

Namespace Sparkle.CSharp

Classes

[Game](#)

[SystemInfo](#)

[Time](#)

Structs

[GameSettings](#)

Class Game

Namespace: [Sparkle.CSharp](#)

Assembly: Sparkle.dll

```
public class Game : Disposable, IDisposable
```

Inheritance

[object](#) ← Disposable ← Game

Implements

[IDisposable](#)

Derived

[TestGame](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

Game(GameSettings)

Initializes the [Game](#) with specified settings.

```
public Game(GameSettings settings)
```

Parameters

`settings` [GameSettings](#)

The game settings.

Fields

ShouldClose

Flag to indicate if the game should close.

```
public bool ShouldClose
```

Field Value

[bool](#)

Version

The version of the game engine (Sparkle).

```
public static readonly Version Version
```

Field Value

[Version](#)

Properties

CommandList

The command list used for rendering commands.

```
public CommandList CommandList { get; }
```

Property Value

[CommandList](#)

Content

The content manager used to load game assets.

```
public ContentManager Content { get; }
```

Property Value

[ContentManager](#)

FullScreenRenderPass

Represents the fullscreen render pass used during the rendering process.

```
public FullScreenRenderer FullScreenRenderPass { get; }
```

Property Value

FullScreenRenderer

GlobalImmediateRenderer

The default (global) immediate renderer used for rendering (Vertices...) in immediate mode.

```
public ImmediateRenderer GlobalImmediateRenderer { get; }
```

Property Value

ImmediateRenderer

GlobalPrimitiveBatch

The default (global) primitive batch used for rendering 2D geometric primitives.

```
public PrimitiveBatch GlobalPrimitiveBatch { get; }
```

Property Value

PrimitiveBatch

GlobalSpriteBatch

The default (global) sprite batch used for rendering 2D sprites.

```
public SpriteBatch GlobalSpriteBatch { get; }
```

Property Value

SpriteBatch

GraphicsContext

An instance encapsulating core graphics rendering components associated with the game.

```
public GraphicsContext GraphicsContext { get; }
```

Property Value

[GraphicsContext](#)

GraphicsDevice

The graphics device for rendering.

```
public GraphicsDevice GraphicsDevice { get; }
```

Property Value

[GraphicsDevice](#) ↗

Instance

The singleton instance of the game.

```
public static Game? Instance { get; }
```

Property Value

[Game](#)

MainWindow

The main window of the game.

```
public IWindow MainWindow { get; }
```

Property Value

IWindow

Settings

The settings for the game.

```
public GameSettings Settings { get; }
```

Property Value

[GameSettings](#)

Methods

AfterUpdate(double)

Final update after regular updates are completed.

```
protected virtual void AfterUpdate(double delta)
```

Parameters

delta [double](#) ↗

The time delta since the last update.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

`disposing` [bool](#)

True if called from user code; false if called from a finalizer.

Draw(GraphicsContext, Framebuffer)

Renders the game scene to the screen.

```
protected virtual void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

`context` [GraphicsContext](#)

`framebuffer` [Framebuffer](#)

FixedUpdate(double)

Executes fixed update logic with the specified time step.

```
protected virtual void FixedUpdate(double fixedStep)
```

Parameters

`fixedStep` [double](#)

The fixed time step in seconds.

GetSampleCount()

Retrieves the texture sample count currently used by the game's MSAA render target texture.

```
public TextureSampleCount? GetSampleCount()
```

Returns

[TextureSampleCount](#)?

The sample count of the MSAA render target texture.

GetTargetFps()

Gets the target frames per second.

```
public int GetTargetFps()
```

Returns

[int](#)

Init()

Initializes global game resources.

```
protected virtual void Init()
```

Load(ContentManager)

Loads the required game content and resources.

```
protected virtual void Load(ContentManager content)
```

Parameters

content [ContentManager](#)

OnClose()

Handles the logic to be executed when the application shuts down. This method can be overridden by derived classes to include custom shutdown behavior.

```
protected virtual void OnClose()
```

OnResize(Rectangle)

Handles window resizing events.

```
protected virtual void OnResize(Rectangle rectangle)
```

Parameters

`rectangle` Rectangle

OnRun()

Virtual method for additional setup when the game starts.

```
protected virtual void OnRun()
```

Run(Scene?)

Starts the game loop, initializing all necessary components and running the game.

```
public void Run(Scene? scene)
```

Parameters

`scene` [Scene](#)

The scene to load initially.

SetSampleCount(TextureSampleCount)

Sets the sample count for multi-sample anti-aliasing (MSAA).

```
public void SetSampleCount(TextureSampleCount sampleCount)
```

Parameters

sampleCount [TextureSampleCount](#)

The texture sample count to apply, defining the level of anti-aliasing.

SetTargetFps(int)

Sets the target frames per second.

```
public void SetTargetFps(int fps)
```

Parameters

fps [int](#)

Update(double)

Updates the game state, including scene and UI management.

```
protected virtual void Update(double delta)
```

Parameters

delta [double](#)

The time delta since the last update.

Struct GameSettings

Namespace: [Sparkle.CSharp](#)

Assembly: Sparkle.dll

```
public struct GameSettings
```

Inherited Members

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

Constructors

GameSettings()

Initializes a new instance of the [GameSettings](#) struct with default values.

```
public GameSettings()
```

Properties

Backend

The graphics backend to use for rendering (e.g., Direct3D, Vulkan, Metal, OpenGL, OpenGL-ES).

```
public GraphicsBackend Backend { readonly get; init; }
```

Property Value

[GraphicsBackend](#)

FixedTimeStep

The fixed time step for game updates, typically used for physics calculations.

```
public float FixedTimeStep { readonly get; init; }
```

Property Value

[float](#)

IconPath

The file path to the icon used for the game window.

```
public string IconPath { readonly get; init; }
```

Property Value

[string](#)

LogDirectory

The directory path where log files will be stored.

```
public string LogDirectory { readonly get; init; }
```

Property Value

[string](#)

MinSize

The minimum allowed dimensions for the game window, defined as a tuple of width and height.

```
public (int Width, int Height) MinSize { readonly get; init; }
```

Property Value

[\(int](#) [Width](#), [int](#) [Height](#))

SampleCount

The sample count for anti-aliasing. Higher values result in smoother edges but increase performance cost.

```
public TextureSampleCount SampleCount { readonly get; init; }
```

Property Value

[TextureSampleCount](#)

Size

The dimensions of the game window, defined as a tuple containing the width and height in pixels.

```
public (int Width, int Height) Size { readonly get; init; }
```

Property Value

([int](#), [Width](#), [int](#), [Height](#))

TargetFps

The target frames per second (FPS). A value of 0 means no FPS cap.

```
public int TargetFps { readonly get; init; }
```

Property Value

[int](#)

Title

The title of the game window.

```
public string Title { readonly get; init; }
```

Property Value

[string](#) ↗

VSync

Indicates whether vertical sync (VSync) is enabled to prevent screen tearing.

```
public bool VSync { readonly get; init; }
```

Property Value

[bool](#) ↗

WindowFlags

The window state flags that define the behavior and appearance of the game window.

```
public WindowState WindowFlags { readonly get; init; }
```

Property Value

WindowState

Class SystemInfo

Namespace: [Sparkle.CSharp](#)

Assembly: Sparkle.dll

```
public static class SystemInfo
```

Inheritance

[object](#) ← SystemInfo

Inherited Members

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

Fields

Cpu

Gets the name of the processor used in the system.

```
public static readonly string Cpu
```

Field Value

[string](#)

MemoryInfo

Gets the memory info.

```
public static readonly (int Total, int Available) MemoryInfo
```

Field Value

([int](#) [Width](#) , [int](#) [Height](#))

Os

Gets the operating system version string.

```
public static readonly string Os
```

Field Value

[string](#)

Threads

Gets the number of available processor threads.

```
public static readonly int Threads
```

Field Value

[int](#)

Class Time

Namespace: [Sparkle.CSharp](#)

Assembly: Sparkle.dll

```
public static class Time
```

Inheritance

[object](#) ← Time

Inherited Members

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

Properties

Delta

Gets the time elapsed between the current and the previous frame.

```
public static double Delta { get; }
```

Property Value

[double](#)

FixedAccumulator

Represents an accumulator used to track fixed time steps for the physics or fixed update loop.

```
public static double FixedAccumulator { get; }
```

Property Value

[double](#)

FixedStep

Gets the fixed time step interval defined in the current game settings.

```
public static double FixedStep { get; }
```

Property Value

[double](#)

Total

Gets the total elapsed time since the start of the application.

```
public static double Total { get; }
```

Property Value

[double](#)

Namespace Sparkle.CSharp.Content

Classes

[ContentManager](#)

Class ContentManager

Namespace: [Sparkle.CSharp.Content](#)

Assembly: Sparkle.dll

```
public class ContentManager : Disposable, IDisposable
```

Inheritance

[object](#) ← Disposable ← ContentManager

Implements

[IDisposable](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

ContentManager(GraphicsDevice)

Initializes a new instance of the [ContentManager](#) class.

```
public ContentManager(GraphicsDevice graphicsDevice)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for loading assets.

Properties

GraphicsDevice

The graphics device used for loading graphical assets.

```
public GraphicsDevice GraphicsDevice { get; }
```

Property Value

[GraphicsDevice](#)

Methods

AddProcessors(Type, IContentProcessor)

Adds a new content processor for a specific content type.

```
public void AddProcessors(Type type, IContentProcessor processor)
```

Parameters

type [Type](#)

The type of content the processor will handle.

processor [IContentProcessor](#)

The content processor to associate with the specified type.

AddUnmanagedItem<T>(T)

Adds an unmanaged item to the content manager.

```
public void AddUnmanagedItem<T>(T item)
```

Parameters

item T

The item to add.

Type Parameters

The type of content.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

GetProcessor(Type)

Retrieves a content processor for the specified type, if one exists.

```
public IContentProcessor? GetProcessor(Type type)
```

Parameters

type [Type](#)

The type of content for which the processor is requested.

Returns

[IContentProcessor](#)

The content processor for the specified type, or null if no processor is registered.

