[Straight3D.LineSetupType](#)

[TransformationConvention](#)

# Class AffineMatrix3x3

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public abstract class AffineMatrix3x3
```

## Inheritance

[object](#) ← AffineMatrix3x3

## Derived

[AffineMatrix4x4](#)

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Class AffineMatrix4x4

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class AffineMatrix4x4 : AffineMatrix3x3
```

## Inheritance

[object](#) ← [AffineMatrix3x3](#) ← AffineMatrix4x4

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### AffineMatrix4x4()

```
public AffineMatrix4x4()
```

### AffineMatrix4x4(TransformationConvention, RotationConvention)

```
public AffineMatrix4x4(TransformationConvention TConvention = TransformationConvention.RDS,  
RotationConvention RConvention = RotationConvention.X1X2X3)
```

## Parameters

TConvention [TransformationConvention](#)

RConvention [RotationConvention](#)

## Fields

## ATM4x4

```
public MatrixMxN ATM4x4
```

Field Value

[MatrixMxN](#)

## TConvention

```
public TransformationConvention TConvention
```

Field Value

[TransformationConvention](#)

## dilationSub

```
public DilationMatrix3x3 dilationSub
```

Field Value

[DilationMatrix3x3](#)

## rotationQ

```
public Quaternion rotationQ
```

Field Value

[Quaternion](#)

## rotationSub

```
public RotationMatrix3x3 rotationSub
```

Field Value

[RotationMatrix3x3](#)

sheerSub

```
public SheerMatrix3D sheerSub
```

Field Value

[SheerMatrix3D](#)

translation

```
public HomogeneousObject3D translation
```

Field Value

[HomogeneousObject3D](#)

# Class AugmentedMatrix

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class AugmentedMatrix
```

## Inheritance

[object](#) ← AugmentedMatrix

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### AugmentedMatrix(IMatrix, IMatrix)

```
public AugmentedMatrix(IMatrix matrix, IMatrix augmentation)
```

## Parameters

matrix [IMatrix](#)

augmentation [IMatrix](#)

## Properties

### Augmentation

```
public IMatrix Augmentation { get; set; }
```

## Property Value

[IMatrix](#)

# Matrix

```
public IMatrix Matrix { get; set; }
```

Property Value

[IMatrix](#)

## Methods

### RowOperation(int, int, MatrixOperation, float)

result = target operation source (eg. target row + source row = new target row) If the operation is a scalar operation then use float f and input something random into source as it won't be used.

```
public void RowOperation(int target, int source, MatrixOperation operation, float f = 0)
```

Parameters

target [int](#)

source [int](#)

operation [MatrixOperation](#)

f [float](#)

### SwapRows(int, int)

```
public void SwapRows(int from, int to)
```

Parameters

from [int](#)

to [int](#)



# Enum Axis

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum Axis
```

## Fields

X1 = 0

X2 = 1

X3 = 2

# Enum BaseDirections2D

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum BaseDirections2D
```

## Fields

Horizontal = 0

Vertical = 1

# Enum BasePlane

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum BasePlane
```

## Fields

X1X2 = 0

X1X3 = 2

X2X3 = 1

# Class DilationMatrix2x2

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class DilationMatrix2x2
```

## Inheritance

[object](#) ← DilationMatrix2x2

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### DilationMatrix2x2(float, float)

```
public DilationMatrix2x2(float x, float y)
```

## Parameters

x [float](#)

y [float](#)

## Fields

### D2x2

```
public readonly MatrixMxN D2x2
```

## Field Value

[MatrixMxN](#)

# Operators

## implicit operator MatrixMxN(DilationMatrix2x2)

```
public static implicit operator MatrixMxN(DilationMatrix2x2 d)
```

### Parameters

d [DilationMatrix2x2](#)

### Returns

[MatrixMxN](#)

# Class DilationMatrix3x3

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class DilationMatrix3x3
```

## Inheritance

[object](#) ← DilationMatrix3x3

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### DilationMatrix3x3(float, float, float)

```
public DilationMatrix3x3(float x1, float x2, float x3)
```

## Parameters

x1 [float](#)

x2 [float](#)

x3 [float](#)

## Fields

### D3x3

```
public readonly Matrix3x3 D3x3
```

Field Value

[Matrix3x3](#)

## Operators

implicit operator Matrix3x3(DilationMatrix3x3)

```
public static implicit operator Matrix3x3(DilationMatrix3x3 d)
```

Parameters

d [DilationMatrix3x3](#)

Returns

[Matrix3x3](#)

# Class FloatEnumerator

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class FloatEnumerator : IEnumerator<float>, IEnumerator, IDisposable
```

## Inheritance

[object](#) ← FloatEnumerator

## Implements

[IEnumerator](#)<[float](#)>, [IEnumerator](#), [IDisposable](#)

## Inherited Members

[object.ToString\(\)](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),  
[object.MemberwiseClone\(\)](#)

## Constructors

### FloatEnumerator(float[])

```
public FloatEnumerator(float[] data)
```

## Parameters

data [float](#)[]

## Properties

### Current

```
public float Current { get; }
```

## Property Value

[float](#)

## Methods

### Dispose()

```
public void Dispose()
```

### Dispose(bool)

```
protected virtual void Dispose(bool disposing)
```

#### Parameters

`disposing` [bool](#)

### ~FloatEnumerator()

```
protected ~FloatEnumerator()
```

### MoveNext()

```
public bool MoveNext()
```

#### Returns

[bool](#)

### Reset()

```
public void Reset()
```

# Class GCS3

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class GCS3
```

## Inheritance

[object](#) ← GCS3

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

GCS3()

```
public GCS3()
```

# Fields

## Axes

```
public static readonly Straight3D[] Axes
```

## Field Value

[Straight3D\[\]](#)

# Properties

## LightSources

```
public List<LightSource> LightSources { get; }
```

Property Value

[List](#)<[LightSource](#)>

## SpatialPartitioning

```
public SpatialPartitioning SpatialPartitioning { get; }
```

Property Value

[SpatialPartitioning](#)

## Methods

### Add(LightSource)

```
public void Add(LightSource light)
```

Parameters

**light** [LightSource](#)

### AddLCSToBase(LCS3)

Adds a local system to the base of the scene

```
public void AddLCSToBase(LCS3 system)
```

Parameters

**system** [LCS3](#)

## GetGroup(Guid)

```
public LCS3ElementGroup GetGroup(Guid GroupID)
```

Parameters

GroupID [Guid](#)

Returns

[LCS3ElementGroup](#)

## GetGroupContaining(LCS3)

```
public LCS3ElementGroup GetGroupContaining(LCS3 system)
```

Parameters

system [LCS3](#)

Returns

[LCS3ElementGroup](#)

## GetLCS3Element(LCS3)

```
public LCS3Element GetLCS3Element(LCS3 system)
```

Parameters

system [LCS3](#)

Returns

[LCS3Element](#)

## GetLCS3ElementIndex(LCS3)

```
public int GetLCS3ElementIndex(LCS3 system)
```

Parameters

system [LCS3](#)

Returns

[int](#)

## GroupExists(Guid)

```
public bool GroupExists(Guid GroupID)
```

Parameters

GroupID [Guid](#)

Returns

[bool](#)

## RegisterLCS(LCS3)

Register local system

```
public void RegisterLCS(LCS3 system)
```

Parameters

system [LCS3](#)

## Remove(LightSource)

```
public void Remove(LightSource light)
```

Parameters

**light** [LightSource](#)

## RemoveFromBase(LCS3)

Remove a local system from the base of the scene

```
public void RemoveFromBase(LCS3 system)
```

Parameters

**system** [LCS3](#)

## ReregisterAllLCS()

```
public void ReregisterAllLCS()
```

## ReregisterLCS(LCS3)

```
public void ReregisterLCS(LCS3 system)
```

Parameters

**system** [LCS3](#)

## UnregisterLCS(LCS3)

Unregister local system

```
public void UnregisterLCS(LCS3 system)
```

## Parameters

system [LCS3](#)

# Class Gauss

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class Gauss
```

## Inheritance

[object](#) ← Gauss

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Methods

## GaussianElimination(AugmentedMatrix, out EliminationResult)

Takes only 3x3 Matrix and 3x1 Augmentation

```
public static bool GaussianElimination(AugmentedMatrix matrix, out  
Gauss.EliminationResult result)
```

## Parameters

matrix [AugmentedMatrix](#)

result [Gauss.EliminationResult](#)

## Returns

[bool](#)

# Class Gauss.EliminationResult

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class Gauss.EliminationResult
```

## Inheritance

[object](#) ← Gauss.EliminationResult

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Fields

### Floats

```
public List<float> Floats
```

#### Field Value

[List](#) <[float](#)>

### Plane

```
public Plane Plane
```

#### Field Value

[Plane](#)

### Straight

```
public Straight3D Straight
```

Field Value

[Straight3D](#)

# Enum HomogeneousType

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum HomogeneousType
```

## Fields

Direction = 0

Position = 1

# Class HomogeneousObject3D

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class HomogeneousObject3D
```

## Inheritance

[object](#) ← HomogeneousObject3D

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### HomogeneousObject3D(Vector3, HomogeneousType)

```
public HomogeneousObject3D(Vector3 vector, HomogeneousType type)
```

#### Parameters

vector [Vector3](#)

type [HomogeneousType](#)

### HomogeneousObject3D(Vector3, float)

```
public HomogeneousObject3D(Vector3 vector, float w)
```

#### Parameters

vector [Vector3](#)

w [float](#)

## HomogeneousObject3D(float, float, float, float)

```
public HomogeneousObject3D(float x1, float x2, float x3, float w)
```

### Parameters

x1 [float](#)

x2 [float](#)

x3 [float](#)

w [float](#)

### Fields

#### W

```
public float w
```

### Field Value

[float](#)

### Properties

#### X1

```
public float x1 { get; set; }
```

### Property Value

[float](#)

## X2

```
public float X2 { get; set; }
```

Property Value

[float](#)

## X3

```
public float X3 { get; set; }
```

Property Value

[float](#)

## Methods

ToEuclideanSpace()

```
public Vector3 ToEuclideanSpace()
```

Returns

[Vector3](#)

## Operators

implicit operator Vector(HomogeneousObject3D)

```
public static implicit operator Vector(HomogeneousObject3D v)
```

Parameters

v [HomogeneousObject3D](#)

Returns

[Vector](#)

# Interface IMatrix

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public interface IMatrix
```

## Properties

### ColumnCount

```
int ColumnCount { get; }
```

### Property Value

[int](#)

### this[int, int]

```
float this[int r, int c] { get; set; }
```

### Parameters

r [int](#)

c [int](#)

### Property Value

[float](#)

### RowCount

```
int RowCount { get; }
```

Property Value

[int](#)

## Methods

### Operation(IMatrix, MatrixOperation, float)

```
IMatrix Operation(IMatrix m, MatrixOperation operation, float f = 0)
```

Parameters

m [IMatrix](#)

operation [MatrixOperation](#)

f [float](#)

Returns

[IMatrix](#)

### RowOperation(int, int, MatrixOperation, float)

```
void RowOperation(int target, int source, MatrixOperation operation, float f = 0)
```

Parameters

target [int](#)

source [int](#)

operation [MatrixOperation](#)

f [float](#)

## SwapRows(int, int)

```
void SwapRows(int from, int to)
```

### Parameters

from [int](#)

to [int](#)

# Enum IntegrationApproximation

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum IntegrationApproximation
```

## Fields

MidpointRule = 2

SimpsonsRule = 1

TrapezoidalRule = 0

# Struct Limits

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public struct Limits
```

## Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetType\(\)](#)

## Fields

### increment

The increment is the internal step width

```
public float increment
```

#### Field Value

[float](#)

### lowerLimit

```
public float lowerLimit
```

#### Field Value

[float](#)

### upperLimit

```
public float upperLimit
```

Field Value

[float](#)

## Methods

Contains(float)

```
public bool Contains(float f)
```

Parameters

f [float](#)

Returns

[bool](#)

# Class Mathf

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public static class Mathf
```

## Inheritance

[object](#) ← Mathf

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Fields

## CosinePiQuarter

```
public static readonly float CosinePiQuarter
```

### Field Value

[float](#)

## PiHalf

```
public static readonly float PiHalf
```

### Field Value

[float](#)

## PiQuarter

```
public static readonly float PiQuarter
```

Field Value

[float](#)

## SinePiQuarter

```
public static readonly float SinePiQuarter
```

Field Value

[float](#)

e

```
public const float e = 2.7182817
```

Field Value

[float](#)

pi

```
public const float pi = 3.1415927
```

Field Value

[float](#)

## Methods

### Abs(float)

```
public static float Abs(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Acos(float)

```
public static float Acos(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Approximately(float, float)

```
public static bool Approximately(float a, float b)
```

Parameters

a [float](#)

b [float](#)

Returns

[bool](#)