GetProcessors()

Retrieves all content processors currently registered in the content manager.

```
public IEnumerable<IContentProcessor> GetProcessors()
```

Returns

[IEnumerable](#) <[IContentProcessor](#)>

An enumerable collection of the registered content processors.

HasProcessor(Type)

Determines whether a content processor exists for the specified type.

```
public bool HasProcessor(Type type)
```

Parameters

type [Type](#)

The type to check for an associated content processor.

Returns

[bool](#)

True if a processor exists for the specified type; otherwise, false.

Load<T>(IContentType<T>)

Loads content using an appropriate processor.

```
public T Load<T>(IContentType<T> type)
```

Parameters

type [IContentType](#)<T>

The content type to load.

Returns

T

The loaded content.

Type Parameters

T

The type of content to load.

TryAddProcessors(Type, IContentProcessor)

Attempts to add a content processor for a specific type.

```
public bool TryAddProcessors(Type type, IContentProcessor processor)
```

Parameters

type [Type](#)

The type the processor handles.

processor [IContentProcessor](#)

The content processor instance.

Returns

[bool](#)

True if successfully added, otherwise false.

TryGetProcessor(Type, out IContentProcessor?)

Attempts to retrieve a content processor for a specified type.

```
public bool TryGetProcessor(Type type, out IContentProcessor? processor)
```

Parameters

type [Type ↗](#)

The type of content.

processor [IContentProcessor](#)

The found content processor, or null if not found.

Returns

[bool ↗](#)

True if a processor was found, otherwise false.

Unload<T>(T)

Unloads a content item, freeing associated resources.

```
public void Unload<T>(T item)
```

Parameters

item T

The item to unload.

Type Parameters

T

The type of content to unload.

Namespace Sparkle.CSharp.Content.Processors

Classes

[AudioClipProcessor](#)

[CubemapProcessor](#)

[FontProcessor](#)

[ModelProcessor](#)

[TextureProcessor](#)

Interfaces

[IContentProcessor](#)

Class AudioClipProcessor

Namespace: [Sparkle.CSharp.Content.Processors](#)

Assembly: Sparkle.dll

```
public class AudioClipProcessor : IContentProcessor
```

Inheritance

[object](#) ← AudioClipProcessor

Implements

[IContentProcessor](#)

Inherited Members

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

Methods

Load<T>(GraphicsDevice, IContentType<T>)

Loads an audio clip from the specified path and returns the loaded audio clip object.

```
public object Load<T>(GraphicsDevice graphicsDevice, IContentType<T> type)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for managing content.

type [IContentType](#)<T>

The content type descriptor that contains information about the audio clip, including its path and whether to stream it from disk.

Returns

[object](#)

An object representing the loaded audio clip.

Type Parameters

T

The type of content to load, specifically an audio clip.

Unload(object)

Unloads and disposes the specified content item, freeing any resources it holds.

```
public void Unload(object item)
```

Parameters

item object

The content item to be unloaded and disposed.

Class CubemapProcessor

Namespace: [Sparkle.CSharp.Content.Processors](#)

Assembly: Sparkle.dll

```
public class CubemapProcessor : IContentProcessor
```

Inheritance

[object](#) ← CubemapProcessor

Implements

[IContentProcessor](#)

Inherited Members

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

Methods

Load<T>(GraphicsDevice, IContentType<T>)

Loads a cubemap resource into memory using the provided graphics device and content type.

```
public object Load<T>(GraphicsDevice graphicsDevice, IContentType<T> type)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for creating GPU resources.

type [IContentType](#)<T>

The content type containing metadata about the resource to be loaded.

Returns

[object](#)

An object representing the loaded cubemap resource.

Type Parameters

T

The type of content that will be loaded.

Unload(object)

Unloads a previously loaded cubemap resource and releases its associated GPU resources.

```
public void Unload(object item)
```

Parameters

item object ↗

The cubemap resource to be unloaded.

Class FontProcessor

Namespace: [Sparkle.CSharp.Content.Processors](#)

Assembly: Sparkle.dll

```
public class FontProcessor : IContentProcessor
```

Inheritance

[object](#) ← FontProcessor

Implements

[IContentProcessor](#)

Inherited Members

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

Methods

Load<T>(GraphicsDevice, IContentType<T>)

Loads a font resource from the specified content type.

```
public object Load<T>(GraphicsDevice graphicsDevice, IContentType<T> type)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for rendering.

type [IContentType](#)<T>

The content type containing the font path.

Returns

[object](#)

A new Bliss.CSharp.Fonts.Font instance loaded from the specified path.

Type Parameters

T

The type of content being loaded.

Unload(object)

Unloads the specified content item and releases any resources associated with it.

```
public void Unload(object item)
```

Parameters

item object

The content item to unload, typically of a resource type such as a font.

Interface IContentProcessor

Namespace: [Sparkle.CSharp.Content.Processors](#)

Assembly: Sparkle.dll

```
public interface IContentProcessor
```

Methods

Load<T>(GraphicsDevice, IContentType<T>)

Loads the specified content of type **T** using the appropriate content processor.

```
object Load<T>(GraphicsDevice graphicsDevice, IContentType<T> type)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for loading content.

type [IContentType](#)<T>

The content type descriptor that includes information about the path and content.

Returns

[object](#)

The loaded content item of type **T**.

Type Parameters

T

The type of the content to load.

Unload(object)

Unloads the specified content item and releases any resources associated with it.

```
void Unload(object item)
```

Parameters

item [object](#)

The content item to unload.

Class ModelProcessor

Namespace: [Sparkle.CSharp.Content.Processors](#)

Assembly: Sparkle.dll

```
public class ModelProcessor : IContentProcessor
```

Inheritance

[object](#) ← ModelProcessor

Implements

[IContentProcessor](#)

Inherited Members

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

Methods

Load<T>(GraphicsDevice, IContentType<T>)

Loads a model using the given graphics device and content type descriptor.

```
public object Load<T>(GraphicsDevice graphicsDevice, IContentType<T> type)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used to load the model.

type [IContentType](#)<T>

The content type descriptor that contains information such as the path, material loading option, and UV flip configuration.

Returns

[object](#)

A loaded model instance.

Type Parameters

T

The type of the content to be loaded.

Unload(object)

Unloads the specified model and releases any associated resources.

```
public void Unload(object item)
```

Parameters

item object

The model instance to unload.

Class TextureProcessor

Namespace: [Sparkle.CSharp.Content.Processors](#)

Assembly: Sparkle.dll

```
public class TextureProcessor : IContentProcessor
```

Inheritance

[object](#) ← TextureProcessor

Implements

[IContentProcessor](#)

Inherited Members

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

Methods

Load<T>(GraphicsDevice, IContentType<T>)

Loads a texture resource of type [T](#) using the provided graphics device and content type descriptor.

```
public object Load<T>(GraphicsDevice graphicsDevice, IContentType<T> type)
```

Parameters

[graphicsDevice](#) [GraphicsDevice](#)

The graphics device used to create and manage the texture resource.

[type](#) [IContentType](#)<T>

The content type descriptor, containing information such as the texture path and configuration.

Returns

[object](#)

The loaded texture resource as an object.

Type Parameters

T

The type of the texture resource to load.

Unload(object)

Unloads and disposes of a previously loaded texture resource.

```
public void Unload(object item)
```

Parameters

item object

The texture resource to unload, typically an instance of Bliss.CSharp.Textures.Texture2D.

Namespace Sparkle.CSharp.Content.Types

Classes

[AudioClipContent](#)

[CubemapContent](#)

[FontContent](#)

[ModelContent](#)

[TextureContent](#)

Interfaces

[IContentType<T>](#)

Class AudioClipContent

Namespace: [Sparkle.CSharp.Content.Types](#)

Assembly: Sparkle.dll

```
public class AudioClipContent : IContentType<AudioClip>
```

Inheritance

[object](#) ← AudioClipContent

Implements

[IContentType](#)<AudioClip>

Inherited Members

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

Constructors

AudioClipContent(string, bool)

Initializes a new instance of the [AudioClipContent](#) class.

```
public AudioClipContent(string path, bool streamFromDisk = true)
```

Parameters

path [string](#)

The file path of the audio clip.

streamFromDisk [bool](#)

Whether to stream from disk (default: true).

Properties

Path

The file path of the audio clip.

```
public string Path { get; }
```

Property Value

[string](#)

StreamFromDisk

Determines whether the audio clip should be streamed from disk instead of loading into memory.

```
public bool StreamFromDisk { get; }
```

Property Value

[bool](#)

Class CubemapContent

Namespace: [Sparkle.CSharp.Content.Types](#)

Assembly: Sparkle.dll

```
public class CubemapContent : IContentType<Cubemap>
```

Inheritance

[object](#) ← CubemapContent

Implements

[IContentType](#)<Cubemap>

Inherited Members

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

Constructors

CubemapContent(string, CubemapLayout, bool, bool)

Initializes a new instance of the [CubemapContent](#) class.

```
public CubemapContent(string path, CubemapLayout layout = CubemapLayout.AutoDetect, bool  
mipmap = true, bool srgb = false)
```

Parameters

path [string](#)

The file path of the cubemap.

layout [CubemapLayout](#)

The layout configuration of the cubemap (default is AutoDetect).

mipmap [bool](#)

Whether to generate mipmaps for the cubemap (default is true).

srgb [bool](#)

Whether to use the sRGB format for the cubemap (default is false).

Properties

Layout

The layout configuration of the cubemap.

```
public CubemapLayout Layout { get; }
```

Property Value

CubemapLayout

Mipmap

Indicates whether to generate mipmaps for the cubemap.

```
public bool Mipmap { get; }
```

Property Value

[bool](#)

Path

The file path of the cubemap.

```
public string Path { get; }
```

Property Value

[string](#)

UseSrgbFormat

Indicates whether to use the sRGB color format for the cubemap.

```
public bool UseSrgbFormat { get; }
```

Property Value

[bool](#)

Class FontContent

Namespace: [Sparkle.CSharp.Content.Types](#)

Assembly: Sparkle.dll

```
public class FontContent : IContentType<Font>
```

Inheritance

[object](#) ← FontContent

Implements

[IContentType](#)

Inherited Members

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

Constructors

FontContent(string, FontSystemSettings?)

Initializes a new instance of the [FontContent](#) class with the specified path and optional settings.