## Approximately(float, float, float)

```
public static bool Approximately(float a, float b, float tolerance)
```

Parameters

a [float](#)

b [float](#)

tolerance [float](#)

Returns

[bool](#)

## Asin(float)

```
public static float Asin(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Atan(float)

```
public static float Atan(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Atan2(float, float)

```
public static float Atan2(float x, float y)
```

Parameters

x [float](#)

y [float](#)

Returns

[float](#)

## Ceiling(float)

```
public static float Ceiling(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Cos(float)

```
public static float Cos(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Cosh(float)

```
public static float Cosh(float f)
```

Parameters

[f float](#)

Returns

[float](#)

## Cot(float)

```
public static float Cot(float f)
```

Parameters

[f float](#)

Returns

[float](#)

## Csc(float)

```
public static float Csc(float f)
```

Parameters

[f float](#)

Returns

[float](#)

## Deg2Rad(float)

```
public static float Deg2Rad(float deg)
```

Parameters

**deg** [float](#)

Returns

[float](#)

## Exp(float)

```
public static float Exp(float f)
```

Parameters

**f** [float](#)

Returns

[float](#)

## Factorial(int)

Iterative Implementation

```
public static int Factorial(int i)
```

Parameters

i int ↗

Returns

int ↗

## FactorialHelper(int, int)

```
public static int FactorialHelper(int i, int current)
```

Parameters

i int ↗

current int ↗

Returns

int ↗

## FactorialR(int)

Recursive Implementation

```
public static int FactorialR(int i)
```

Parameters

i int ↗

Returns

int ↗

## Floor(float)

```
public static float Floor(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Gamma(float)

```
public static float Gamma(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## IIEEERemainder(float, float)

```
public static float IIEEERemainder(float x, float y)
```

Parameters

x [float](#)

y [float](#)

Returns

[float](#)

## Integral(Limits, float, Func<float, float>, IntegrationApproximation)

Definite or improper integral

```
public static float Integral(Limits limits, float f, Func<float, float> innerFunction,  
IntegrationApproximation approximation)
```

Parameters

limits [Limits](#)

f [float](#)

innerFunction [Func<float, float>](#)

approximation [IntegrationApproximation](#)

Returns

[float](#)

## Log(float)

```
public static float Log(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Log(float, float)

```
public static float Log(float f, float newBase)
```

Parameters

f [float](#)

newBase [float](#)

Returns

[float](#)

## Log10(float)

```
public static float Log10(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Max(float, float)

```
public static float Max(float val1, float val2)
```

Parameters

val1 [float](#)

val2 [float](#)

Returns

[float](#)

## Min(float, float)

```
public static float Min(float val1, float val2)
```

Parameters

val1 [float](#)

val2 [float](#)

Returns

[float](#)

## NthSqrt(float, int)

```
public static float NthSqrt(float f, int n)
```

Parameters

f [float](#)

n [int](#)

Returns

[float](#)

## Pi(Func<float, float>, int, int)

```
public static float Pi(Func<float, float> innerFunction, int limit1, int limit2)
```

Parameters

innerFunction [Func](#)<[float](#), [float](#)>

limit1 [int](#)

limit2 [int](#)

Returns

[float](#)

## Pi(Func<float, float>, float, float, float)

```
public static float Pi(Func<float, float> innerFunction, float limit1, float limit2,  
float increment)
```

Parameters

innerFunction [Func](#)<[float](#), [float](#)>

limit1 [float](#)

limit2 [float](#)

increment [float](#)

Returns

[float](#)

## Pow(float, float)

```
public static float Pow(float b, float p)
```

Parameters

b [float](#)

p [float](#)

Returns

[float](#)

## Rad2Deg(float)

```
public static float Rad2Deg(float rad)
```

Parameters

rad [float](#)

Returns

[float](#)

## Round(float)

```
public static float Round(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Round(float, int)

```
public static float Round(float f, int digits)
```

Parameters

f [float](#)

digits [int](#)

Returns

[float](#)

## Round(float, int, MidpointRounding)

```
public static float Round(float f, int digits, MidpointRounding midpointRounding)
```

### Parameters

f [float](#)

digits [int](#)

midpointRounding [MidpointRounding](#)

### Returns

[float](#)

## RoundToInt(float)

```
public static int RoundToInt(float f)
```

### Parameters

f [float](#)

### Returns

[int](#)

## Sec(float)

```
public static float Sec(float f)
```

### Parameters

f [float](#)

Returns

[float](#)

## Sigma(Func<float, float>, int, int)

Iterative Implementation

```
public static float Sigma(Func<float, float> innerFunction, int limit1, int limit2)
```

Parameters

innerFunction [Func](#)<[float](#), [float](#)>

limit1 [int](#)

limit2 [int](#)

Returns

[float](#)

## Sigma(Func<float, float>, float, float, float)

Iterative Implementation

```
public static float Sigma(Func<float, float> innerFunction, float limit1, float limit2,  
float increment)
```

Parameters

innerFunction [Func](#)<[float](#), [float](#)>

limit1 [float](#)

limit2 [float](#)

increment [float](#)

Returns

[float](#)

## Sigma(params float[])

Iterative Sum

```
public static float Sigma(params float[] floats)
```

Parameters

[floats](#) [float](#)[]

Returns

[float](#)

## SigmaPow(float, List<float>)

Iterative Sum

```
public static float SigmaPow(float power, List<float> floats)
```

Parameters

[power](#) [float](#)

[floats](#) [List](#)<[float](#)>

Returns

[float](#)

## SigmaPow(float, params float[])

Iterative Sum

```
public static float SigmaPow(float power, params float[] floats)
```

Parameters

power [float](#)

floats [float](#)[]

Returns

[float](#)

## SigmaR(List<float>)

Recursive Sum

```
public static float SigmaR(List<float> floats)
```

Parameters

floats [List](#)<[float](#)>

Returns

[float](#)

## SigmaR(Func<float, float>, int, int, float)

Recursive Implementation

```
public static float SigmaR(Func<float, float> innerFunction, int limit1, int limit2, float increment = 1E-45)
```

Parameters

innerFunction [Func](#)<[float](#), [float](#)>

limit1 [int](#)

`limit2` [int](#)

`increment` [float](#)

Returns

[float](#)

## SigmaR(params float[])

Recursive Sum

```
public static float SigmaR(params float[] floats)
```

Parameters

`floats` [float](#)[]

Returns

[float](#)

## SigmaR\_PowElements(float, List<float>)

```
public static float SigmaR_PowElements(float power, List<float> floats)
```

Parameters

`power` [float](#)

`floats` [List](#)<[float](#)>

Returns

[float](#)

## SigmaR\_PowElements(float, params float[])

```
public static float SigmaR_PowElements(float power, params float[] floats)
```

Parameters

power [float](#)

floats [float](#)[]

Returns

[float](#)

## Sign(float)

```
public static int Sign(float f)
```

Parameters

f [float](#)

Returns

[int](#)

## Sin(float)

```
public static float Sin(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Sinc(float, bool)

```
public static float Sinc(float f, bool Pi = false)
```

Parameters

f [float](#)

Pi [bool](#)

Returns

[float](#)

## Sinh(float)

```
public static float Sinh(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Sqrt(float)

```
public static float Sqrt(float f)
```

Parameters

f [float](#)

Returns

[float](#)

## Tan(float)

```
public static float Tan(float f)
```

### Parameters

f [float](#)

### Returns

[float](#)

## Tanh(float)

```
public static float Tanh(float f)
```

### Parameters

f [float](#)

### Returns

[float](#)

## Truncate(float)

```
public static float Truncate(float f)
```

### Parameters

f [float](#)

### Returns

[float](#) ↗

# Class Matrix

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public abstract class Matrix : IMatrix
```

Inheritance

[object](#) ← Matrix

Implements

[IMatrix](#)

Derived

[Matrix3x3](#), [MatrixMxN](#)

Inherited Members

[object.ToString\(\)](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),  
[object.MemberwiseClone\(\)](#)

## Fields

I2x2

2x2 Identity Matrix

```
public static readonly Matrix I2x2
```

Field Value

[Matrix](#)

I3x3

3x3 Identity Matrix

```
public static readonly Matrix I3x3
```

Field Value

[Matrix](#)

I4x4

4x4 Identity Matrix

```
public static readonly Matrix I4x4
```

Field Value

[Matrix](#)

## Properties

ColumnCount

```
public abstract int ColumnCount { get; }
```

Property Value

[int](#)

this[int, int]

```
public abstract float this[int r, int c] { get; set; }
```

Parameters

r [int](#)

c [int](#)

Property Value

[float](#) ↗

RowCount

```
public abstract int RowCount { get; }
```

Property Value

[int](#) ↗

## Methods

GetIdentityMatrix<T>(int, out T)

```
public static bool GetIdentityMatrix<T>(int n, out T identityMatrix) where T : Matrix,  
IMatrix, new()
```

Parameters

n [int](#) ↗

identityMatrix T

Returns

[bool](#) ↗

Type Parameters

T

GetInverse<T>(T, out T)

```
public static bool GetInverse<T>(T matrix, out T inverseMatrix) where T : Matrix, IMatrix
```

Parameters

**matrix** T

**inverseMatrix** T

Returns

[bool](#)

Type Parameters

T

## Operation(IMatrix, MatrixOperation, float)

```
public abstract IMatrix Operation(IMatrix m, MatrixOperation operation, float f = 0)
```

Parameters

**m** [IMatrix](#)

**operation** [MatrixOperation](#)

**f** [float](#)

Returns

[IMatrix](#)

## RowOperation(int, int, MatrixOperation, float)

```
public abstract void RowOperation(int target, int source, MatrixOperation operation, float f = 0)
```

Parameters

target [int](#)

source [int](#)

operation [MatrixOperation](#)

f [float](#)

SwapRows(int, int)

```
public abstract void SwapRows(int from, int to)
```

Parameters

from [int](#)

to [int](#)

# Class Matrix3x3

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class Matrix3x3 : Matrix, IMatrix
```

## Inheritance

[object](#) ← [Matrix](#) ← Matrix3x3

## Implements

[IMatrix](#)

## Inherited Members

[Matrix.I2x2](#) , [Matrix.I4x4](#) , [Matrix.GetIdentityMatrix<T>\(int, out T\)](#) , [Matrix.GetInverse<T>\(T, out T\)](#) ,  
[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### Matrix3x3()

```
public Matrix3x3()
```

### Matrix3x3(List<Vector3>)

```
public Matrix3x3(List<Vector3> rows)
```

## Parameters

rows [List](#)<[Vector3](#)>

### Matrix3x3(params float[])

left to right and row by row

```
public Matrix3x3(params float[] values)
```

Parameters

values [float](#)[]

Exceptions

[ArgumentOutOfRangeException](#)

## Fields

I3x3

```
public static readonly Matrix3x3 I3x3
```

Field Value

[Matrix3x3](#)

## Properties

ColumnCount

```
public override int ColumnCount { get; }
```

Property Value

[int](#)

this[int, int]

```
public override float this[int r, int c] { get; set; }
```

Parameters

r [int](#)

c [int](#)

Property Value

[float](#)

## RowCount

```
public override int RowCount { get; }
```

Property Value

[int](#)

## Methods

### Addition(Matrix3x3)

```
public Matrix3x3 Addition(Matrix3x3 m)
```

Parameters

m [Matrix3x3](#)

Returns

[Matrix3x3](#)

### Addition(Matrix3x3, Matrix3x3)

```
public static Matrix3x3 Addition(Matrix3x3 n, Matrix3x3 m)
```

Parameters

n [Matrix3x3](#)

m [Matrix3x3](#)

Returns

[Matrix3x3](#)

## GetColumn(int)

```
public Vector3 GetColumn(int c)
```

Parameters

c [int](#)

Returns

[Vector3](#)

## GetDeterminant()

```
public float GetDeterminant()
```

Returns

[float](#)

## GetIdentityMatrix()

```
public static Matrix3x3 GetIdentityMatrix()
```

Returns

[Matrix3x3](#)

## GetInverse(Matrix3x3, out Matrix3x3)

```
public static bool GetInverse(Matrix3x3 m, out Matrix3x3 inverseMatrix)
```

Parameters

m [Matrix3x3](#)

inverseMatrix [Matrix3x3](#)

Returns

bool ↗

## GetRow(int)

```
public Vector3 GetRow(int r)
```

Parameters

r [int](#) ↗

Returns

[Vector3](#)

## GetValue(int, int)

```
public float GetValue(int r, int c)
```

Parameters

r [int](#)

c [int](#)

Returns

[float](#)

## Multiplication(Matrix3x3)

```
public Matrix3x3 Multiplication(Matrix3x3 m)
```

Parameters

m [Matrix3x3](#)

Returns

[Matrix3x3](#)

## Multiplication(Matrix3x3, Matrix3x3)

```
public static Matrix3x3 Multiplication(Matrix3x3 n, Matrix3x3 m)
```

Parameters

n [Matrix3x3](#)

m [Matrix3x3](#)

Returns

[Matrix3x3](#)

## Operation(IMatrix, MatrixOperation, float)

```
public override IMatrix Operation(IMatrix m, MatrixOperation operation, float f = 0)
```

Parameters

m [IMatrix](#)

operation [MatrixOperation](#)

f [float](#)

Returns

[IMatrix](#)

## Operation(Matrix3x3, Matrix3x3, MatrixOperation, float)

```
public static Matrix3x3 Operation(Matrix3x3 n, Matrix3x3 m, MatrixOperation operation, float f = 0)
```

Parameters

n [Matrix3x3](#)

m [Matrix3x3](#)

operation [MatrixOperation](#)

f [float](#)

Returns

[Matrix3x3](#)

## Operation(Matrix3x3, MatrixOperation, float)

Operates on the matrix (changes the target row) and outputs the row after the calculation as a vector.

```
public Matrix3x3 Operation(Matrix3x3 m, MatrixOperation operation, float f = 0)
```

## Parameters

m [Matrix3x3](#)

operation [MatrixOperation](#)

f [float](#)

## Returns

[Matrix3x3](#)

## RowAddition(int, int)

Calculates l+r. Doesn't change any row in the matrix l left side of the equation r right side of the equation

```
public Vector3 RowAddition(int l, int r)
```

## Parameters

l [int](#)

r [int](#)

## Returns

[Vector3](#)

Returns a row vector of the row l+r

## RowOperation(Matrix3x3, int, int, MatrixOperation, float)

```
public static void RowOperation(Matrix3x3 n, int target, int source, MatrixOperation
operation, float f = 0)
```

## Parameters

n [Matrix3x3](#)

target [int](#)

source [int](#)

operation [MatrixOperation](#)

f [float](#)

## RowOperation(int, int, MatrixOperation, float)

```
public override void RowOperation(int target, int source, MatrixOperation operation, float f  
= 0)
```

### Parameters

target [int](#)

source [int](#)

operation [MatrixOperation](#)

f [float](#)

## RowSubstraction(int, int)

```
public Vector3 RowSubstraction(int l, int r)
```

### Parameters

l [int](#)

r [int](#)

### Returns

[Vector3](#)

## ScalarDivision(Matrix3x3, float)

```
public static Matrix3x3 ScalarDivision(Matrix3x3 n, float f)
```

Parameters

n [Matrix3x3](#)

f [float](#)

Returns

[Matrix3x3](#)

## ScalarDivision(float)

```
public Matrix3x3 ScalarDivision(float f)
```

Parameters

f [float](#)

Returns

[Matrix3x3](#)

## ScalarMultiplication(Matrix3x3, float)

```
public static Matrix3x3 ScalarMultiplication(Matrix3x3 n, float f)
```

Parameters

n [Matrix3x3](#)

f [float](#)

Returns

[Matrix3x3](#)

## ScalarMultiplication(float)

```
public Matrix3x3 ScalarMultiplication(float f)
```

Parameters

f [float](#)

Returns

[Matrix3x3](#)

## SetValue(int, int, float)

```
public void SetValue(int r, int c, float f)
```

Parameters

r [int](#)

c [int](#)

f [float](#)

## Substraction(Matrix3x3)

```
public Matrix3x3 Substraction(Matrix3x3 m)
```

Parameters

m [Matrix3x3](#)

Returns

[Matrix3x3](#)

## Subtraction(Matrix3x3, Matrix3x3)

```
public static Matrix3x3 Subtraction(Matrix3x3 n, Matrix3x3 m)
```

### Parameters

n [Matrix3x3](#)

m [Matrix3x3](#)

### Returns

[Matrix3x3](#)

## SwapRows(int, int)

```
public override void SwapRows(int from, int to)
```

### Parameters

from [int](#)

to [int](#)

## Operators

### operator +(Matrix3x3, Matrix3x3)

```
public static Matrix3x3 operator +(Matrix3x3 n, Matrix3x3 m)
```

### Parameters

n [Matrix3x3](#)

m [Matrix3x3](#)

Returns

[Matrix3x3](#)

## operator /(Matrix3x3, float)

```
public static Matrix3x3 operator /(Matrix3x3 n, float f)
```

Parameters

n [Matrix3x3](#)

f [float](#)

Returns

[Matrix3x3](#)

## implicit operator MatrixMxN(Matrix3x3)

```
public static implicit operator MatrixMxN(Matrix3x3 m)
```

Parameters

m [Matrix3x3](#)

Returns

[MatrixMxN](#)

## operator \*(Matrix3x3, Matrix3x3)

```
public static Matrix3x3 operator *(Matrix3x3 n, Matrix3x3 m)
```

Parameters

n [Matrix3x3](#)

m [Matrix3x3](#)

Returns

[Matrix3x3](#)

operator \*(Matrix3x3, float)

```
public static Matrix3x3 operator *(Matrix3x3 n, float f)
```

Parameters

n [Matrix3x3](#)

f [float](#)

Returns

[Matrix3x3](#)

operator \*(float, Matrix3x3)

```
public static Matrix3x3 operator *(float f, Matrix3x3 n)
```

Parameters

f [float](#)

n [Matrix3x3](#)

Returns

[Matrix3x3](#)

## operator -(Matrix3x3, Matrix3x3)

```
public static Matrix3x3 operator -(Matrix3x3 n, Matrix3x3 m)
```

Parameters

n [Matrix3x3](#)

m [Matrix3x3](#)

Returns

[Matrix3x3](#)

## operator -(Matrix3x3)

```
public static Matrix3x3 operator -(Matrix3x3 n)
```

Parameters

n [Matrix3x3](#)

Returns

[Matrix3x3](#)

# Class MatrixMxN

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class MatrixMxN : Matrix, IMatrix
```

## Inheritance

[object](#) ← [Matrix](#) ← MatrixMxN

## Implements

[IMatrix](#)

## Inherited Members

[Matrix.I2x2](#) , [Matrix.I3x3](#) , [Matrix.I4x4](#) , [Matrix.GetIdentityMatrix<T>\(int, out T\)](#) ,  
[Matrix.GetInverse<T>\(T, out T\)](#) , [object.ToString\(\)](#) , [object.Equals\(object\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,  
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#)

## Constructors

### MatrixMxN()

```
public MatrixMxN()
```

### MatrixMxN(List<Vector>)

```
public MatrixMxN(List<Vector> rows)
```

## Parameters

rows [List](#)<[Vector](#)>

### MatrixMxN(int, int)

```
public MatrixMxN(int r, int c)
```

Parameters

r [int](#)

c [int](#)

## Properties

ColumnCount

```
public override int ColumnCount { get; }
```

Property Value

[int](#)

this[int, int]

```
public override float this[int r, int c] { get; set; }
```

Parameters

r [int](#)

c [int](#)

Property Value

[float](#)

RowCount

```
public override int RowCount { get; }
```

Property Value

[int ↗](#)

## Methods

### AddColumn(Vector)

```
public void AddColumn(Vector col)
```

Parameters

[col Vector](#)

### AddRow(Vector)

```
public void AddRow(Vector row)
```

Parameters

[row Vector](#)

### Addition(MatrixMxN)

```
public MatrixMxN Addition(MatrixMxN m)
```

Parameters

[m MatrixMxN](#)

Returns

[MatrixMxN](#)

## Addition(MatrixMxN, MatrixMxN)

```
public static MatrixMxN Addition(MatrixMxN n, MatrixMxN m)
```

### Parameters

n [MatrixMxN](#)

m [MatrixMxN](#)

### Returns

[MatrixMxN](#)

## Get2x2SubMatrices(MatrixMxN, out List<MatrixMxN>)

```
public bool Get2x2SubMatrices(MatrixMxN m, out List<MatrixMxN> subs)
```

### Parameters

m [MatrixMxN](#)

subs [List](#)<[MatrixMxN](#)>

### Returns

[bool](#)

## GetAdjointMatrix(out MatrixMxN)

```
public bool GetAdjointMatrix(out MatrixMxN adj)
```

### Parameters

adj [MatrixMxN](#)

### Returns

[bool](#)

## GetCofactorMatrix(out MatrixMxN)

```
public bool GetCofactorMatrix(out MatrixMxN fac)
```

Parameters

[fac](#) [MatrixMxN](#)

Returns

[bool](#)

## GetColumn(int)

```
public Vector GetColumn(int c)
```

Parameters

[c](#) [int](#)

Returns

[Vector](#)

## GetDeterminant(out float)

```
public bool GetDeterminant(out float determinant)
```

Parameters

[determinant](#) [float](#)

Returns

[bool](#)

## GetIdentityMatrix(int)

Creates a nxn identity and returns it.

```
public static MatrixMxN GetIdentityMatrix(int n)
```

Parameters

n [int](#)

Returns

[MatrixMxN](#)

## GetInverse(MatrixMxN, out MatrixMxN)

```
public static bool GetInverse(MatrixMxN m, out MatrixMxN inverseMatrix)
```

Parameters

m [MatrixMxN](#)

inverseMatrix [MatrixMxN](#)

Returns

[bool](#)

## GetMinorMatrix(out MatrixMxN)

```
public bool GetMinorMatrix(out MatrixMxN minor)
```

Parameters

`minor` [MatrixMxN](#)

Returns

`bool` ↗

## GetRow(int)

```
public Vector GetRow(int r)
```

Parameters

`r` [int](#) ↗

Returns

[Vector](#)

## GetSubMatrix(int, int, out MatrixMxN)

```
public bool GetSubMatrix(int r, int c, out MatrixMxN subMatrix)
```

Parameters

`r` [int](#) ↗

`c` [int](#) ↗

`subMatrix` [MatrixMxN](#)

Returns

`bool` ↗

## GetValue(int, int)

```
public float GetValue(int r, int c)
```

Parameters

r [int](#)

c [int](#)

Returns

[float](#)

## InsertRow(Vector, int)

```
public void InsertRow(Vector row, int index)
```

Parameters

row [Vector](#)

index [int](#)

## Multiplication(MatrixMxN)

```
public MatrixMxN Multiplication(MatrixMxN m)
```

Parameters

m [MatrixMxN](#)

Returns

[MatrixMxN](#)

## Multiplication(MatrixMxN, MatrixMxN)

```
public static MatrixMxN Multiplication(MatrixMxN n, MatrixMxN m)
```

Parameters

n [MatrixMxN](#)

m [MatrixMxN](#)

Returns

[MatrixMxN](#)

## Operation(IMatrix, MatrixOperation, float)

```
public override IMatrix Operation(IMatrix m, MatrixOperation operation, float f = 0)
```

Parameters

m [IMatrix](#)

operation [MatrixOperation](#)

f [float](#)

Returns

[IMatrix](#)

## Operation(MatrixMxN, MatrixMxN, MatrixOperation, float)

```
public static MatrixMxN Operation(MatrixMxN n, MatrixMxN m, MatrixOperation operation, float f = 0)
```

Parameters

n [MatrixMxN](#)

`m` [MatrixMxN](#)

`operation` [MatrixOperation](#)

`f` [float](#)

Returns

[MatrixMxN](#)

## Operation(MatrixMxN, MatrixOperation, float)

Operates on the matrix (changes the target row) and outputs the row after the calculation as a vector.

```
public MatrixMxN Operation(MatrixMxN m, MatrixOperation operation, float f = 0)
```

Parameters

`m` [MatrixMxN](#)

`operation` [MatrixOperation](#)

`f` [float](#)

Returns

[MatrixMxN](#)

## RemoveColumn(int)

```
public void RemoveColumn(int index)
```

Parameters

`index` [int](#)

## RemoveRow(int)

```
public void RemoveRow(int index)
```

Parameters

index [int](#)

## RowAddition(int, int)

Calculates  $I+r$ . Doesn't change any row in the matrix [l](#) left side of the equation [r](#) right side of the equation

```
public Vector RowAddition(int l, int r)
```

Parameters

[l](#) [int](#)

[r](#) [int](#)

Returns

[Vector](#)

Returns a row vector of the row  $I+r$

## RowOperation(MatrixMxN, int, int, MatrixOperation, float)

```
public static void RowOperation(MatrixMxN n, int target, int source, MatrixOperation  
operation, float f = 0)
```

Parameters

[n](#) [MatrixMxN](#)

[target](#) [int](#)

[source](#) [int](#)

`operation` [MatrixOperation](#)

`f` [float](#)

## RowOperation(int, int, MatrixOperation, float)

```
public override void RowOperation(int target, int source, MatrixOperation operation, float f = 0)
```

### Parameters

`target` [int](#)

`source` [int](#)

`operation` [MatrixOperation](#)

`f` [float](#)

## RowSubstraction(int, int)

```
public Vector RowSubstraction(int l, int r)
```

### Parameters

`l` [int](#)

`r` [int](#)

### Returns

[Vector](#)

## ScalarDivision(MatrixMxN, float)

```
public static MatrixMxN ScalarDivision(MatrixMxN n, float f)
```

Parameters

n [MatrixMxN](#)

f [float](#)