```
public FontContent(string path, FontSystemSettings? settings = null)
```

Parameters

path [string](#)

The file path of the font to load.

settings [FontSystemSettings](#)

Optional font system settings; if not provided, default settings are used.

Fields

Settings

The settings used to configure the font system when loading this font.

```
public FontSystemSettings? Settings
```

Field Value

FontSystemSettings

Properties

Path

The file path of the font.

```
public string Path { get; }
```

Property Value

[string](#)

Interface IContentType<T>

Namespace: [Sparkle.CSharp.Content.Types](#)

Assembly: Sparkle.dll

```
public interface IContentType<T>
```

Type Parameters

T

Properties

Path

The file path of the content.

```
string Path { get; }
```

Property Value

[string](#) ↗

Class ModelContent

Namespace: [Sparkle.CSharp.Content.Types](#)

Assembly: Sparkle.dll

```
public class ModelContent : IContentType<Model>
```

Inheritance

[object](#) ← ModelContent

Implements

[IContentType](#)<Model>

Inherited Members

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

Constructors

ModelContent(string, bool, bool)

Initializes a new instance of the [ModelContent](#) class.

```
public ModelContent(string path, bool loadMaterial = true, bool flipUv = false)
```

Parameters

path [string](#)

The file path of the model.

loadMaterial [bool](#)

Whether to load the model's material (default is true).

flipUv [bool](#)

Whether to flip the model's UV mapping (default is false).

Properties

FlipUv

Indicates whether to flip the model's UV mapping.

```
public bool FlipUv { get; }
```

Property Value

[bool](#) ↗

LoadMaterial

Indicates whether to load the material with the model.

```
public bool LoadMaterial { get; }
```

Property Value

[bool](#) ↗

Path

The file path of the model content.

```
public string Path { get; }
```

Property Value

[string](#) ↗

Class TextureContent

Namespace: [Sparkle.CSharp.Content.Types](#)

Assembly: Sparkle.dll

```
public class TextureContent : IContentType<Texture2D>
```

Inheritance

[object](#) ← TextureContent

Implements

[IContentType](#)<Texture2D>

Inherited Members

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

Constructors

TextureContent(string, bool, bool)

Initializes a new instance of the [TextureContent](#) class.

```
public TextureContent(string path, bool mipmap = true, bool srgb = false)
```

Parameters

path [string](#)

The file path of the texture.

mipmap [bool](#)

Whether to generate mipmaps for the texture (default is true).

srgb [bool](#)

Whether to use the sRGB format for the texture (default is false).

Properties

Mipmap

Indicates whether to generate mipmaps for the texture.

```
public bool Mipmap { get; }
```

Property Value

[bool](#)

Path

The file path of the texture content.

```
public string Path { get; }
```

Property Value

[string](#)

UseSrgbFormat

Indicates whether to use the sRGB color format for the texture.

```
public bool UseSrgbFormat { get; }
```

Property Value

[bool](#)

Namespace Sparkle.CSharp.Effects.Filters

Classes

[BloomEffect](#)

[BlurEffect](#)

[PixelizerEffect](#)

[PosterizationEffect](#)

[PredatorEffect](#)

[SobelEffect](#)

Class BloomEffect

Namespace: [Sparkle.CSharp.Effects.Filters](#)

Assembly: Sparkle.dll

```
public class BloomEffect : Effect, IDisposable
```

Inheritance

[object](#) ← Disposable ← Effect ← BloomEffect

Implements

[IDisposable](#)

Inherited Members

Effect.Shader , Effect.VertexLayouts , Effect.ShaderSet , [Effect.LoadBytecodeFromFile\(string\)](#) ,
[Effect.LoadTextCodeFromFile\(string\)](#) , Effect.GetBufferLayouts() , [Effect.GetBufferLayout\(string\)](#) ,
[Effect.GetBufferLayoutSlot\(string\)](#) ,
[Effect.AddBufferLayout\(string, uint, SimpleBufferType, ShaderStages\)](#) , Effect.GetTextureLayouts() ,
[Effect.GetTextureLayout\(string\)](#) , [Effect.GetTextureLayoutSlot\(string\)](#) ,
[Effect.AddTextureLayout\(string, uint\)](#) , Effect.GetPipeline(SimplePipelineDescription) ,
Effect.GraphicsDevice , Effect.Specializations , Effect.Macros , Disposable.Dispose() ,
Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

BloomEffect(GraphicsDevice, VertexLayoutDescription, CrossCompileOptions, float, float)

Initializes a new instance of the [BloomEffect](#) class.

```
public BloomEffect(GraphicsDevice graphicsDevice, VertexLayoutDescription vertexLayout,  
CrossCompileOptions compileOptions, float samples = 5, float quality = 2.5)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for rendering.

vertexLayout [VertexLayoutDescription](#)

The vertex layout for full-screen rendering.

compileOptions CrossCompileOptions

Optional cross-compilation options used when creating the shaders.

samples [float](#)

The number of blur samples to use in the bloom effect.

quality [float](#)

The quality factor of the bloom blur.

Fields

FragPath

Path to the fragment shader.

```
public static readonly string FragPath
```

Field Value

[string](#)

VertPath

Path to the vertex shader.

```
public static readonly string VertPath
```

Field Value

[string](#)

Properties

Quality

Gets or sets the quality factor of the bloom blur (Affects the blur strength or sharpness of the effect).

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

Property Value

[float](#)

Samples

Gets or sets the number of blur samples used in the bloom effect (Higher values result in smoother, more spread-out bloom).

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

Property Value

[float](#)

Methods

Apply(CommandList, Material?)

Applies the bloom effect using the current parameter values.

```
public override void Apply(CommandList commandList, Material? material = null)
```

Parameters

commandList [CommandList](#)

The command list used to issue GPU commands.

material Material

An optional material to apply with the effect.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Resize()

Called when the window is resized to update the internal resolution parameter.

```
protected virtual void Resize()
```

Class BlurEffect

Namespace: [Sparkle.CSharp.Effects.Filters](#)

Assembly: Sparkle.dll

```
public class BlurEffect : Effect, IDisposable
```

Inheritance

[object](#) ← Disposable ← Effect ← BlurEffect

Implements

[IDisposable](#)

Inherited Members

Effect.Shader , Effect.VertexLayouts , Effect.ShaderSet , [Effect.LoadBytecodeFromFile\(string\)](#) ,
[Effect.LoadTextCodeFromFile\(string\)](#) , Effect.GetBufferLayouts() , [Effect.GetBufferLayout\(string\)](#) ,
[Effect.GetBufferLayoutSlot\(string\)](#) ,
[Effect.AddBufferLayout\(string, uint, SimpleBufferType, ShaderStages\)](#) , Effect.GetTextureLayouts() ,
[Effect.GetTextureLayout\(string\)](#) , [Effect.GetTextureLayoutSlot\(string\)](#) ,
[Effect.AddTextureLayout\(string, uint\)](#) , Effect.GetPipeline(SimplePipelineDescription) ,
Effect.GraphicsDevice , Effect.Specializations , Effect.Macros , Disposable.Dispose() ,
Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

BlurEffect(GraphicsDevice, VertexLayoutDescription, CrossCompileOptions, float, int)

Initializes a new instance of the [BlurEffect](#) class.

```
public BlurEffect(GraphicsDevice graphicsDevice, VertexLayoutDescription vertexLayout,  
CrossCompileOptions compileOptions, float intensity = 3, int radius = 5)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used to create GPU resources.

vertexLayout [VertexLayoutDescription](#)

The layout of the vertices used in the full-screen pass.

compileOptions CrossCompileOptions

Optional cross-compilation options used when creating the shaders.

intensity [float](#)

The strength of the blur effect.

radius [int](#)

The blur radius determining the spread of the blur.

Fields

FragPath

Path to the fragment shader.

```
public static readonly string FragPath
```

Field Value

[string](#)

VertPath

Path to the vertex shader.

```
public static readonly string VertPath
```

Field Value

[string](#)

Properties

Intensity

Gets or sets the intensity of the blur effect (Higher values result in stronger blurring).

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

Property Value

[float ↗](#)

Radius

Gets or sets the blur radius (Controls how far neighboring pixels are sampled).

```
public int Radius { get; set; }
```

Property Value

[int ↗](#)

Methods

Apply(CommandList, Material?)

Applies the blur effect using the current parameter values.

```
public override void Apply(CommandList commandList, Material? material = null)
```

Parameters

commandList [CommandList ↗](#)

The command list used to issue GPU commands.

material [Material](#)

An optional material to apply with the effect.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Resize()

Called when the window is resized to update the internal resolution parameter.

```
protected virtual void Resize()
```

Class PixelizerEffect

Namespace: [Sparkle.CSharp.Effects.Filters](#)

Assembly: Sparkle.dll

```
public class PixelizerEffect : Effect, IDisposable
```

Inheritance

[object](#) ← Disposable ← Effect ← PixelizerEffect

Implements

[IDisposable](#)

Inherited Members

Effect.Shader , Effect.VertexLayouts , Effect.ShaderSet , [Effect.LoadBytecodeFromFile\(string\)](#) ,
[Effect.LoadTextCodeFromFile\(string\)](#) , Effect.GetBufferLayouts() , [Effect.GetBufferLayout\(string\)](#) ,
[Effect.GetBufferLayoutSlot\(string\)](#) ,
[Effect.AddBufferLayout\(string, uint, SimpleBufferType, ShaderStages\)](#) , Effect.GetTextureLayouts() ,
[Effect.GetTextureLayout\(string\)](#) , [Effect.GetTextureLayoutSlot\(string\)](#) ,
[Effect.AddTextureLayout\(string, uint\)](#) , Effect.GetPipeline(SimplePipelineDescription) ,
Effect.GraphicsDevice , Effect.Specializations , Effect.Macros , Disposable.Dispose() ,
Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

PixelizerEffect(GraphicsDevice, VertexLayoutDescription, CrossCompileOptions, Vector2?)

Initializes a new instance of the [PixelizerEffect](#) class.

```
public PixelizerEffect(GraphicsDevice graphicsDevice, VertexLayoutDescription vertexLayout,  
CrossCompileOptions compileOptions, Vector2? pixelSize = null)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device for buffer and resource allocation.

vertexLayout [VertexLayoutDescription](#)?

Vertex layout used by the full-screen pass.

compileOptions CrossCompileOptions

Optional cross-compilation options used when creating the shaders.

pixelSize [Vector2](#)?

The pixel size to simulate (defaults to 8x8 if null).

Fields

FragPath

Path to the fragment shader.

```
public static readonly string FragPath
```

Field Value

[string](#)?

VertPath

Path to the vertex shader.

```
public static readonly string VertPath
```

Field Value

[string](#)?

Properties

PixelSize

Gets or sets the size of the individual pixels for the pixelizer effect.

```
public Vector2 PixelSize { get; set; }
```

Property Value

[Vector2](#)

Methods

Apply(CommandList, Material?)

Applies the pixelizer effect using the current parameter values.

```
public override void Apply(CommandList commandList, Material? material = null)
```

Parameters

commandList [CommandList](#)

The command list used to issue GPU commands.

material Material

An optional material to apply with the effect.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Resize()

Called when the window is resized to update the internal resolution parameter.

```
protected virtual void Resize()
```

Class PosterizationEffect

Namespace: [Sparkle.CSharp.Effects.Filters](#)

Assembly: Sparkle.dll

```
public class PosterizationEffect : Effect, IDisposable
```

Inheritance

[object](#) ← Disposable ← Effect ← PosterizationEffect

Implements

[IDisposable](#)

Inherited Members

Effect.Shader , Effect.VertexLayouts , Effect.ShaderSet , [Effect.LoadBytecodeFromFile\(string\)](#) ,
[Effect.LoadTextCodeFromFile\(string\)](#) , Effect.GetBufferLayouts() , [Effect.GetBufferLayout\(string\)](#) ,
[Effect.GetBufferLayoutSlot\(string\)](#) ,
[Effect.AddBufferLayout\(string, uint, SimpleBufferType, ShaderStages\)](#) , Effect.GetTextureLayouts() ,
[Effect.GetTextureLayout\(string\)](#) , [Effect.GetTextureLayoutSlot\(string\)](#) ,
[Effect.AddTextureLayout\(string, uint\)](#) , Effect.GetPipeline(SimplePipelineDescription) ,
Effect.GraphicsDevice , Effect.Specializations , Effect.Macros , Disposable.Dispose() ,
Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

PosterizationEffect(GraphicsDevice, VertexLayoutDescription, CrossCompileOptions, int)

Initializes a new instance of the [PosterizationEffect](#) class.

```
public PosterizationEffect(GraphicsDevice graphicsDevice, VertexLayoutDescription  
vertexLayout, CrossCompileOptions compileOptions, int numColors = 8)
```

Parameters

[graphicsDevice](#) [GraphicsDevice](#)

The graphics device for buffer and resource allocation.

vertexLayout [VertexLayoutDescription](#)

Vertex layout used by the full-screen pass.

compileOptions CrossCompileOptions

Optional cross-compilation options used when creating the shaders.

numOfColors [int](#)

Number of discrete color levels to use (defaults to 8).

Fields

FragPath

Path to the fragment shader.

```
public static readonly string FragPath
```

Field Value

[string](#)

VertPath

Path to the vertex shader.

```
public static readonly string VertPath
```

Field Value

[string](#)

Properties

NumOfColors

Gets or sets the number of colors used in the posterization effect.

```
public int NumOfColors { get; set; }
```

Property Value

[int](#)

Methods

Apply(CommandList, Material?)

Applies the posterization effect using the current parameter values.

```
public override void Apply(CommandList commandList, Material? material = null)
```

Parameters

commandList [CommandList](#)

The command list used to issue GPU commands.

material Material

An optional material to apply with the effect.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Resize()

Called when the window is resized to update the internal resolution parameter.

```
protected virtual void Resize()
```

Class PredatorEffect

Namespace: [Sparkle.CSharp.Effects.Filters](#)

Assembly: Sparkle.dll

```
public class PredatorEffect : Effect, IDisposable
```

Inheritance

[object](#) ← Disposable ← Effect ← PredatorEffect

Implements

[IDisposable](#)

Inherited Members

Effect.Shader , Effect.VertexLayouts , Effect.ShaderSet , [Effect.LoadBytecodeFromFile\(string\)](#) ,
[Effect.LoadTextCodeFromFile\(string\)](#) , Effect.GetBufferLayouts() , [Effect.GetBufferLayout\(string\)](#) ,
[Effect.GetBufferLayoutSlot\(string\)](#) ,
[Effect.AddBufferLayout\(string, uint, SimpleBufferType, ShaderStages\)](#) , Effect.GetTextureLayouts() ,
[Effect.GetTextureLayout\(string\)](#) , [Effect.GetTextureLayoutSlot\(string\)](#) ,
[Effect.AddTextureLayout\(string, uint\)](#) , Effect.GetPipeline(SimplePipelineDescription) ,
Effect.GraphicsDevice , Effect.Specializations , Effect.Macros , Disposable.Dispose() ,
Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

PredatorEffect(GraphicsDevice, VertexLayoutDescription, CrossCompileOptions)

Initializes a new instance of the [PredatorEffect](#) class.

```
public PredatorEffect(GraphicsDevice graphicsDevice, VertexLayoutDescription vertexLayout,  
CrossCompileOptions compileOptions)
```

Parameters

graphicsDevice [GraphicsDevice](#)

Graphics device to create GPU resources.

vertexLayout [VertexLayoutDescription](#)

Vertex layout description for the full-screen quad.

compileOptions CrossCompileOptions

Optional cross-compilation options used when creating the shaders.

Fields

FragPath

Path to the fragment shader.

```
public static readonly string FragPath
```

Field Value

[string](#)

VertPath

Path to the vertex shader.

```
public static readonly string VertPath
```

Field Value

[string](#)

Methods

Apply(CommandList, Material?)

Applies the predator effect using the current parameter values.

```
public override void Apply(CommandList commandList, Material? material = null)
```

Parameters

commandList [CommandList](#)

The command list used to issue GPU commands.

material Material

An optional material to apply with the effect.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Resize()

Called when the window is resized to update the internal resolution parameter.

```
protected virtual void Resize()
```

Class SobelEffect

Namespace: [Sparkle.CSharp.Effects.Filters](#)

Assembly: Sparkle.dll

```
public class SobelEffect : Effect, IDisposable
```

Inheritance

[object](#) ← Disposable ← Effect ← SobelEffect

Implements

[IDisposable](#)

Inherited Members

Effect.Shader , Effect.VertexLayouts , Effect.ShaderSet , [Effect.LoadBytecodeFromFile\(string\)](#) ,
[Effect.LoadTextCodeFromFile\(string\)](#) , Effect.GetBufferLayouts() , [Effect.GetBufferLayout\(string\)](#) ,
[Effect.GetBufferLayoutSlot\(string\)](#) ,
[Effect.AddBufferLayout\(string, uint, SimpleBufferType, ShaderStages\)](#) , Effect.GetTextureLayouts() ,
[Effect.GetTextureLayout\(string\)](#) , [Effect.GetTextureLayoutSlot\(string\)](#) ,
[Effect.AddTextureLayout\(string, uint\)](#) , Effect.GetPipeline(SimplePipelineDescription) ,
Effect.GraphicsDevice , Effect.Specializations , Effect.Macros , Disposable.Dispose() ,
Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

SobelEffect(GraphicsDevice, VertexLayoutDescription, CrossCompileOptions)

Initializes a new instance of the [SobelEffect](#) class.

```
public SobelEffect(GraphicsDevice graphicsDevice, VertexLayoutDescription vertexLayout,  
CrossCompileOptions compileOptions)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used to create GPU resources.

vertexLayout [VertexLayoutDescription](#)

The vertex layout description for the full-screen quad.

compileOptions CrossCompileOptions

Optional cross-compilation options used when creating the shaders.

Fields

FragPath

Path to the fragment shader.

```
public static readonly string FragPath
```

Field Value

[string](#)

VertPath

Path to the vertex shader.

```
public static readonly string VertPath
```

Field Value

[string](#)

Methods

Apply(CommandList, Material?)

Applies the sobel effect using the current parameter values.

```
public override void Apply(CommandList commandList, Material? material = null)
```

Parameters

commandList [CommandList](#)

The command list used to issue GPU commands.

material Material

An optional material to apply with the effect.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Resize()

Called when the window is resized to update the internal resolution parameter.

```
protected virtual void Resize()
```

Namespace Sparkle.CSharp.Effects.Posts

Classes

[FxaaEffect](#)

Class FxaaEffect

Namespace: [Sparkle.CSharp.Effects.Posts](#)

Assembly: Sparkle.dll

```
public class FxaaEffect : Effect, IDisposable
```

Inheritance

[object](#) ← Disposable ← Effect ← FxaaEffect

Implements

[IDisposable](#)

Inherited Members

Effect.Shader , Effect.VertexLayouts , Effect.ShaderSet , [Effect.LoadBytecodeFromFile\(string\)](#) ,
[Effect.LoadTextCodeFromFile\(string\)](#) , Effect.GetBufferLayouts() , [Effect.GetBufferLayout\(string\)](#) ,
[Effect.GetBufferLayoutSlot\(string\)](#) ,
[Effect.AddBufferLayout\(string, uint, SimpleBufferType, ShaderStages\)](#) , Effect.GetTextureLayouts() ,
[Effect.GetTextureLayout\(string\)](#) , [Effect.GetTextureLayoutSlot\(string\)](#) ,
[Effect.AddTextureLayout\(string, uint\)](#) , Effect.GetPipeline(SimplePipelineDescription) ,
Effect.GraphicsDevice , Effect.Specializations , Effect.Macros , Disposable.Dispose() ,
Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

FxaaEffect(GraphicsDevice, VertexLayoutDescription, CrossCompileOptions, float, float, float)

Initializes a new instance of the [FxaaEffect](#) class with optional FXAA tuning values.