Returns

[MatrixMxN](#)

## ScalarDivision(float)

```
public MatrixMxN ScalarDivision(float f)
```

Parameters

f [float](#)

Returns

[MatrixMxN](#)

## ScalarMultiplication(MatrixMxN, float)

```
public static MatrixMxN ScalarMultiplication(MatrixMxN n, float f)
```

Parameters

n [MatrixMxN](#)

f [float](#)

Returns

[MatrixMxN](#)

## ScalarMultiplication(float)

```
public MatrixMxN ScalarMultiplication(float f)
```

Parameters

f [float](#)

Returns

[MatrixMxN](#)

## SetValue(int, int, float)

```
public void SetValue(int r, int c, float value)
```

Parameters

r [int](#)

c [int](#)

value [float](#)

## Substraction(MatrixMxN)

```
public MatrixMxN Substraction(MatrixMxN m)
```

Parameters

m [MatrixMxN](#)

Returns

[MatrixMxN](#)

## Substraction(MatrixMxN, MatrixMxN)

```
public static MatrixMxN Subtraction(MatrixMxN n, MatrixMxN m)
```

Parameters

n [MatrixMxN](#)

m [MatrixMxN](#)

Returns

[MatrixMxN](#)

## SwapRows(int, int)

```
public override void SwapRows(int from, int to)
```

Parameters

from [int](#)

to [int](#)

## ToVector(out Vector)

```
public bool ToVector(out Vector v)
```

Parameters

v [Vector](#)

Returns

[bool](#)

## ToVector3(out Vector3)

```
public bool ToVector3(out Vector3 v)
```

Parameters

v [Vector3](#)

Returns

[bool](#)

## Transpose()

```
public void Transpose()
```

# Operators

### operator +(MatrixMxN, MatrixMxN)

```
public static MatrixMxN operator +(MatrixMxN n, MatrixMxN m)
```

Parameters

n [MatrixMxN](#)

m [MatrixMxN](#)

Returns

[MatrixMxN](#)

### operator /(MatrixMxN, float)

```
public static MatrixMxN operator /(MatrixMxN n, float f)
```

Parameters

n [MatrixMxN](#)

f [float](#)

Returns

[MatrixMxN](#)

implicit operator Matrix3x3(MatrixMxN)

```
public static implicit operator Matrix3x3(MatrixMxN m)
```

Parameters

m [MatrixMxN](#)

Returns

[Matrix3x3](#)

operator \*(MatrixMxN, MatrixMxN)

```
public static MatrixMxN operator *(MatrixMxN n, MatrixMxN m)
```

Parameters

n [MatrixMxN](#)

m [MatrixMxN](#)

Returns

[MatrixMxN](#)

operator \*(MatrixMxN, float)

```
public static MatrixMxN operator *(MatrixMxN n, float f)
```

Parameters

n [MatrixMxN](#)

f [float](#)

Returns

[MatrixMxN](#)

## operator \*(float, MatrixMxN)

```
public static MatrixMxN operator *(float f, MatrixMxN n)
```

Parameters

f [float](#)

n [MatrixMxN](#)

Returns

[MatrixMxN](#)

## operator -(MatrixMxN, MatrixMxN)

```
public static MatrixMxN operator -(MatrixMxN n, MatrixMxN m)
```

Parameters

n [MatrixMxN](#)

m [MatrixMxN](#)

Returns

## operator -(MatrixMxN)

```
public static MatrixMxN operator -(MatrixMxN n)
```

### Parameters

n [MatrixMxN](#)

### Returns

[MatrixMxN](#)

# Enum MatrixOperation

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum MatrixOperation
```

## Fields

Addition = 0

Multiplication = 2

ScalarDivision = 4

ScalarMultiplication = 3

Substraction = 1

# Class MirrorMatrix3x3

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class MirrorMatrix3x3
```

## Inheritance

[object](#) ← MirrorMatrix3x3

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

MirrorMatrix3x3(Vector3)

```
public MirrorMatrix3x3(Vector3 normal)
```

## Parameters

normal [Vector3](#)

## Fields

M3x3

```
public readonly Matrix3x3 M3x3
```

## Field Value

[Matrix3x3](#)

## Mirror\_45Degree\_X1\_Roof

```
public static readonly Matrix3x3 Mirror_45Degree_X1_Roof
```

Field Value

[Matrix3x3](#)

## Mirror\_90Degree\_X1\_Roof

```
public static readonly Matrix3x3 Mirror_90Degree_X1_Roof
```

Field Value

[Matrix3x3](#)

## Mirror\_90Degree\_X2\_Roof

```
public static readonly Matrix3x3 Mirror_90Degree_X2_Roof
```

Field Value

[Matrix3x3](#)

## Mirror\_90Degree\_X3\_Roof

```
public static readonly Matrix3x3 Mirror_90Degree_X3_Roof
```

Field Value

[Matrix3x3](#)

## Mirror\_Cube\_Corner

```
public static readonly Matrix3x3 Mirror_Cube_Corner
```

Field Value

[Matrix3x3](#)

## Mirror\_FreeSpace

```
public static readonly Matrix3x3 Mirror_FreeSpace
```

Field Value

[Matrix3x3](#)

## Mirror\_X1

```
public static readonly Matrix3x3 Mirror_X1
```

Field Value

[Matrix3x3](#)

## Mirror\_X2

```
public static readonly Matrix3x3 Mirror_X2
```

Field Value

[Matrix3x3](#)

## Mirror\_X3

```
public static readonly Matrix3x3 Mirror_X3
```

Field Value

[Matrix3x3](#)

# Class Plane

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class Plane
```

## Inheritance

[object](#) ← Plane

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### Plane(Vector3, Vector3)

```
public Plane(Vector3 p, Vector3 n)
```

#### Parameters

p [Vector3](#)

n [Vector3](#)

### Plane(Vector3, Vector3, Vector3, PlaneSetupType)

```
public Plane(Vector3 p1, Vector3 s1, Vector3 s2, Plane.PlaneSetupType setupType)
```

#### Parameters

p1 [Vector3](#)

s1 [Vector3](#)

s2 [Vector3](#)

setupType [Plane.PlaneSetupType](#)

## Plane(Vector3, float)

```
public Plane(Vector3 n, float b)
```

Parameters

n [Vector3](#)

b [float](#)

## Properties

B

The "b" value of the coordinate-form " $n_1x_1+n_2x_2+n_3x_3-b$ "

```
public float B { get; }
```

Property Value

[float](#)

N

The normal vector of the plane

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

Property Value

[Vector3](#)

# P

A position vector on the plane

```
public Vector3 P { get; }
```

Property Value

[Vector3](#)

# S1

Span Vector3 1

```
public Vector3 S1 { get; }
```

Property Value

[Vector3](#)

# S2

Span Vector3 2

```
public Vector3 S2 { get; }
```

Property Value

[Vector3](#)

# Methods

## Contains(Straight3D)

```
public bool Contains(Straight3D s)
```

Parameters

s [Straight3D](#)

Returns

[bool](#)

## Contains(Vector3)

```
public bool Contains(Vector3 v)
```

Parameters

v [Vector3](#)

Returns

[bool](#)

## Contains(Vector3, out bool?)

```
public bool Contains(Vector3 v, out bool? above)
```

Parameters

v [Vector3](#)

above [bool](#)?

Returns

[bool](#)

## Distance(Plane)

```
public float Distance(Plane p)
```

Parameters

p [Plane](#)

Returns

[float](#)

## Distance(Straight3D)

```
public float Distance(Straight3D s)
```

Parameters

s [Straight3D](#)

Returns

[float](#)

## Distance(Straight3D, out Vector3)

```
public float Distance(Straight3D s, out Vector3 l)
```

Parameters

s [Straight3D](#)

l [Vector3](#)

Returns

[float](#)

## GetPoint\_00X()

Gets a point on the plane

```
public Vector3 GetPoint_00X()
```

Returns

[Vector3](#)

Returns the point for X1 = 0, X2 = 0 and X3 = b/n3

## GetPositionVectorAt(float, float)

```
public Vector3 GetPositionVectorAt(float s1, float s2)
```

Parameters

s1 [float](#)

s2 [float](#)

Returns

[Vector3](#)

## Intersection(Plane, Plane, out Straight3D, out Plane, out Vector3)

Get plane ∩ plane and outputs its results

```
public static bool Intersection(Plane p1, Plane p2, out Straight3D s, out Plane p, out  
Vector3 v)
```

Parameters

p1 [Plane](#)

p2 [Plane](#)

s [Straight3D](#)

p [Plane](#)

v [Vector3](#)

Returns

[bool](#) ↗

True if plane  $\cap$  plane  $\neq \emptyset$

## Intersection(Plane, Straight3D, out Straight3D, out Vector3)

Get plane  $\cap$  line and outputs its results

```
public static bool Intersection(Plane plane, Straight3D line, out Straight3D  
intersectionLine, out Vector3 intersectionPoint)
```

Parameters

plane [Plane](#)

line [Straight3D](#)

intersectionLine [Straight3D](#)

intersectionPoint [Vector3](#)

Returns

[bool](#) ↗

True if plane  $\cap$  line  $\neq \emptyset$

## Intersects(Plane, Straight3D)

```
public bool Intersects(Plane plane, Straight3D line)
```

Parameters

plane [Plane](#)

line [Straight3D](#)

Returns

[bool](#) ↗

# Enum Plane.PlaneSetupType

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum Plane.PlaneSetupType
```

## Fields

`_1PositionVector2SpanVectors = 1`

`_3PositionVectors = 0`

# Class Quaternion

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
[Obsolete("Quite none of it is implemented!")]
public class Quaternion
```

## Inheritance

[object](#) ← Quaternion

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#)

# Properties

## this[int]

```
public float this[int i] { get; set; }
```

## Parameters

i [int](#)

## Property Value

[float](#)

# Methods

## GetValue(int, out float)

```
public bool GetValue(int i, out float res)
```

Parameters

i [int](#)

res [float](#)

Returns

[bool](#)

**SetValue(int, float)**

```
public bool SetValue(int i, float value)
```

Parameters

i [int](#)

value [float](#)

Returns

[bool](#)

# Enum RotationConvention

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum RotationConvention
```

## Fields

X1X2X3 = 0

X1X3X2 = 1

X2X1X3 = 2

X2X3X1 = 3

X3X1X2 = 4

X3X2X1 = 5

# Class RotationMatrix2x2

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class RotationMatrix2x2
```

## Inheritance

[object](#) ← RotationMatrix2x2

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### RotationMatrix2x2(float)

```
public RotationMatrix2x2(float phi)
```

## Parameters

phi [float](#)

## Fields

### PiQuarter

```
public static readonly MatrixMxN PiQuarter
```

## Field Value

[MatrixMxN](#)

## R2x2

```
public readonly MatrixMxN R2x2
```

### Field Value

[MatrixMxN](#)

## Operators

implicit operator MatrixMxN(RotationMatrix2x2)

```
public static implicit operator MatrixMxN(RotationMatrix2x2 r)
```

### Parameters

r [RotationMatrix2x2](#)

### Returns

[MatrixMxN](#)

# Class RotationMatrix3x3

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class RotationMatrix3x3
```

## Inheritance

[object](#) ← RotationMatrix3x3

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### RotationMatrix3x3(float, Axis)

Creates a globalRotation matrix for the angle around the specific axis. Uses angles that are measured in radians.

```
public RotationMatrix3x3(float phi, Axis axis)
```

#### Parameters

phi [float](#)

axis [Axis](#)

### RotationMatrix3x3(float, float, float, RotationConvention)

Creates a combined globalRotation matrix. Uses angles that are measured in radians.

```
public RotationMatrix3x3(float alpha, float beta, float gamma, RotationConvention convention  
= RotationConvention.X1X2X3)
```

## Parameters

alpha [float](#)

beta [float](#)

gamma [float](#)

convention [RotationConvention](#)

## Fields

### MirrorX1

```
public static readonly Matrix3x3 MirrorX1
```

#### Field Value

[Matrix3x3](#)

### MirrorX2

```
public static readonly Matrix3x3 MirrorX2
```

#### Field Value

[Matrix3x3](#)

### MirrorX3

```
public static readonly Matrix3x3 MirrorX3
```

#### Field Value

[Matrix3x3](#)

## R3x3

```
public readonly Matrix3x3 R3x3
```

### Field Value

[Matrix3x3](#)

## Operators

### implicit operator Matrix3x3(RotationMatrix3x3)

```
public static implicit operator Matrix3x3(RotationMatrix3x3 r)
```

#### Parameters

r [RotationMatrix3x3](#)

#### Returns

[Matrix3x3](#)

### implicit operator MatrixMxN(RotationMatrix3x3)

```
public static implicit operator MatrixMxN(RotationMatrix3x3 r)
```

#### Parameters

r [RotationMatrix3x3](#)