```
public FxaaEffect(GraphicsDevice graphicsDevice, VertexLayoutDescription vertexLayout,  
CrossCompileOptions compileOptions, float reduceMin = 0.0078125, float reduceMul = 0.125,  
float spanMax = 8)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for rendering.

vertexLayout [VertexLayoutDescription](#)

The layout of vertices for the full-screen pass.

compileOptions [CrossCompileOptions](#)

Optional cross-compilation options used when creating the shaders.

reduceMin [float](#)

Minimum amount of local contrast reduction.

reduceMul [float](#)

Multiplier for local contrast reduction.

spanMax [float](#)

Maximum blur span.

Fields

FragPath

Path to the fragment shader.

```
public static readonly string FragPath
```

Field Value

[string](#)

VertPath

Path to the vertex shader.

```
public static readonly string VertPath
```

Field Value

[string](#) ↗

Properties

ReduceMin

Controls the minimum contrast threshold below which edges are ignored during processing.

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

Property Value

[float](#) ↗

ReduceMul

Determines the multiplier for reducing contrast, influencing how strongly contrast is diminished.

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

Property Value

[float](#) ↗

SpanMax

Specifies the maximum extent of the blur effect, with higher values extending the anti-aliasing span across more pixels.

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

Property Value

[float](#) ↗

Methods

Apply(CommandList, Material?)

Applies the FXAA effect using the current parameter values.

```
public override void Apply(CommandList commandList, Material? material = null)
```

Parameters

commandList [CommandList](#)

The command list used to issue GPU commands.

material Material

An optional material to apply with the effect.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Resize()

Called when the window is resized to update the internal resolution parameter.

```
protected virtual void Resize()
```

Namespace Sparkle.CSharp.Entities

Classes

[Camera2D](#)

[Camera3D](#)

[Entity](#)

Class Camera2D

Namespace: [Sparkle.CSharp.Entities](#)

Assembly: Sparkle.dll

```
public class Camera2D : Entity, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Entity](#) ← Camera2D

Implements

[IDisposable](#)

Inherited Members

[Entity.Scene](#), [Entity.Id](#), [Entity.Tag](#), [Entity.Init\(\)](#), [Entity.AfterUpdate\(double\)](#), [Entity.FixedUpdate\(double\)](#),
[Entity.Draw\(GraphicsContext, Framebuffer\)](#), [Entity.GetComponents\(\)](#), [Entity.HasComponent<T>\(\)](#),
[Entity.GetComponent<T>\(\)](#), [Entity.TryGetComponent<T>\(out T\)](#), [Entity.AddComponent\(Component\)](#),
[Entity.TryAddComponent\(Component\)](#), [Entity.RemoveComponent\(Component\)](#),
[Entity.TryRemoveComponent\(Component\)](#), [Entity.RemoveComponent<T>\(\)](#),
[Entity.TryRemoveComponent<T>\(\)](#), [Entity.Dispose\(bool\)](#), [Disposable.Dispose\(\)](#),
[Disposable.ThrowIfDisposed\(\)](#), [Disposable.HasDisposed](#), [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Constructors

Camera2D(Vector2, Vector2, Rectangle, CameraFollowMode,
Vector2?, float, float)

Initializes a new instance of the [Camera2D](#) class.

```
public Camera2D(Vector2 position, Vector2 target, Rectangle size, CameraFollowMode mode,  
Vector2? offset = null, float rotation = 0, float zoom = 1)
```

Parameters

position [Vector2](#)

The initial position of the camera.

target [Vector2](#)?

The target position the camera follows.

size Rectangle

The size of the camera viewport.

mode CameraFollowMode

The follow mode of the camera.

offset [Vector2](#)?

Optional offset applied to the camera.

rotation float

Initial rotation of the camera in degrees.

zoom float

Initial zoom level of the camera.

Properties

FractionFollowSpeed

Gets or sets the fraction follow speed for smoothing camera movement.

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

Property Value

[float](#)

MinFollowEffectLength

Gets or sets the minimum follow effect length.

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

Property Value

[float](#)

MinFollowSpeed

Gets or sets the minimum follow speed of the camera.

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

Property Value

[float](#)

Mode

Gets or sets the camera follow mode.

```
public CameraFollowMode Mode { get; set; }
```

Property Value

[CameraFollowMode](#)

Offset

Gets or sets the offset applied to the camera position.

```
public Vector2 Offset { get; set; }
```

Property Value

[Vector2](#)

Position

Gets or sets the position of the camera.

```
public Vector2 Position { get; set; }
```

Property Value

[Vector2](#)

Rotation

Gets or sets the camera rotation in degrees.

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

Property Value

[float](#)

Size

Gets the size of the camera viewport.

```
public Rectangle Size { get; }
```

Property Value

Rectangle

Target

Gets or sets the target position the camera follows.

```
public Vector2 Target { get; set; }
```

Property Value

[Vector2](#) ↗

Transform

Gets the transform representation of the camera.

```
public Transform Transform { get; }
```

Property Value

Transform

Zoom

Gets or sets the zoom level of the camera.

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

Property Value

[float](#) ↗

Methods

Begin()

Begins rendering with this camera.

```
public void Begin()
```

End()

Ends rendering with this camera.

```
public void End()
```

GetCamera2D()

Gets the underlying Bliss.CSharp.Camera.Dim2.Cam2D instance used for camera logic.

```
public Cam2D GetCamera2D()
```

Returns

Cam2D

The internal Bliss.CSharp.Camera.Dim2.Cam2D instance.

GetScreenToWorld(Vector2)

Converts a screen position to world coordinates.

```
public Vector2 GetScreenToWorld(Vector2 position)
```

Parameters

position [Vector2](#)

The screen position.

Returns

[Vector2](#)

The corresponding world position.

GetView()

Gets the view matrix of the camera.

```
public Matrix4x4 GetView()
```

Returns

[Matrix4x4](#)

The view matrix.

GetVisibleArea()

Calculates and returns the visible area of the camera based on its current position and viewport size.

```
public RectangleF GetVisibleArea()
```

Returns

RectangleF

A Bliss.CSharp.Transformations.RectangleF representing the visible area in world coordinates.

GetWorldToScreen(Vector2)

Converts a world position to screen coordinates.

```
public Vector2 GetWorldToScreen(Vector2 position)
```

Parameters

position [Vector2](#)

The world position.

Returns

[Vector2](#)

The corresponding screen position.

Resize(Rectangle)

Resizes the camera to match the dimensions of the given rectangle.

```
protected override void Resize(Rectangle rectangle)
```

Parameters

rectangle Rectangle

The rectangle defining the new dimensions for the camera's viewport.

Update(double)

Updates the state of the [Camera2D](#) object on each frame.

```
protected override void Update(double delta)
```

Parameters

delta [double](#)

The time elapsed since the last frame, in seconds.

Class Camera3D

Namespace: [Sparkle.CSharp.Entities](#)

Assembly: Sparkle.dll

```
public class Camera3D : Entity, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Entity](#) ← Camera3D

Implements

[IDisposable](#)

Inherited Members

[Entity.Scene](#), [Entity.Id](#), [Entity.Tag](#), [Entity.Init\(\)](#), [Entity.AfterUpdate\(double\)](#), [Entity.FixedUpdate\(double\)](#),
[Entity.Draw\(GraphicsContext, Framebuffer\)](#), [Entity.GetComponents\(\)](#), [Entity.HasComponent<T>\(\)](#),
[Entity.GetComponent<T>\(\)](#), [Entity.TryGetComponent<T>\(out T\)](#), [Entity.AddComponent\(Component\)](#),
[Entity.TryAddComponent\(Component\)](#), [Entity.RemoveComponent\(Component\)](#),
[Entity.TryRemoveComponent\(Component\)](#), [Entity.RemoveComponent<T>\(\)](#),
[Entity.TryRemoveComponent<T>\(\)](#), [Entity.Dispose\(bool\)](#), [Disposable.Dispose\(\)](#),
[Disposable.ThrowIfDisposed\(\)](#), [Disposable.HasDisposed](#), [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Constructors

Camera3D(Vector3, Vector3, float, Vector3?, ProjectionType, CameraMode, float, float, float)

Initializes a new instance of the [Camera3D](#) class.

```
public Camera3D(Vector3 position, Vector3 target, float aspectRatio, Vector3? up = null,  
ProjectionType projectionType = ProjectionType.Perspective, CameraMode mode =  
CameraMode.Free, float fov = 70, float nearPlane = 0.1, float farPlane = 1000)
```

Parameters

position [Vector3](#)

The initial position of the camera.

target [Vector3](#)

The target point the camera is looking at.

aspectRatio [float](#)

The aspect ratio (width / height) of the camera view.

up [Vector3](#)?

The upward direction vector. Defaults to (0,1,0).

projectionType [ProjectionType](#)

The projection type of the camera (Perspective or Orthographic).

mode [CameraMode](#)

The movement mode of the camera.

fov [float](#)

The field of view in degrees. Should be between 1 and 179.

nearPlane [float](#)

The near clipping plane. Must be positive and smaller than **farPlane**.

farPlane [float](#)

The far clipping plane. Must be positive and greater than **nearPlane**.

Properties

AspectRatio

The aspect ratio of the camera.

```
public float AspectRatio { get; }
```

Property Value

[float](#)

FarPlane

The far clipping plane distance.

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

Property Value

[float](#)

Fov

The field of view of the camera.

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

Property Value

[float](#)

Mode

The movement mode of the camera.

```
public CameraMode Mode { get; set; }
```

Property Value

CameraMode

MouseSensitivity

The mouse sensitivity for camera movement.

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

Property Value

[float](#) ↗

MovementSpeed

The movement speed of the camera.

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

Property Value

[float](#) ↗

NearPlane

The near clipping plane distance.

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

Property Value

[float](#) ↗

OrbitalSpeed

The orbital speed of the camera.

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

Property Value

[float](#) ↗

Position

The position of the camera in world space.

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

Property Value

[Vector3](#)

ProjectionType

The projection type of the camera.

```
public ProjectionType ProjectionType { get; set; }
```

Property Value

ProjectionType

Target

The target position the camera is looking at.

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

Property Value

[Vector3](#)

Transform

Gets the transformation of the camera.

```
public Transform Transform { get; }
```

Property Value

Transform

Up

The up direction vector of the camera.

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

Property Value

[Vector3](#)

Methods

Begin()

Begins the camera rendering process.

```
public void Begin()
```

End()

Ends the camera rendering process.

```
public void End()
```

GetCamera3D()

Gets the underlying Bliss.CSharp.Camera.Dim3.Cam3D instance used for camera logic.

```
public Cam3D GetCamera3D()
```

Returns

Cam3D

The internal Bliss.CSharp.Camera.Dim3.Cam3D instance.

GetForward()

Gets the forward direction vector of the camera.

```
public Vector3 GetForward()
```

Returns

[Vector3](#)

GetFrustum()

Gets the camera's frustum, used for visibility and culling calculations.

```
public Frustum GetFrustum()
```

Returns

Frustum

GetPitch()

Gets the pitch (rotation around the right axis) of the camera.

```
public float GetPitch()
```

Returns

[float](#)

GetProjection()

Gets the projection matrix of the camera.

```
public Matrix4x4 GetProjection()
```

Returns

[Matrix4x4](#)

GetRight()

Gets the right direction vector of the camera.

```
public Vector3 GetRight()
```

Returns

[Vector3](#)

GetRoll()

Gets the roll (rotation around the forward axis) of the camera.

```
public float GetRoll()
```

Returns

[float](#)

GetRotation()

Gets the current rotation of the camera.

```
public Vector3 GetRotation()
```

Returns

[Vector3](#)

GetScreenToWorld(Vector2)

Converts a screen-space position to world space coordinates.

```
public Vector3 GetScreenToWorld(Vector2 position)
```

Parameters

position [Vector2](#)

The screen-space position.

Returns

[Vector3](#)

The corresponding world-space position.

GetView()

Gets the view matrix of the camera.

```
public Matrix4x4 GetView()
```

Returns

[Matrix4x4](#)

GetWorldToScreen(Vector3)

Converts a screen-space position to world space coordinates.

```
public Vector2 GetWorldToScreen(Vector3 position)
```

Parameters

position [Vector3](#)

The screen-space position.

Returns

[Vector2](#)

The corresponding world-space position.

GetYaw()

Gets the yaw (rotation around the up axis) of the camera.

```
public float GetYaw()
```

Returns

[float](#)

MoveForward(float, bool)

Moves the camera forward.

```
public void MoveForward(float distance, bool moveInWorldPlane)
```

Parameters

distance [float](#)

The distance to move forward.

moveInWorldPlane [bool](#)

Whether to move in the world plane.

MoveRight(float, bool)

Moves the camera to the right.

```
public void MoveRight(float distance, bool moveInWorldPlane)
```

Parameters

distance [float](#)

The distance to move right.

moveInWorldPlane [bool](#)

Whether to move in the world plane.

MoveToTarget(float)

Moves the camera closer to the target.

```
public void MoveToTarget(float distance)
```

Parameters

distance [float](#)

The distance to move.

MoveUp(float)

Moves the camera upwards.

```
public void MoveUp(float distance)
```

Parameters

distance [float](#)

The distance to move up.

Resize(Rectangle)

Resizes the camera to match the dimensions of the given rectangle.

```
protected override void Resize(Rectangle rectangle)
```

Parameters

rectangle Rectangle

The rectangle defining the new dimensions for the camera's viewport.

SetPitch(float, bool)

Sets the pitch (rotation around the right axis) of the camera.

```
public void SetPitch(float angle, bool rotateAroundTarget)
```

Parameters

angle float

The angle of rotation.

rotateAroundTarget bool

Whether to rotate around the target.

SetRoll(float)

Sets the roll (rotation around the forward axis) of the camera.

```
public void SetRoll(float angle)
```

Parameters

angle float

The angle of rotation.

SetYaw(float, bool)

Sets the yaw (rotation around the up axis) of the camera.

```
public void SetYaw(float angle, bool rotateAroundTarget)
```

Parameters

angle [float](#)

The angle of rotation.

rotateAroundTarget [bool](#)

Whether to rotate around the target.

Update(double)

Updates the state of the [Camera3D](#) object each frame.

```
protected override void Update(double delta)
```

Parameters

delta [double](#)

The time elapsed since the last frame update, in seconds.

Class Entity

Namespace: [Sparkle.CSharp.Entities](#)

Assembly: Sparkle.dll

```
public class Entity : Disposable, IDisposable
```

Inheritance

[object](#) ← Disposable ← Entity

Implements

[IDisposable](#)

Derived

[Camera2D](#), [Camera3D](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

Entity(Transform, string?)

Initializes a new instance of the [Entity](#) class.

```
public Entity(Transform transform, string? tag = null)
```

Parameters

transform Transform

The transform defining the position, rotation, and scale of the entity.

tag [string](#)

An optional tag used to categorize or identify the entity.

Fields

Tag

A tag used to categorize or identify the entity.

```
public string Tag
```

Field Value

[string](#)

Transform

The transform representing the position, rotation, and scale of the entity.

```
public Transform Transform
```

Field Value

Transform

Properties

Id

The unique identifier for this entity.

```
public uint Id { get; }
```

Property Value

[uint](#)

Scene

The scene that the entity belongs to.

```
public Scene Scene { get; }
```

Property Value

[Scene](#)

Methods

AddComponent(Component)

Adds a component to this entity.

```
public void AddComponent(Component component)
```

Parameters

component [Component](#)

The component to add.

Exceptions

[Exception](#) ↗

Thrown if the component is already attached to the entity.

AfterUpdate(double)

Called after the main update phase.

```
protected virtual void AfterUpdate(double delta)
```

Parameters

delta [double](#) ↗

The time delta since the last update.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

`disposing` [bool](#)

True if called from user code; false if called from a finalizer.

Draw(GraphicsContext, Framebuffer)

Renders the entity and its components using the provided graphics context and framebuffer.

```
protected virtual void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

`context` [GraphicsContext](#)

The graphics context that provides rendering utilities.

`framebuffer` [Framebuffer](#)

The framebuffer to which the entity will be rendered.

FixedUpdate(double)

Updates the entity and its components at fixed time intervals.

```
protected virtual void FixedUpdate(double fixedStep)
```

Parameters

fixedStep [double](#)

The fixed time step interval at which the update occurs.

GetComponent<T>()

Retrieves a specific component attached to this entity.

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

Returns

T

The component if found, otherwise null.

Type Parameters

T

The type of component to retrieve.

GetComponents()

Retrieves all components attached to the entity.

```
public IEnumerable<Component> GetComponents()
```

Returns

[IEnumerable](#) <[Component](#)>

An enumerable collection of components associated with the entity.

HasComponent<T>()

Checks if the entity has a specific type of component.

```
public bool HasComponent<T>() where T : Component
```

Returns

[bool](#)

True if the entity has the component, otherwise false.

Type Parameters

T

The type of component to check for.

[Init\(\)](#)

Called once when the entity is initialized.

```
protected virtual void Init()
```

[RemoveComponent\(Component\)](#)

Removes a component from this entity.

```
public void RemoveComponent(Component component)
```

Parameters

component [Component](#)

The component to remove.

Exceptions

[Exception](#)

Thrown if the component does not exist on this entity.

RemoveComponent<T>()

Removes a component of a specific type from this entity.

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

Type Parameters

T

The type of component to remove.

Exceptions

[Exception ↗](#)

Thrown if the component type does not exist on this entity.

Resize(Rectangle)

Called when the screen or viewport size changes.

```
protected virtual void Resize(Rectangle rectangle)
```

Parameters

rectangle Rectangle

The new screen or viewport dimensions.

TryAddComponent(Component)

Attempts to add a component to this entity.

```
public bool TryAddComponent(Component component)
```

Parameters

component [Component](#)

The component to add.

Returns

[bool](#) ↗

True if the component was added, otherwise false.

TryGetComponent<T>(out T?)

Attempts to retrieve a specific component attached to this entity.

```
public bool TryGetComponent<T>(out T? component) where T : Component
```

Parameters

component [T](#)

The retrieved component if found.

Returns

[bool](#) ↗

True if the component was found, otherwise false.

Type Parameters

T

The type of component to retrieve.

TryRemoveComponent(Component)

Attempts to remove a component from this entity.

```
public bool TryRemoveComponent(Component component)
```

Parameters

`component` [Component](#)

The component to remove.

Returns

`bool` ↗

True if the component was removed, otherwise false.

`TryRemoveComponent<T>()`

Attempts to remove a component of a specific type from this entity.

```
public bool TryRemoveComponent<T>() where T : Component
```

Returns

`bool` ↗

True if the component was removed, otherwise false.

Type Parameters

`T`

The type of component to remove.

`Update(double)`

Called every tick to update the entity's logic.

```
protected virtual void Update(double delta)
```

Parameters

`delta` [double](#) ↗

The time delta since the last update.

Namespace Sparkle.CSharp.Entities.Components

Classes

[Component](#)

[InstancedRenderProxy](#)

[InterpolatedComponent](#)

[MeshRenderer](#)

[ModelRenderer](#)

[RigidBody2D](#)

[RigidBody3D](#)

[SoftBody3D](#)

[Sprite](#)

Class Component

Namespace: [Sparkle.CSharp.Entities.Components](#)

Assembly: Sparkle.dll

```
public abstract class Component : Disposable, IDisposable
```

Inheritance

[object](#) ← Disposable ← Component

Implements

[IDisposable](#)

Derived

[InterpolatedComponent](#), [RigidBody2D](#), [RigidBody3D](#), [SoftBody3D](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

Component(Vector3)

Initializes a new instance of the [Component](#) class with a specified offset position.

```
protected Component(Vector3 offsetPosition)
```

Parameters

offsetPosition [Vector3](#)

The offset position relative to the entity's transform.

Fields

OffsetPosition

The local offset position relative to the entity's transform.

```
public Vector3 OffsetPosition
```

Field Value

[Vector3](#)

Properties

Entity

Gets or sets the entity to which this component is attached.

```
protected Entity Entity { get; }
```

Property Value

[Entity](#)

GlobalPosition

Gets the global position of the component, calculated as the entity's position plus the offset.

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

Property Value

[Vector3](#)

GraphicsDevice

Gets the graphics device used by the game.