#### Returns

[MatrixMxN](#)

# Enum RotationMeasure

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum RotationMeasure
```

## Fields

DEG = 1

RAD = 0

# Enum RotationType

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum RotationType
```

## Fields

Quaternion = 1

RotationMatrix = 0

# Class RotationVector3

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

Saves a globalRotation in RAD, allows input in degree or rad measure.

```
public class RotationVector3 : Vector3
```

## Inheritance

[object](#) ← [Vector3](#) ← RotationVector3

## Inherited Members

[Vector3.Zero](#), [Vector3.One](#), [Vector3.Length](#), [Vector3.LengthSquared](#), [Vector3.Normalized](#),  
[Vector3.Dimension](#), [Vector3.this\[int\]](#), [Vector3.ToMatrix\(Vector3, bool\)](#),  
[Vector3.CrossProduct\(Vector3, Vector3\)](#), [Vector3.DotProduct\(Vector3, Vector3\)](#),  
[Vector3.Normalize\(Vector3\)](#), [Vector3.GetLength\(Vector3\)](#), [Vector3.AngleBetween\(Vector3, Vector3\)](#),  
[Vector3.OrthogonalityCheck\(Vector3, Vector3\)](#), [Vector3.Normalize\(\)](#), [Vector3.GetLength\(\)](#),  
[Vector3.CrossProduct\(Vector3\)](#), [Vector3.DotProduct\(Vector3\)](#), [Vector3.ScalarProduct\(float\)](#),  
[Vector3.Addition\(Vector3\)](#), [Vector3.Subtraction\(Vector3\)](#), [Vector3.Division\(float\)](#),  
[Vector3.IsPerpendicularTo\(Vector3\)](#), [Vector3.GetNormalVector\(int\)](#), [Vector3.IsZero\(\)](#),  
[Vector3.ToMatrix\(bool\)](#), [Vector3.Project\(Vector3\)](#), [Vector3.Project\(Plane\)](#), [Vector3.ToString\(\)](#),  
[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.ReferenceEquals\(object, object\)](#),  
[object.GetHashCode\(\)](#), [object.GetType\(\)](#), [object.MemberwiseClone\(\)](#)

## Constructors

### RotationVector3(float, float, float, RotationMeasure)

```
public RotationVector3(float x1, float x2, float x3, RotationMeasure measure)
```

## Parameters

x1 [float](#)

x2 [float](#)

x3 [float](#)

`measure RotationMeasure`

## Properties

### X1

X1 in RAD

```
public float X1 { get; set; }
```

Property Value

[float](#) ↗

### X2

X2 in RAD

```
public float X2 { get; set; }
```

Property Value

[float](#) ↗

### X3

X3 in RAD

```
public float X3 { get; set; }
```

Property Value

[float](#) ↗

## Methods

## GetDegreeRotation()

```
public Vector3 GetDegreeRotation()
```

Returns

[Vector3](#)

## SetRotation(float, Axis, RotationMeasure)

```
public void SetRotation(float f, Axis axis, RotationMeasure measure)
```

Parameters

f [float](#)

axis [Axis](#)

measure [RotationMeasure](#)

## SetRotation(float, float, float, RotationMeasure)

```
public void SetRotation(float x1, float x2, float x3, RotationMeasure measure)
```

Parameters

x1 [float](#)

x2 [float](#)

x3 [float](#)

measure [RotationMeasure](#)

## Operators

## operator +(RotationVector3, RotationVector3)

```
public static RotationVector3 operator +(RotationVector3 v1, RotationVector3 v2)
```

Parameters

v1 [RotationVector3](#)

v2 [RotationVector3](#)

Returns

[RotationVector3](#)

## implicit operator MatrixMxN(RotationVector3)

```
public static implicit operator MatrixMxN(RotationVector3 v)
```

Parameters

v [RotationVector3](#)

Returns

[MatrixMxN](#)

## implicit operator RotationMatrix3x3(RotationVector3)

Creates a [RotationMatrix3x3](#) if the given vector using the X1X2X3 convention.

```
public static implicit operator RotationMatrix3x3(RotationVector3 v)
```

Parameters

v [RotationVector3](#)

Returns

## RotationMatrix3x3

# Enum ShearConvention

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum ShearConvention
```

## Fields

AsymmetricCot = 3

AsymmetricTan = 2

SymmetricCot = 1

SymmetricTan = 0

# Class SheerMatrix2x2

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class SheerMatrix2x2
```

## Inheritance

[object](#) ← SheerMatrix2x2

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### SheerMatrix2x2(float, float, ShearConvention)

X and Y in radians for (the) tan or cot convention.

```
public SheerMatrix2x2(float X, float Y, ShearConvention convention)
```

## Parameters

X [float](#)

Y [float](#)

convention [ShearConvention](#)

## Fields

### S2x2

```
public readonly MatrixMxN S2x2
```

Field Value

[MatrixMxN](#)

## Operators

implicit operator MatrixMxN(SheerMatrix2x2)

```
public static implicit operator MatrixMxN(SheerMatrix2x2 sm)
```

Parameters

sm [SheerMatrix2x2](#)

Returns

[MatrixMxN](#)

# Class SheerMatrix3D

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class SheerMatrix3D
```

## Inheritance

[object](#) ← SheerMatrix3D

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

**SheerMatrix3D(ShearConvention, params float[])**

All angles  $\theta$  are in radians.

```
public SheerMatrix3D(ShearConvention convention, params float[] θ)
```

## Parameters

**convention** [ShearConvention](#)

**θ** [float](#)[]

## Fields

**S3x3**

3x3 Submatrix

```
public Matrix3x3 S3x3
```

Field Value

[Matrix3x3](#)

S4x4

```
public MatrixMxN S4x4
```

Field Value

[MatrixMxN](#)

## Methods

Multiply(SheerMatrix3D, Vector3)

```
public Vector3 Multiply(SheerMatrix3D s, Vector3 v)
```

Parameters

s [SheerMatrix3D](#)

v [Vector3](#)

Returns

[Vector3](#)

## Operators

implicit operator Matrix3x3(SheerMatrix3D)

```
public static implicit operator Matrix3x3(SheerMatrix3D sm)
```

Parameters

`sm` [SheerMatrix3D](#)

Returns

[Matrix3x3](#)

implicit operator MatrixMxN(SheerMatrix3D)

```
public static implicit operator MatrixMxN(SheerMatrix3D sm)
```

Parameters

`sm` [SheerMatrix3D](#)

Returns

[MatrixMxN](#)

operator \*(SheerMatrix3D, Vector3)

```
public static Vector operator *(SheerMatrix3D s, Vector3 v)
```

Parameters

`s` [SheerMatrix3D](#)

`v` [Vector3](#)

Returns

[Vector](#)

# Class Straight3D

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

This code defines a class called Straight3D that represents a straight line in 3D space. It has properties such as OA, OB, and Dir, which represent the endpoints of the line, and the direction of the line, respectively. The class also contains methods such as GetPointAt, Intersection, Distance, and Distance with 2 output parameters, which return the point on the line at a given position, determine if the line intersects with another line, find the distance between the line and a point in 3D space, and find the distance between the line and another line in 3D space along with the closest points on both lines to the point of intersection, respectively. The class also has some static fields, such as x1\_Axis, x2\_Axis, and x3\_Axis, which represent the x, y, and z axes in 3D space, respectively.

```
public class Straight3D
```

## Inheritance

[object](#) ← Straight3D

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) ,  
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#)

## Constructors

### Straight3D(Vector3, Vector3, LineSetupType)

```
public Straight3D(Vector3 OA, Vector3 OB, Straight3D.LineSetupType setupType)
```

## Parameters

OA [Vector3](#)

OB [Vector3](#)

setupType [Straight3D.LineSetupType](#)

## Fields

### x1\_Axis

```
public static readonly Straight3D x1_Axis
```

#### Field Value

[Straight3D](#)

### x2\_Axis

```
public static readonly Straight3D x2_Axis
```

#### Field Value

[Straight3D](#)

### x3\_Axis

```
public static readonly Straight3D x3_Axis
```

#### Field Value

[Straight3D](#)

## Properties

### Dir

```
public Vector3 Dir { get; }
```

#### Property Value

[Vector3](#)

## OA

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

Property Value

[Vector3](#)

## OB

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

Property Value

[Vector3](#)

## Methods

### Distance(Straight3D)

```
public float Distance(Straight3D s)
```

Parameters

s [Straight3D](#)

Returns

[float](#)

### Distance(Straight3D, Straight3D, out Vector3, out Vector3)

```
public static float Distance(Straight3D g, Straight3D h, out Vector3 OG, out Vector3 OH)
```

Parameters

g [Straight3D](#)

h [Straight3D](#)

OG [Vector3](#)

OH [Vector3](#)

Returns

[float](#) ↗

## Distance(Straight3D, out Vector3, out Vector3)

```
public float Distance(Straight3D s, out Vector3 OG, out Vector3 OH)
```

Parameters

s [Straight3D](#)

OG [Vector3](#)

OH [Vector3](#)

Returns

[float](#) ↗

## Distance(Vector3, Straight3D, out Vector3)

```
public static float Distance(Vector3 a, Straight3D b, out Vector3 l)
```

Parameters

a [Vector3](#)

b [Straight3D](#)

l [Vector3](#)

Returns

[float](#)

## Distance(Vector3, out Vector3)

```
public float Distance(Vector3 x, out Vector3 l)
```

Parameters

x [Vector3](#)

l [Vector3](#)

Returns

[float](#)

## GetPointAt(float)

```
public Vector3 GetPointAt(float f)
```

Parameters

f [float](#)

Returns

[Vector3](#)

## Intersection(Straight3D, Straight3D, out Vector3)

```
public static bool Intersection(Straight3D a, Straight3D b, out Vector3 I)
```

Parameters

a [Straight3D](#)

b [Straight3D](#)

I [Vector3](#)

Returns

[bool](#)

**ToString()**

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

Returns

[string](#)

# Enum Straight3D.LineSetupType

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum Straight3D.LineSetupType
```

## Fields

\_1Point1Dir = 1

\_2Points = 0

# Enum TransformationConvention

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public enum TransformationConvention
```

## Fields

DRS = 0

DSR = 2

RDS = 1

RSD = 3

SDR = 5

SRD = 4

# Class Vector

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

A vector with n dimensions

```
public class Vector : IEnumerable<float>, IEnumerable
```

Inheritance

[object](#) ← Vector

Implements

[IEnumerable](#) <[float](#)>, [IEnumerable](#)

Inherited Members

[object.ToString\(\)](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),  
[object.MemberwiseClone\(\)](#)

## Constructors

Vector(Vector)

```
public Vector(Vector v)
```

Parameters

v [Vector](#)

Vector(List<float>)

```
public Vector(List<float> values)
```

Parameters

values [List](#)<[float](#)>

## Vector(params float[])

public [Vector](#)(params float[] values)

Parameters

values [float](#)[]

## Properties

### Dimension

public int Dimension { get; }

Property Value

[int](#)

### this[int]

public float this[int index] { get; set; }

Parameters

index [int](#)

Property Value

[float](#)

### Length

```
public float Length { get; }
```

Property Value

[float](#)

## LengthSquared

```
public float LengthSquared { get; }
```

Property Value

[float](#)

## Normalized

```
public Vector Normalized { get; }
```

Property Value

[Vector](#)

## Methods

### AddValue(float)

Adds a dimension to the vector and adds the given value.

```
public void AddValue(float value)
```

Parameters

**value** [float](#)

## Addition(Vector)

```
public Vector Addition(Vector vector)
```

Parameters

vector [Vector](#)

Returns

[Vector](#)

## Addition(Vector, Vector)

```
public static Vector Addition(Vector a, Vector b)
```

Parameters

a [Vector](#)

b [Vector](#)

Returns

[Vector](#)

## Cross(Vector, Vector)

```
public static Vector Cross(Vector a, Vector b)
```

Parameters

a [Vector](#)

b [Vector](#)

Returns

## [Vector](#)

### CrossProduct(Vector)

```
public Vector CrossProduct(Vector b)
```

#### Parameters

b [Vector](#)

#### Returns

[Vector](#)

### Division(Vector, float)

```
public static Vector Division(Vector a, float f)
```

#### Parameters

a [Vector](#)

f [float](#)

#### Returns

[Vector](#)

### Division(float)

```
public Vector Division(float scalar)
```

#### Parameters

scalar [float](#)

Returns

[Vector](#)

## Dot(Vector, Vector)

```
public static float Dot(Vector a, Vector b)
```

Parameters

a [Vector](#)

b [Vector](#)

Returns

[float](#) ↗

## DotProduct(Vector)

```
public float DotProduct(Vector vector)
```

Parameters

vector [Vector](#)

Returns

[float](#) ↗

## GetEnumerator()

```
public IEnumerator<float> GetEnumerator()
```

Returns

[IEnumerator](#) <[float](#)>

## GetLength()

```
public float GetLength()
```

Returns

[float](#)

## GetValue(int)

```
public float GetValue(int index)
```

Parameters

index [int](#)

Returns

[float](#)

## IsDimensionEqual(Vector, Vector)

```
public bool IsDimensionEqual(Vector a, Vector b)
```

Parameters

a [Vector](#)

b [Vector](#)

Returns

[bool](#)

## IsValidIndex(int)

Checks if the vector contains the given index.

```
public bool IsValidIndex(int index)
```

Parameters

`index` [int](#)

Returns

[bool](#)

## IsZeroVector()

```
public bool IsZeroVector()
```

Returns

[bool](#)

## Normalize()

```
public Vector Normalize()
```

Returns

[Vector](#)

## Scalar(Vector, float)

```
public static Vector Scalar(Vector a, float f)
```

Parameters

a [Vector](#)

f [float](#)

Returns

[Vector](#)

## ScalarProduct(float)

```
public Vector ScalarProduct(float scalar)
```

Parameters

scalar [float](#)

Returns

[Vector](#)

## SetValue(float, int)

```
public void SetValue(float value, int index)
```

Parameters

value [float](#)

index [int](#)

## Substraction(Vector)

```
public Vector Substraction(Vector vector)
```

Parameters

`vector` [Vector](#)

Returns

[Vector](#)

## Subtraction(Vector, Vector)

```
public static Vector Subtraction(Vector a, Vector b)
```

Parameters

`a` [Vector](#)

`b` [Vector](#)

Returns

[Vector](#)

## ToMatrix(Vector, bool)

Turns the vector into single-row matrix, as long as `isColumn` is false. Otherwise it turns the vector into a single-column matrix

```
public static MatrixMxN ToMatrix(Vector v, bool isColumn = false)
```

Parameters

`v` [Vector](#)

`isColumn` [bool](#) ↗

Returns

[MatrixMxN](#)

# Operators

## operator +(Vector, Vector)

```
public static Vector operator +(Vector v1, Vector v2)
```

### Parameters

v1 [Vector](#)

v2 [Vector](#)

### Returns

[Vector](#)

## operator /(Vector, float)

```
public static Vector operator /(Vector v1, float f)
```

### Parameters

v1 [Vector](#)

f [float](#)

### Returns

[Vector](#)

## operator \*(Vector, Vector)

```
public static float operator *(Vector v1, Vector v2)
```

### Parameters

v1 [Vector](#)

v2 [Vector](#)

Returns

[float](#) ↗

operator \*(Vector, float)

```
public static Vector operator *(Vector v1, float f)
```

Parameters

v1 [Vector](#)

f [float](#) ↗

Returns

[Vector](#)

operator \*(float, Vector)

```
public static Vector operator *(float f, Vector v1)
```

Parameters

f [float](#) ↗

v1 [Vector](#)

Returns

[Vector](#)

operator -(Vector, Vector)

```
public static Vector operator -(Vector v1, Vector v2)
```

Parameters

v1 [Vector](#)

v2 [Vector](#)

Returns

[Vector](#)

# Class Vector2

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class Vector2
```

## Inheritance

[object](#) ← Vector2

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### Vector2(Vector2)

```
public Vector2(Vector2 vector)
```

#### Parameters

vector [Vector2](#)

### Vector2(float, float)

```
public Vector2(float x, float y)
```

#### Parameters

x [float](#)

y [float](#)

# Fields

## One

```
public static readonly Vector2 One
```

### Field Value

[Vector2](#)

## Zero

```
public static readonly Vector2 Zero
```

### Field Value

[Vector2](#)

# Properties

## Dimension

```
public int Dimension { get; }
```

### Property Value

[int](#)

## this[int]

```
public float this[int xindex] { get; set; }
```

### Parameters

xindex [int](#)

Property Value

[float](#)

## Length

```
public float Length { get; }
```

Property Value

[float](#)

## LengthSquared

```
public float LengthSquared { get; }
```

Property Value

[float](#)

## Normalized

```
public Vector2 Normalized { get; }
```

Property Value

[Vector2](#)

## X

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

Property Value

[float](#) ↗

Y

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

Property Value

[float](#) ↗

## Methods

Addition(Vector2)

```
public Vector2 Addition(Vector2 vector)
```

Parameters

vector [Vector2](#)

Returns

[Vector2](#)

AngleBetween(Vector2, Vector2)

```
public static float AngleBetween(Vector2 a, Vector2 b)
```

Parameters

a [Vector2](#)

b [Vector2](#)

Returns

[float](#)

## CrossProduct(Vector2, Vector2, out Vector3)

```
public static float CrossProduct(Vector2 u, Vector2 v, out Vector3 resultAsVector3)
```

Parameters

u [Vector2](#)

v [Vector2](#)

resultAsVector3 [Vector3](#)

Returns

[float](#)

## CrossProduct(Vector2, out Vector3)

```
public float CrossProduct(Vector2 v, out Vector3 resultAsVector3)
```

Parameters

v [Vector2](#)

resultAsVector3 [Vector3](#)

Returns

[float](#)

## Division(float)

```
public Vector2 Division(float scalar)
```

Parameters

scalar [float](#)

Returns

[Vector2](#)

## DotProduct(Vector2)

```
public float DotProduct(Vector2 v)
```

Parameters

v [Vector2](#)

Returns

[float](#)

## DotProduct(Vector2, Vector2)

```
public static float DotProduct(Vector2 u, Vector2 v)
```

Parameters

u [Vector2](#)

v [Vector2](#)

Returns

[float](#)

## GetLength()

```
public float GetLength()
```

Returns

[float](#)

## GetLength(Vector2)

```
public static float GetLength(Vector2 v)
```

Parameters

v [Vector2](#)

Returns

[float](#)

## GetNormalVector(bool)

```
public Vector2 GetNormalVector(bool negateYEntryInstead)
```

Parameters

negateYEntryInstead [bool](#)

Returns

[Vector2](#)

## IsPerpendicularTo(Vector2)

```
public bool IsPerpendicularTo(Vector2 v)
```

Parameters

v [Vector2](#)

Returns

[bool](#)

**IsZero()**

```
public bool IsZero()
```

Returns

[bool](#)

**Normalize()**

```
public Vector2 Normalize()
```

Returns

[Vector2](#)

**Normalize(Vector2)**

```
public static Vector2 Normalize(Vector2 v)
```

Parameters

v [Vector2](#)

Returns

[Vector2](#)

## OrthogonalityCheck(Vector2, Vector2)

```
public static bool OrthogonalityCheck(Vector2 a, Vector2 b)
```

Parameters

a [Vector2](#)

b [Vector2](#)

Returns

[bool](#) ↗

## ScalarProduct(float)

```
public Vector2 ScalarProduct(float scalar)
```

Parameters

scalar [float](#) ↗

Returns

[Vector2](#)

## Substraction(Vector2)

```
public Vector2 Substraction(Vector2 v)
```

Parameters

v [Vector2](#)

Returns

[Vector2](#)

## ToMatrix(Vector2, bool)

Turns the vector into single-row matrix, as long as `isColumn` is false. Otherwise it turns the vector into a single-column matrix

```
public static MatrixMxN ToMatrix(Vector2 v, bool isColumn = false)
```

Parameters

`v` [Vector2](#)

`isColumn` [bool](#)

Returns

[MatrixMxN](#)

## ToMatrix(bool)

```
public MatrixMxN ToMatrix(bool isColumn = false)
```

Parameters

`isColumn` [bool](#)

Returns

[MatrixMxN](#)

## Operators

### operator +(Vector2, Vector2)

```
public static Vector2 operator +(Vector2 v1, Vector2 v2)
```

Parameters

v1 [Vector2](#)

v2 [Vector2](#)

Returns

[Vector2](#)

operator /(Vector2, float)

```
public static Vector2 operator /(Vector2 v, float f)
```

Parameters

v [Vector2](#)

f [float](#)

Returns

[Vector2](#)

implicit operator Vector(Vector2)

```
public static implicit operator Vector(Vector2 v)
```

Parameters

v [Vector2](#)

Returns

[Vector](#)

operator \*(Vector2, Vector2)

```
public static float operator *(Vector2 v1, Vector2 v2)
```

Parameters

v1 [Vector2](#)

v2 [Vector2](#)

Returns

[float](#)

## operator \*(Vector2, float)

```
public static Vector2 operator *(Vector2 v, float f)
```

Parameters

v [Vector2](#)

f [float](#)

Returns

[Vector2](#)

## operator \*(float, Vector2)

```
public static Vector2 operator *(float f, Vector2 v)
```

Parameters

f [float](#)

v [Vector2](#)

Returns

## [Vector2](#)

### operator -(Vector2, Vector2)

```
public static Vector2 operator -(Vector2 v1, Vector2 v2)
```

#### Parameters

v1 [Vector2](#)

v2 [Vector2](#)

#### Returns

[Vector2](#)

# Class Vector3

Namespace: [Engine.Core.Maths](#)

Assembly: Engine.dll

```
public class Vector3
```

## Inheritance

[object](#) ← Vector3

## Derived

[RotationVector3](#)

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) ,  
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#)

# Constructors

## Vector3(Vector3)

```
public Vector3(Vector3 vector)
```

### Parameters

vector [Vector3](#)

## Vector3(float, float, float)

```
public Vector3(float x1, float x2, float x3)
```

### Parameters

x1 [float](#)

x2 [float](#)

x3 [float](#)

## Fields

### One

```
public static readonly Vector3 One
```

#### Field Value

[Vector3](#)

### Zero

```
public static readonly Vector3 Zero
```

#### Field Value

[Vector3](#)

## Properties

### Dimension

```
public int Dimension { get; }
```

#### Property Value

[int](#)

### this[int]

```
public float this[int xindex] { get; set; }
```

## Parameters

xindex [int](#)

## Property Value

[float](#)

## Length

```
public float Length { get; }
```

## Property Value

[float](#)

## LengthSquared

```
public float LengthSquared { get; }
```

## Property Value

[float](#)

## Normalized

```
public Vector3 Normalized { get; }
```

## Property Value

[Vector3](#)

## X1

```
public float X1 { get; set; }
```

Property Value

[float](#)

X2

```
public float X2 { get; set; }
```

Property Value

[float](#)

X3

```
public float X3 { get; set; }
```

Property Value

[float](#)

## Methods

Addition(Vector3)

```
public Vector3 Addition(Vector3 vector)
```

Parameters

vector [Vector3](#)

Returns

## [Vector3](#)

### AngleBetween(Vector3, Vector3)

```
public static float AngleBetween(Vector3 a, Vector3 b)
```

#### Parameters

a [Vector3](#)

b [Vector3](#)

#### Returns

[float](#) ↗

### CrossProduct(Vector3)

```
public Vector3 CrossProduct(Vector3 v)
```

#### Parameters

v [Vector3](#)

#### Returns

[Vector3](#)

### CrossProduct(Vector3, Vector3)

```
public static Vector3 CrossProduct(Vector3 u, Vector3 v)
```

#### Parameters

u [Vector3](#)

v [Vector3](#)

Returns

[Vector3](#)

## Division(float)

```
public Vector3 Division(float scalar)
```

Parameters

scalar [float](#)

Returns

[Vector3](#)

## DotProduct(Vector3)

```
public float DotProduct(Vector3 v)
```

Parameters

v [Vector3](#)

Returns

[float](#)

## DotProduct(Vector3, Vector3)

```
public static float DotProduct(Vector3 u, Vector3 v)
```

Parameters

u [Vector3](#)

v [Vector3](#)

Returns

[float](#) ↗

## GetLength()

```
public float GetLength()
```

Returns

[float](#) ↗

## GetLength(Vector3)

```
public static float GetLength(Vector3 v)
```

Parameters

v [Vector3](#)

Returns

[float](#) ↗

## GetNormalVector(int)

`zeroAt` ∈ [1;3]; everything else results in a null vector.

```
public Vector3 GetNormalVector(int zeroAt)
```

Parameters

`zeroAt` `int`

Returns

[Vector3](#)

## IsPerpendicularTo(Vector3)

```
public bool IsPerpendicularTo(Vector3 v)
```

Parameters

`v` [Vector3](#)

Returns

`bool`

## IsZero()

```
public bool IsZero()
```

Returns

`bool`

## Normalize()

```
public Vector3 Normalize()
```

Returns

[Vector3](#)

## Normalize(Vector3)

```
public static Vector3 Normalize(Vector3 v)
```

Parameters

v [Vector3](#)

Returns

[Vector3](#)

## OrthogonalityCheck(Vector3, Vector3)

```
public static bool OrthogonalityCheck(Vector3 a, Vector3 b)
```

Parameters

a [Vector3](#)

b [Vector3](#)

Returns

[bool](#)

## Project(Plane)

```
public Vector3 Project(Plane to)
```

Parameters

to [Plane](#)

Returns

[Vector3](#)

## Project(Vector3)

```
public Vector3 Project(Vector3 to)
```

Parameters

to [Vector3](#)