```
public GraphicsDevice GraphicsDevice { get; }
```

Property Value

[GraphicsDevice](#)

InCompatibleTypes

The collection of component types that are incompatible.

```
public virtual IReadOnlyList<Type> InCompatibleTypes { get; }
```

Property Value

[IReadOnlyList](#) <[Type](#)>

Methods

AfterUpdate(double)

Called after the main update phase to handle additional logic.

```
protected virtual void AfterUpdate(double delta)
```

Parameters

delta [double](#)

The time delta since the last update.

ConflictsWith<T>(T)

Determines if the current component has a conflict with another specified component type.

```
public bool ConflictsWith<T>(T component) where T : Component
```

Parameters

component `T`

Returns

`bool` ↗

True if the current component conflicts with the specified component type; otherwise, false.

Type Parameters

`T`

The type of the component to check for conflicts.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing `bool` ↗

True if called from user code; false if called from a finalizer.

Draw(GraphicsContext, Framebuffer)

Called to render the component using the provided graphics context and framebuffer.

```
protected virtual void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context `GraphicsContext`

The graphics context used for rendering operations.

framebuffer [Framebuffer](#)

The framebuffer where the rendering occurs.

FixedUpdate(double)

Called at fixed time intervals to update the component's logic with a consistent step size.

```
protected virtual void FixedUpdate(double fixedStep)
```

Parameters

fixedStep [double](#)

The fixed time step for the update.

Init()

Called when the component is first initialized.

```
protected virtual void Init()
```

Resize(Rectangle)

Called when the window is resized.

```
protected virtual void Resize(Rectangle rectangle)
```

Parameters

rectangle Rectangle

The rectangle specifying the window's updated size.

Update(double)

Called every frame to update the component's logic.

```
protected virtual void Update(double delta)
```

Parameters

delta [double](#)

The time delta since the last update.

Class InstancedRenderProxy

Namespace: [Sparkle.CSharp.Entities.Components](#)

Assembly: Sparkle.dll

```
public class InstancedRenderProxy : InterpolatedComponent, IDisposable
```

Inheritance

[object](#) ← Disposable ← Component ← InterpolatedComponent ← InstancedRenderProxy

Implements

[IDisposable](#)

Inherited Members

[InterpolatedComponent.LerpedPosition](#) , [InterpolatedComponent.LerpedGlobalPosition](#) ,
[InterpolatedComponent.LerpedRotation](#) , [InterpolatedComponent.LerpedScale](#) ,
[InterpolatedComponent.FixedUpdate\(double\)](#) , [Component.GraphicsDevice](#) , [Component.Entity](#) ,
[Component.InCompatibleTypes](#) , [Component.GlobalPosition](#) , [Component.OffsetPosition](#) ,
[Component.Update\(double\)](#) , [Component.AfterUpdate\(double\)](#) ,
[Component.Draw\(GraphicsContext, Framebuffer\)](#) , [Component.Resize\(Rectangle\)](#) ,
[Component.ConflictsWith<T>\(T\)](#) , [Disposable.Dispose\(\)](#) , [Disposable.ThrowIfDisposed\(\)](#) ,
[Disposable.HasDisposed](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

InstancedRenderProxy(MultiInstanceRenderer, Vector3, bool, Color?)

Initializes a new instance of the [InstancedRenderProxy](#) class.

```
public InstancedRenderProxy(MultiInstanceRenderer multiInstanceRenderer, Vector3  
offsetPosition, bool drawBox = false, Color? boxColor = null)
```

Parameters

multiInstanceRenderer [MultiInstanceRenderer](#)

The multi-instance renderer responsible for drawing this proxy.

offsetPosition [Vector3](#)

The local position offset applied to the proxy transform.

drawBox [bool](#)

If true, enables rendering of the instance bounding box.

boxColor [Color?](#)

Optional color used to render the bounding box.

Fields

BoxColor

Specifies the color used when rendering the bounding box.

```
public Color BoxColor
```

Field Value

Color

DrawBox

Determines whether the bounding box for this instance should be drawn.

```
public bool DrawBox
```

Field Value

[bool](#)

Properties

MultinstanceRenderer

Gets the multi-instance renderer that owns this render proxy.

```
public MultiInstanceRenderer MultiInstanceRenderer { get; }
```

Property Value

[MultinstanceRenderer](#)

Methods

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

GetRenderableBoneMatricesByMesh(Mesh)

Retrieves the bone matrix array used by the renderable associated with the specified mesh.

```
public Matrix4x4[]? GetRenderableBoneMatricesByMesh(Mesh mesh)
```

Parameters

mesh Mesh

The mesh whose bone matrices are requested.

Returns

[Matrix4x4](#) []

The bone matrix array if the mesh is skinned, otherwise null.

GetRenderableMaterialByMesh(Mesh)

Retrieves a reference to the material used by the renderable associated with the specified mesh.

```
public ref Material GetRenderableMaterialByMesh(Mesh mesh)
```

Parameters

mesh Mesh

The mesh whose renderable material is requested.

Returns

Material

A reference to the material used by the mesh renderable.

Init()

Initializes the render proxy and registers it with the multi-instance renderer.

```
protected override void Init()
```

UpdateFrustumBox()

Updates the frustum-aligned bounding box based on the interpolated transform state.

```
protected void UpdateFrustumBox()
```

Class InterpolatedComponent

Namespace: [Sparkle.CSharp.Entities.Components](#)

Assembly: Sparkle.dll

```
public abstract class InterpolatedComponent : Component, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Component](#) ← InterpolatedComponent

Implements

[IDisposable](#)

Derived

[InstancedRenderProxy](#), [MeshRenderer](#), [ModelRenderer](#), [Sprite](#)

Inherited Members

[Component.GraphicsDevice](#), [Component.Entity](#), [Component.InCompatibleTypes](#),
[Component.GlobalPosition](#), [Component.OffsetPosition](#), [Component.Init\(\)](#), [Component.Update\(double\)](#),
[Component.AfterUpdate\(double\)](#), [Component.Draw\(GraphicsContext, Framebuffer\)](#),
[Component.Resize\(Rectangle\)](#), [Component.ConflictsWith<T>\(T\)](#), [Component.Dispose\(bool\)](#),
Disposable.Dispose(), Disposable.ThrowIfDisposed(), Disposable.HasDisposed, [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Constructors

InterpolatedComponent(Vector3)

Initializes a new instance of the [InterpolatedComponent](#) class.

```
protected InterpolatedComponent(Vector3 offsetPosition)
```

Parameters

offsetPosition [Vector3](#)

The offset position relative to the entity's transform.

Properties

LerpedGlobalPosition

Interpolated global position.

```
public Vector3 LerpedGlobalPosition { get; }
```

Property Value

[Vector3](#)

LerpedPosition

Interpolated local position.

```
public Vector3 LerpedPosition { get; }
```

Property Value

[Vector3](#)

LerpedRotation

Interpolated rotation.

```
public Quaternion LerpedRotation { get; }
```

Property Value

[Quaternion](#)

LerpedScale

Interpolated scale.

```
public Vector3 LerpedScale { get; }
```

Property Value

[Vector3](#) ↗

Methods

FixedUpdate(double)

Updates the interpolated component's state in fixed intervals, aligning the current and previous transform properties (position, rotation, and scale) with the associated entity's state.

```
protected override void FixedUpdate(double fixedStep)
```

Parameters

fixedStep [double](#) ↗

The fixed time step duration, typically used for physics updates.

Class MeshRenderer

Namespace: [Sparkle.CSharp.Entities.Components](#)

Assembly: Sparkle.dll

```
public class MeshRenderer : InterpolatedComponent, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Component](#) ← [InterpolatedComponent](#) ← MeshRenderer

Implements

[IDisposable](#)

Inherited Members

[InterpolatedComponent.LerpedPosition](#) , [InterpolatedComponent.LerpedGlobalPosition](#) ,
[InterpolatedComponent.LerpedRotation](#) , [InterpolatedComponent.LerpedScale](#) ,
[InterpolatedComponent.FixedUpdate\(double\)](#) , [Component.GraphicsDevice](#) , [Component.Entity](#) ,
[Component.InCompatibleTypes](#) , [Component.GlobalPosition](#) , [Component.OffsetPosition](#) ,
[Component.Init\(\)](#) , [Component.Update\(double\)](#) , [Component.AfterUpdate\(double\)](#) ,
[Component.Resize\(Rectangle\)](#) , [Component.ConflictsWith<T>\(T\)](#) , [Component.Dispose\(bool\)](#) ,
Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

MeshRenderer(Mesh, Vector3, Material, bool, Color?)

Initializes a new instance of the [MeshRenderer](#) class.

```
public MeshRenderer(Mesh mesh, Vector3 offsetPosition, Material material, bool drawBox =  
false, Color? boxColor = null)
```

Parameters

mesh Mesh

The mesh to be rendered.

`offsetPosition` [Vector3](#)

The positional offset applied to the renderer.

`material` Material

The material used to render the mesh.

`drawBox` [bool](#)

Whether to render the bounding box for debugging.

`boxColor` Color?

The color used to render the bounding box.

MeshRenderer(Mesh, Vector3, bool, bool, Color?)

Initializes a new instance of the [MeshRenderer](#) class.

```
public MeshRenderer(Mesh mesh, Vector3 offsetPosition, bool copyMeshMaterial = false, bool drawBox = false, Color? boxColor = null)
```

Parameters

`mesh` Mesh

The mesh to be rendered.

`offsetPosition` [Vector3](#)

The positional offset applied to the renderer.

`copyMeshMaterial` [bool](#)

Whether to clone the mesh material for independent modification.

`drawBox` [bool](#)

Whether to render the bounding box for debugging.

`boxColor` Color?

The color used to render the bounding box.

Fields

BoxColor

The color used for rendering the bounding box of the mesh.

```
public Color BoxColor
```

Field Value

Color

DrawBox

Whether the bounding box should be rendered for the associated mesh.

```
public bool DrawBox
```

Field Value

[bool](#)

Properties

BoneMatrics

A reference to the bone matrices for skeletal animation, if applicable.

```
public Matrix4x4[]? BoneMatrics { get; }
```

Property Value

[Matrix4x4](#)[]

Material

A reference to the material used for rendering the mesh.

```
public ref Material Material { get; }
```

Property Value

Material

Mesh

The 3D mesh to render.

```
public Mesh Mesh { get; }
```

Property Value

Mesh

Methods

Draw(GraphicsContext, Framebuffer)

Draws the mesh and optionally its wireframe and bounding box to the screen.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for issuing draw commands.

framebuffer [Framebuffer](#)

The framebuffer to which the mesh should be rendered.