Returns

[Vector3](#)

## ScalarProduct(float)

```
public Vector3 ScalarProduct(float scalar)
```

Parameters

scalar [float](#)

Returns

[Vector3](#)

## Substraction(Vector3)

```
public Vector3 Substraction(Vector3 v)
```

Parameters

v [Vector3](#)

Returns

## [Vector3](#)

### ToMatrix(Vector3, bool)

Turns the vector into single-row matrix, as long as `isColumn` is false. Otherwise it turns the vector into a single-column matrix

```
public static MatrixMxN ToMatrix(Vector3 v, bool isColumn = false)
```

Parameters

`v` [Vector3](#)

`isColumn` [bool](#)

Returns

[MatrixMxN](#)

### ToMatrix(bool)

```
public MatrixMxN ToMatrix(bool isColumn = false)
```

Parameters

`isColumn` [bool](#)

Returns

[MatrixMxN](#)

### ToString()

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

Returns

[string](#)

## Operators

### operator +(Vector3, Vector3)

```
public static Vector3 operator +(Vector3 v1, Vector3 v2)
```

Parameters

v1 [Vector3](#)

v2 [Vector3](#)

Returns

[Vector3](#)

### operator /(Vector3, float)

```
public static Vector3 operator /(Vector3 v, float f)
```

Parameters

v [Vector3](#)

f [float](#)

Returns

[Vector3](#)

### implicit operator Vector(Vector3)

```
public static implicit operator Vector(Vector3 v)
```

Parameters

v [Vector3](#)

Returns

[Vector](#)

## operator \*(Vector3, Vector3)

```
public static float operator *(Vector3 v1, Vector3 v2)
```

Parameters

v1 [Vector3](#)

v2 [Vector3](#)

Returns

[float](#)

## operator \*(Vector3, float)

```
public static Vector3 operator *(Vector3 v, float f)
```

Parameters

v [Vector3](#)

f [float](#)

Returns

[Vector3](#)

## operator \*(float, Vector3)

```
public static Vector3 operator *(float f, Vector3 v)
```

Parameters

f [float](#)

v [Vector3](#)

Returns

[Vector3](#)

## operator -(Vector3, Vector3)

```
public static Vector3 operator -(Vector3 v1, Vector3 v2)
```

Parameters

v1 [Vector3](#)

v2 [Vector3](#)

Returns

[Vector3](#)

# Namespace Engine.Core.Maths.Obsolete Classes

[Gaussian](#)

[Gaussian.GaussianResult](#)

[Matrix](#)

# Class Gaussian

Namespace: [Engine.Core.Maths.Obsolete](#)

Assembly: Engine.dll

```
[Obsolete]  
public class Gaussian
```

## Inheritance

[object](#) ← Gaussian

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Methods

## Elimination(Matrix, Matrix, out GaussianResult)

```
public static bool Elimination(Matrix matrix, Matrix augmentation, out  
Gaussian.GaussianResult result)
```

### Parameters

matrix [Matrix](#)

augmentation [Matrix](#)

result [Gaussian.GaussianResult](#)

### Returns

[bool](#)

# Class Gaussian.GaussianResult

Namespace: [Engine.Core.Maths.Obsolete](#)

Assembly: Engine.dll

```
[Obsolete]  
public class Gaussian.GaussianResult
```

## Inheritance

[object](#) ← Gaussian.GaussianResult

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Fields

### plane

```
public Plane plane
```

#### Field Value

[Plane](#)

### results

```
public List<float> results
```

#### Field Value

[List](#) <[float](#)>

# straight

```
public Straight3D straight
```

Field Value

[Straight3D](#)

# Class Matrix

Namespace: [Engine.Core.Maths.Obsolete](#)

Assembly: Engine.dll

```
[Obsolete("Deprecated")]
public class Matrix
```

## Inheritance

[object](#) ← Matrix

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#)

# Constructors

## Matrix(List<List<float> >)

```
public Matrix(List<List<float>> values)
```

### Parameters

values [List](#)<[List](#)<[float](#)>>

## Matrix(int, int)

```
public Matrix(int rowCount, int columnCount)
```

### Parameters

rowCount [int](#)

columnCount [int](#)

# Properties

## ColumnCount

```
public int ColumnCount { get; }
```

### Property Value

[int ↗](#)

## this[int, int]

```
public float this[int row, int column] { get; set; }
```

### Parameters

row [int ↗](#)

column [int ↗](#)

### Property Value

[float ↗](#)

## RowCount

```
public int RowCount { get; }
```

### Property Value

[int ↗](#)

## Values

```
public List<List<float>> Values { get; set; }
```

Property Value

`List<List<float>>`

## Methods

AddRow(List<float>)

`public void AddRow(List<float> row)`

Parameters

`row List<float>`

AddRow(int, int, float)

`public void AddRow(int targetRow, int sourceRow, float factor)`

Parameters

`targetRow int`

`sourceRow int`

`factor float`

ConvertTo(List<List<float>>)

`public static Matrix ConvertTo(List<List<float>> matrix)`

Parameters

`matrix List<List<float>>`

Returns

## Matrix

### GetColumn(int)

```
public List<float> GetColumn(int index)
```

Parameters

index [int](#)

Returns

[List](#) <[float](#)>

### GetRow(int)

```
public List<float> GetRow(int index)
```

Parameters

index [int](#)

Returns

[List](#) <[float](#)>

### ScaleRow(int, float)

```
public void ScaleRow(int index, float scale)
```

Parameters

index [int](#)

scale [float](#)

## SwapRows(int, int)

```
public void SwapRows(int from, int to)
```

### Parameters

from [int](#)

to [int](#)

## Operators

### operator +(Matrix, Matrix)

```
public static Matrix operator +(Matrix a, Matrix b)
```

### Parameters

a [Matrix](#)

b [Matrix](#)

### Returns

[Matrix](#)

### operator \*(Matrix, Matrix)

```
public static Matrix operator *(Matrix a, Matrix b)
```

### Parameters

a [Matrix](#)

b [Matrix](#)

### Returns

## [Matrix](#)

### operator \*(Matrix, float)

```
public static Matrix operator *(Matrix b, float a)
```

#### Parameters

b [Matrix](#)

a [float](#)

#### Returns

[Matrix](#)

### operator \*(float, Matrix)

```
public static Matrix operator *(float a, Matrix b)
```

#### Parameters

a [float](#)

b [Matrix](#)

#### Returns

[Matrix](#)

### operator -(Matrix, Matrix)

```
public static Matrix operator -(Matrix a, Matrix b)
```

#### Parameters

a [Matrix](#)

b [Matrix](#)

Returns

[Matrix](#)

# Namespace Engine.Core.Physics.Collision Detection

## Classes

[BoundingBox3D](#)

[BoundingVolume3D](#)

[Cell](#)

[Grid](#)

[SpatialPartitioning](#)

## Structs

[ParticleCollision3D](#)

# Class BoundingBox3D

Namespace: [Engine.Core.Physics.CollisionDetection](#)

Assembly: Engine.dll

```
public class BoundingBox3D
```

## Inheritance

[object](#) ← BoundingBox3D

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### BoundingBox3D(BoundingBox3D)

```
public BoundingBox3D(BoundingBox3D box)
```

#### Parameters

box [BoundingBox3D](#)

### BoundingBox3D(float, float, float, float, float, float)

```
public BoundingBox3D(float maxX1, float maxX2, float maxX3, float minX1, float minX2,  
float minX3)
```

#### Parameters

maxX1 [float](#)

maxX2 [float](#)

maxX3 [float](#)

minX1 [float](#)

minX2 [float](#)

minX3 [float](#)

## Fields

### maxX1

`public float maxX1`

Field Value

[float](#)

### maxX2

`public float maxX2`

Field Value

[float](#)

### maxX3

`public float maxX3`

Field Value

[float](#)

### minX1

```
public float minX1
```

Field Value

[float](#) ↗

minX2

```
public float minX2
```

Field Value

[float](#) ↗

minX3

```
public float minX3
```

Field Value

[float](#) ↗

## Properties

BoundingDepth

```
public int BoundingDepth { get; }
```

Property Value

[int](#) ↗

## Methods

## Contains(Vector3)

Checks if the position vector is within the boundaries.

```
public bool Contains(Vector3 globalPosition)
```

Parameters

globalPosition [Vector3](#)

Returns

[bool](#)

True if the position is within the boundaries; else false

## GetBoundingBox(Mesh3D)

```
public static BoundingBox3D GetBoundingBox(Mesh3D mesh)
```

Parameters

mesh [Mesh3D](#)

Returns

[BoundingBox3D](#)

## Operators

### implicit operator BoundingBox3D(Mesh3D)

```
public static implicit operator BoundingBox3D(Mesh3D m)
```

Parameters

m [Mesh3D](#)

Returns

[BoundingBox3D](#)

# Class BoundingVolume3D

Namespace: [Engine.Core.Physics.CollisionDetection](#)

Assembly: Engine.dll

```
public class BoundingVolume3D
```

## Inheritance

[object](#) ← BoundingVolume3D

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### BoundingVolume3D(List<BoundingBox3D>)

```
public BoundingVolume3D(List<BoundingBox3D> boxes)
```

#### Parameters

boxes [List](#)<[BoundingBox3D](#)>

### BoundingVolume3D(List<BoundingVolume3D>)

```
public BoundingVolume3D(List<BoundingVolume3D> volumes)
```

#### Parameters

volumes [List](#)<[BoundingVolume3D](#)>

## BoundingVolume3D(List<BoundingVolume3D>, List<BoundingBox3D>)

```
public BoundingVolume3D(List<BoundingVolume3D> volumes, List<BoundingBox3D> boxes)
```

### Parameters

volumes [List](#)<[BoundingVolume3D](#)>

boxes [List](#)<[BoundingBox3D](#)>

## Fields

### boxes

```
public List<BoundingBox3D> boxes
```

### Field Value

[List](#)<[BoundingBox3D](#)>

### volumes

```
public List<BoundingVolume3D> volumes
```

### Field Value

[List](#)<[BoundingVolume3D](#)>

## Properties

### BoundingDepth

```
public int BoundingDepth { get; }
```

Property Value

[int ↗](#)

## Bounds

`public BoundingBox3D Bounds { get; }`

Property Value

[BoundingBox3D](#)

# Class Cell

Namespace: [Engine.Core.Physics.CollisionDetection](#)

Assembly: Engine.dll

```
public class Cell
```

## Inheritance

[object](#) ← Cell

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## Cell(Vector2)

```
public Cell(Vector2 index)
```

### Parameters

index [Vector2](#)

## Cell(Vector2, Cell)

```
public Cell(Vector2 index, Cell cell)
```

### Parameters

index [Vector2](#)

cell [Cell](#)

# Fields

## meshes

```
public List<Mesh3D> meshes
```

## Field Value

[List](#) <[Mesh3D](#)>

# Properties

## Index

```
public Vector2 Index { get; }
```

## Property Value

[Vector2](#)

# Methods

## Contains(Mesh3D)

```
public bool Contains(Mesh3D mesh)
```

## Parameters

mesh [Mesh3D](#)

## Returns

[bool](#)

# Class Grid

Namespace: [Engine.Core.Physics.CollisionDetection](#)

Assembly: Engine.dll

```
public class Grid
```

## Inheritance

[object](#) ← Grid

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

Grid()

```
public Grid()
```

# Properties

this[Vector2]

```
public Cell this[Vector2 index] { get; set; }
```

## Parameters

index [Vector2](#)

Property Value

[Cell](#)

# Methods

## AddMeshToCell(Vector2, Mesh3D)

```
public void AddMeshToCell(Vector2 index, Mesh3D mesh)
```

### Parameters

index [Vector2](#)

mesh [Mesh3D](#)

## CellContains(Vector2, Mesh3D)

```
public bool CellContains(Vector2 index, Mesh3D mesh)
```

### Parameters

index [Vector2](#)

mesh [Mesh3D](#)

### Returns

[bool](#)

## Contains(Vector2)

```
public bool Contains(Vector2 index)
```

### Parameters

index [Vector2](#)

### Returns

[bool](#)

## GetCell(Vector2)

```
public Cell GetCell(Vector2 index)
```

Parameters

index [Vector2](#)

Returns

[Cell](#)

## GetMeshesInCell(Vector2)

```
public List<Mesh3D> GetMeshesInCell(Vector2 index)
```

Parameters

index [Vector2](#)

Returns

[List](#)<[Mesh3D](#)>

## GetMeshesInCells(params Vector2[])

```
public List<Mesh3D> GetMeshesInCells(params Vector2[] indexes)
```

Parameters

indexes [Vector2\[\]](#)

Returns

## RemoveMeshFromAllCells(Mesh3D)

```
public void RemoveMeshFromAllCells(Mesh3D mesh)
```

### Parameters

mesh [Mesh3D](#)

## RemoveMeshFromCell(Vector2, Mesh3D)

```
public void RemoveMeshFromCell(Vector2 index, Mesh3D mesh)
```

### Parameters

index [Vector2](#)

mesh [Mesh3D](#)

## SetCell(Vector2, Cell)

```
public Cell SetCell(Vector2 index, Cell value)
```

### Parameters

index [Vector2](#)

value [Cell](#)

### Returns

[Cell](#)

# Struct ParticleCollision3D

Namespace: [Engine.Core.Physics.CollisionDetection](#)

Assembly: Engine.dll

```
public readonly struct ParticleCollision3D
```

## Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetType\(\)](#)

## Constructors

ParticleCollision3D(Vector3, Object3D, Mesh3D, Face3D)

```
public ParticleCollision3D(Vector3 at, Object3D object3D, Mesh3D meshHit, Face3D faceHit)
```

## Parameters

at [Vector3](#)

object3D [Object3D](#)

meshHit [Mesh3D](#)

faceHit [Face3D](#)

## Properties

### At

Global position of the collision

```
public Vector3 At { get; }
```

### Property Value

## [Vector3](#)

### Face

The specific face of the mesh that was involved in the collision

```
public Face3D Face { get; }
```

Property Value

[Face3D](#)

### MeshHit

The mesh that was involved in the collision

```
public Mesh3D MeshHit { get; }
```

Property Value

[Mesh3D](#)

### Object

```
public Object3D Object { get; }
```

Property Value

[Object3D](#)

# Class SpatialPartitioning

Namespace: [Engine.Core.Physics.CollisionDetection](#)

Assembly: Engine.dll

```
public class SpatialPartitioning
```

## Inheritance

[object](#) ← SpatialPartitioning

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### SpatialPartitioning()

```
public SpatialPartitioning()
```

## Properties

### CellSize

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

Property Value

[float](#)

## Methods

### GetMeshesAt(Vector3)

Projects **point** onto the partitioning grid.

```
public List<Mesh3D> GetMeshesAt(Vector3 point)
```

Parameters

**point** [Vector3](#)

Returns

[List](#) <[Mesh3D](#)>

Return all meshes within the cell, the point's in

## MapMesh(Mesh3D)

```
public void MapMesh(Mesh3D mesh)
```

Parameters

**mesh** [Mesh3D](#)

## RemoveMesh(Mesh3D)

```
public void RemoveMesh(Mesh3D mesh)
```

Parameters

**mesh** [Mesh3D](#)

## UpdateMesh(Mesh3D)

```
public void UpdateMesh(Mesh3D mesh)
```

Parameters

`mesh` [Mesh3D](#)

# Namespace Engine.Core.Physics.Optics

## Classes

[Lens](#)

[LightRay](#)

[LightSource](#)

[Photon](#)

## Enums

[LensType](#)

# Class Lens

Namespace: [Engine.Core.Physics.Optics](#)

Assembly: Engine.dll

```
public class Lens
```

## Inheritance

[object](#) ← Lens

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

Lens()

```
public Lens()
```

# Properties

FovX

```
public float FovX { get; }
```

Property Value

[float](#)

FovXRad

```
public float FovXRad { get; }
```

Property Value

[float](#)

## FovY

```
public float FovY { get; }
```

Property Value

[float](#)

## FovYRad

```
public float FovYRad { get; }
```

Property Value

[float](#)

## Type

```
public LensType Type { get; }
```

Property Value

[LensType](#)

## Methods

CreateRay(float, float, Camera, out float)

Creates the ray from the image plane to the near plane, and outputs a float "r" defined for when the ray hits the nearplane.

```
public Straight3D CreateRay(float s, float t, Camera cam, out float r)
```

## Parameters

s [float](#)

t [float](#)

cam [Camera](#)

r [float](#)

## Returns

[Straight3D](#)

# Enum LensType

Namespace: [Engine.Core.Physics.Optics](#)

Assembly: Engine.dll

```
public enum LensType
```

## Fields

BackSided = 3

FrontSided = 2

None = 0

Prismatic = 4

TwoSided = 1

# Class LightRay

Namespace: [Engine.Core.Physics.Optics](#)

Assembly: Engine.dll

```
public class LightRay : Ray, IParticleHandler, IRay
```

## Inheritance

[object](#) ← [Ray](#) ← LightRay

## Implements

[IParticleHandler](#), [IRay](#)

## Inherited Members

[Ray.Particle](#) , [Ray.Emit\(float\)](#) , [Ray.EliminateParticle\(ref Particle\)](#) , [Ray.EliminateAllParticles\(\)](#) ,  
[Ray.OnRayCollision\(List<ParticleCollision3D>\)](#) , [object.ToString\(\)](#) , [object.Equals\(object\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,  
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#)

## Constructors

### LightRay(Straight3D, float)

```
public LightRay(Straight3D ray, float rayStart)
```

#### Parameters

ray [Straight3D](#)

rayStart [float](#)

### LightRay(Straight3D, float, float)

```
public LightRay(Straight3D ray, float rayStart, float samplingRate)
```

#### Parameters

`ray` [Straight3D](#)

`rayStart` [float](#)

`samplingRate` [float](#)

# Class LightSource

Namespace: [Engine.Core.Physics.Optics](#)

Assembly: Engine.dll

```
public class LightSource
```

## Inheritance

[object](#) ← LightSource

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Class Photon

Namespace: [Engine.Core.Physics.Optics](#)

Assembly: Engine.dll

```
public class Photon : Particle
```

## Inheritance

[object](#) ← [Particle](#) ← Photon

## Inherited Members

[Particle.Position](#) , [Particle.SetPosition\(Vector3\)](#) , [Particle.Update\(\)](#) , [Particle.GetCollisions\(\)](#) ,  
[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## Photon(Vector3)

```
public Photon(Vector3 position)
```

## Parameters

position [Vector3](#)

# Namespace Engine.Core.Rendering

## Classes

[CameraRay](#)

## Structs

[Pixel](#)

# Class CameraRay

Namespace: [Engine.Core.Rendering](#)

Assembly: Engine.dll

```
public class CameraRay
```

## Inheritance

[object](#) ← CameraRay

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Struct Pixel

Namespace: [Engine.Core.Rendering](#)

Assembly: Engine.dll

```
public struct Pixel
```

## Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetType\(\)](#)

## Fields

### color

```
public Color color
```

#### Field Value

[Color](#)

### X

```
public float x
```

#### Field Value

[float](#)

### y

```
public float y
```

## Field Value

[float](#) ↗

# Namespace Engine.Core.SceneManagement

## Classes

[Scene](#)

[Scene3D](#)

[SceneManager](#)

## Enums

[SceneType](#)

# Class Scene

Namespace: [Engine.Core.SceneManagement](#)

Assembly: Engine.dll

```
public class Scene
```

## Inheritance

[object](#) ← Scene

## Derived

[Scene3D](#)

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## Scene(int, SceneType)

```
public Scene(int sceneId, SceneType type)
```

## Parameters

sceneId [int](#)

type [SceneType](#)

# Fields

## Name

```
public string Name
```

Field Value

[string](#)

## Properties

SceneId

```
public int SceneId { get; }
```

Property Value

[int](#)

Type

```
public SceneType Type { get; }
```

Property Value

[SceneType](#)

## Methods

GetGlobalCoordinateSystem(out GCS3)

Gets the scene's global coordinate system. TODO: Overload method for GCS2

```
public bool GetGlobalCoordinateSystem(out GCS3 gcs)
```

Parameters

[gcs](#) [GCS3](#)

Returns

bool ↗

Returns whether there is GCS3 or not

# Class Scene3D

Namespace: [Engine.Core.SceneManagement](#)

Assembly: Engine.dll

```
public class Scene3D : Scene
```

## Inheritance

[object](#) ← [Scene](#) ← Scene3D

## Inherited Members

[Scene.Name](#) , [Scene.ScenId](#) , [Scene.Type](#) , [Scene.GetGlobalCoordinateSystem\(out GCS3\)](#) ,  
[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

Scene3D(int)

```
public Scene3D(int sceneId)
```

## Parameters

sceneId [int](#)

## Properties

GlobalCoordinateSystem

```
public GCS3 GlobalCoordinateSystem { get; }
```

## Property Value

[GCS3](#)

# Class SceneManager

Namespace: [Engine.Core.SceneManagement](#)

Assembly: Engine.dll

```
public class SceneManager
```

## Inheritance

[object](#) ← SceneManager

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

SceneManager()

```
public SceneManager()
```

## Fields

ActiveScene

```
public Scene ActiveScene
```

Field Value

[Scene](#)

Instance

```
public static SceneManager Instance
```

Field Value

[SceneManager](#)

## Methods

AddScene(Scene)

```
public void AddScene(Scene scene)
```

Parameters

scene [Scene](#)

# Enum SceneType

Namespace: [Engine.Core.SceneManagement](#)

Assembly: Engine.dll

```
public enum SceneType
```

## Fields

Scene2D = 0

Scene3D = 1

# Namespace Engine.Rendering

## Classes

[ColorExtension](#)

[RenderImage](#)

[RenderPoint](#)

[RenderTexture](#)

[RenderingConstants](#)

## Enums

[Projection](#)

# Class ColorExtension

Namespace: [Engine.Rendering](#)

Assembly: Engine.Rendering.dll

```
public static class ColorExtension
```

## Inheritance

[object](#) ← ColorExtension

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Methods

## Interpolate(Color, Color, float)

```
public static Color Interpolate(this Color c1, Color c2, float t)
```

### Parameters

c1 [Color](#)

c2 [Color](#)

t [float](#)

### Returns

[Color](#)

# Enum Projection

Namespace: [Engine.Rendering](#)

Assembly: Engine.Rendering.dll

```
public enum Projection
```

## Fields

Orthographic = 0

Perspective = 1

# Class RenderImage

Namespace: [Engine.Rendering](#)

Assembly: Engine.Rendering.dll

```
public class RenderImage
```

## Inheritance

[object](#) ← RenderImage

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### RenderImage(Projection, ref PictureBox)

```
public RenderImage(Projection projection, ref PictureBox pictureBox)
```

## Parameters

projection [Projection](#)

pictureBox [PictureBox](#)

## Methods

### Render()

```
public void Render()
```

# Class RenderPoint

Namespace: [Engine.Rendering](#)

Assembly: Engine.Rendering.dll

```
public class RenderPoint
```

## Inheritance

[object](#) ← RenderPoint

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### RenderPoint(int, int, Color)

```
public RenderPoint(int x, int y, Color color)
```

#### Parameters

x [int](#)

y [int](#)

color [Color](#)

## Fields

### Color

```
public Color Color
```

Field Value

[Color ↗](#)

X

```
public int X
```

Field Value

[int ↗](#)

Y

```
public int Y
```

Field Value

[int ↗](#)

## Methods

LerpColor(Color, float)

```
public void LerpColor(Color color, float t)
```

Parameters

color [Color ↗](#)

t [float ↗](#)

# Class RenderTexture

Namespace: [Engine.Rendering](#)

Assembly: Engine.Rendering.dll

```
public class RenderTexture
```

## Inheritance

[object](#) ← RenderTexture

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### RenderTexture(PictureBox, float)

```
public RenderTexture(PictureBox pictureBox, float pixelDensity = 90)
```

#### Parameters

pictureBox [PictureBox](#)

pixelDensity [float](#)

## Methods

### RenderPixel(float, float, Camera)

```
public Pixel RenderPixel(float x, float y, Camera camera)
```

#### Parameters

x [float](#)

y [float](#)

camera [Camera](#)

Returns

[Pixel](#)

# Class RenderingConstants

Namespace: [Engine.Rendering](#)

Assembly: Engine.Rendering.dll

```
public static class RenderingConstants
```

## Inheritance

[object](#) ← RenderingConstants

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Fields

### FoV

```
public static int FoV
```

#### Field Value

[int](#)

### Sin135

```
public static readonly float Sin135
```

#### Field Value

[float](#)

## Unit

Defines the Unit size in units

```
public const int Unit = 1
```

Field Value

[int ↗](#)

## \_halfScreenHeight

```
public static int _halfScreenHeight
```

Field Value

[int ↗](#)

## \_halfScreenWidth

```
public static int _halfScreenWidth
```

Field Value

[int ↗](#)

## \_screenHeight

Only applies when rendering with the orthographic perspective UnitXY multiplied with  $\sqrt{2} / 2$

```
public static int _screenHeight
```

Field Value

[int ↗](#)

## \_screenWidth

```
public static int _screenWidth
```

Field Value

[int↗](#)

## Properties

### HalfScreenHeight

```
public static int HalfScreenHeight { get; }
```

Property Value

[int↗](#)

### HalfScreenWidth

```
public static int HalfScreenWidth { get; }
```

Property Value

[int↗](#)

### ScreenHeight

```
public static int ScreenHeight { get; set; }
```

Property Value

[int↗](#)

## ScreenWidth

```
public static int ScreenWidth { get; set; }
```

Property Value

[int ↗](#)