Class ModelRenderer

Namespace: [Sparkle.CSharp.Entities.Components](#)

Assembly: Sparkle.dll

```
public class ModelRenderer : InterpolatedComponent, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Component](#) ← [InterpolatedComponent](#) ← ModelRenderer

Implements

[IDisposable](#)

Inherited Members

[InterpolatedComponent.LerpedPosition](#) , [InterpolatedComponent.LerpedGlobalPosition](#) ,
[InterpolatedComponent.LerpedRotation](#) , [InterpolatedComponent.LerpedScale](#) ,
[InterpolatedComponent.FixedUpdate\(double\)](#) , [Component.GraphicsDevice](#) , [Component.Entity](#) ,
[Component.InCompatibleTypes](#) , [Component.GlobalPosition](#) , [Component.OffsetPosition](#) ,
[Component.Init\(\)](#) , [Component.Update\(double\)](#) , [Component.AfterUpdate\(double\)](#) ,
[Component.Resize\(Rectangle\)](#) , [Component.ConflictsWith<T>\(T\)](#) , [Component.Dispose\(bool\)](#) ,
Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

ModelRenderer(Model, Vector3, bool, bool, Color?)

Initializes a new instance of the [ModelRenderer](#) class.

```
public ModelRenderer(Model model, Vector3 offsetPosition, bool copyModelMaterials = false,  
bool drawBox = false, Color? boxColor = null)
```

Parameters

model Model

The model containing the meshes to be rendered.

`offsetPosition` [Vector3](#)

The positional offset applied to the model renderer.

`copyModelMaterials` [bool](#)

Whether to clone the model's materials so they can be modified independently.

`drawBox` [bool](#)

Whether to render the bounding box for debugging.

`boxColor` [Color?](#)

The color used to render the bounding box.

Fields

BoxColor

The color used for rendering the bounding box of the model.

```
public Color BoxColor
```

Field Value

Color

DrawBox

Whether the bounding box should be rendered for the associated model.

```
public bool DrawBox
```

Field Value

[bool](#)

Properties

Model

The model to be rendered.

```
public Model Model { get; }
```

Property Value

Model

Methods

Draw(GraphicsContext, Framebuffer)

Renders the model associated with this component to the specified framebuffer if it is within the camera frustum.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering.

framebuffer [Framebuffer](#)

The framebuffer to render into.

GetRenderableBoneMatricesByMesh(Mesh)

Retrieves the bone matrices for a specified mesh.

```
public Matrix4x4[]? GetRenderableBoneMatricesByMesh(Mesh mesh)
```

Parameters

mesh Mesh

The mesh for which the bone matrices are to be retrieved.

Returns

[Matrix4x4\[\]](#)

A reference to an array of bone matrices associated with the specified mesh, or null if no matrices exist.

GetRenderableMaterialByMesh(Mesh)

Retrieves the material associated with the specified mesh.

```
public ref Material GetRenderableMaterialByMesh(Mesh mesh)
```

Parameters

mesh Mesh

The mesh for which the material is to be retrieved.

Returns

Material

A reference to the material associated with the specified mesh.

Class RigidBody2D

Namespace: [Sparkle.CSharp.Entities.Components](#)

Assembly: Sparkle.dll

```
public class RigidBody2D : Component, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Component](#) ← RigidBody2D

Implements

[IDisposable](#)

Inherited Members

[Component.GraphicsDevice](#) , [Component.Entity](#) , [Component.GlobalPosition](#) ,
[Component.Update\(double\)](#) , [Component.AfterUpdate\(double\)](#) , [Component.FixedUpdate\(double\)](#) ,
[Component.Draw\(GraphicsContext, Framebuffer\)](#) , [Component.Resize\(Rectangle\)](#) ,
[Component.ConflictsWith<T>\(I\)](#) , [Disposable.Dispose\(\)](#) , [Disposable.ThrowIfDisposed\(\)](#) ,
[Disposable.HasDisposed](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

RigidBody2D(BodyDefinition, IShape2D)

Initializes a new instance of the [RigidBody2D](#) class with a single collision shape.

```
public RigidBody2D(BodyDefinition bodyDef, IShape2D shape)
```

Parameters

bodyDef [BodyDefinition](#)

The body definition containing physical properties.

shape [IShape2D](#)

The collision shape to attach to the body.

RigidBody2D(BodyDefinition, List<IShape2D>)

Initializes a new instance of the [RigidBody2D](#) class with multiple collision shapes.

```
public RigidBody2D(BodyDefinition bodyDef, List<IShape2D> shapes)
```

Parameters

bodyDef [BodyDefinition](#)

The body definition containing physical properties.

shapes [List](#)<[IShape2D](#)>

The list of collision shapes to attach to the body.

Properties

AABB

Gets the axis-aligned bounding box (AABB) of the body.

```
public AABB AABB { get; }
```

Property Value

AABB

AngularDamping

Gets or sets the angular damping of the rigid body, which reduces the angular velocity over time to simulate rotational drag.

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

Property Value

[float](#)

AngularVelocity

Gets or sets the angular velocity of the body in radians per second.

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

Property Value

[float](#)

Awake

Gets or sets a value indicating whether the rigid body is awake.

```
public bool Awake { get; set; }
```

Property Value

[bool](#)

Body

Gets the internal physics body.

```
public Body Body { get; }
```

Property Value

Body

Bullet

Gets or sets a value indicating whether the body should use continuous collision detection.

```
public bool Bullet { get; set; }
```

Property Value

[bool](#)

Contacts

Gets the contact data for the body.

```
public ReadOnlySpan<ContactData> Contacts { get; }
```

Property Value

[ReadOnlySpan](#)<ContactData>

DefaultComparer

Gets the default comparer used for sorting bodies.

```
public static IComparer<Body> DefaultComparer { get; }
```

Property Value

[IComparer](#)<Body>

DefaultEqualityComparer

Gets the default equality comparer for body instances.

```
public static IEqualityComparer<Body> DefaultEqualityComparer { get; }
```

Property Value

[IEqualityComparer](#)<Body>

Enabled

Gets or sets a value indicating whether the body is enabled.

```
public bool Enabled { get; set; }
```

Property Value

[bool](#)

FixedRotation

Gets or sets a value indicating whether the body is prevented from rotating.

```
public bool FixedRotation { get; set; }
```

Property Value

[bool](#)

GravityScale

Gets or sets the factor by which the global gravity is scaled for this body.

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

Property Value

[float](#)

InCompatibleTypes

Gets a list of component types that are incompatible with this component.

```
public override IReadOnlyList<Type> InCompatibleTypes { get; }
```

Property Value

[IReadOnlyList](#)<[Type](#)>

IsValid

Gets a value indicating whether the internal body is valid.

```
public bool IsValid { get; }
```

Property Value

[bool](#)

Joints

Gets the joints attached to the body.

```
public ReadOnlySpan<Joint> Joints { get; }
```

Property Value

[ReadOnlySpan](#)<[Joint](#)>

LinearDamping

Gets or sets the linear damping of the body, which reduces its linear velocity over time.

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

Property Value

[float](#)

LinearVelocity

Gets or sets the linear velocity of the rigid body in 2D space.

```
public Vector2 LinearVelocity { get; set; }
```

Property Value

[Vector2](#)

LocalCenterOfMass

Gets the local center of mass of the body.

```
public Vector2 LocalCenterOfMass { get; }
```

Property Value

[Vector2](#)

Mass

Gets the mass of the body.

```
public float Mass { get; }
```

Property Value

[float](#)

MassData

Gets or sets the mass data of the rigid body, including mass, center of mass, and rotational inertia.

```
public MassData MassData { get; set; }
```

Property Value

Name

Gets or sets the name of the body.

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

Property Value

[string](#)

OffsetPosition

The positional offset of this component, always zero for Rigidbody2D.

```
public Vector3 OffsetPosition { get; }
```

Property Value

[Vector3](#)

Position

Gets the current world position of the body.

```
public Vector2 Position { get; }
```

Property Value

[Vector2](#)

Rotation

Gets the rotation of the body.

```
public Rotation Rotation { get; }
```

Property Value

Rotation

RotationalInertia

Gets the rotational inertia of the body.

```
public float RotationalInertia { get; }
```

Property Value

[float](#)

Shapes

Gets the shapes attached to the body.

```
public ReadOnlySpan<Shape> Shapes { get; }
```

Property Value

[ReadOnlySpan](#)<Shape>

Simulation

Gets the current 2D simulation instance used by the scene. Throws an [InvalidOperationException](#) if the simulation is not of type [Simulation2D](#).

```
public Simulation2D Simulation { get; }
```

Property Value

SleepEnabled

Gets or sets a value indicating whether the body can enter a sleep state when idle.

```
public bool SleepEnabled { get; set; }
```

Property Value

[bool](#)

SleepThreshold

Gets or sets the threshold value that determines the conditions under which the body can enter sleep mode.

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

Property Value

[float](#)

Transform

Gets or sets the transformation of the body, including its position and rotation in the world space.

```
public Transform Transform { get; set; }
```

Property Value

Transform

Type

Gets or sets the type of the rigid body, determining its physical behavior (e.g., static, dynamic, or kinematic).

```
public BodyType Type { get; set; }
```

Property Value

BodyType

UserData

Gets or sets user-defined data associated with the rigid body.

```
public object? UserData { get; set; }
```

Property Value

object ↗

World

Gets the physics world from the current 2D simulation.

```
public World World { get; }
```

Property Value

World

WorldCenterOfMass

Gets the local center of mass of the body.

```
public Vector2 WorldCenterOfMass { get; }
```

Property Value

[Vector2](#) ↗

Methods

ApplyAngularImpulse(float, bool)

Applies an angular impulse to the rigid body, affecting its angular velocity.

```
public void ApplyAngularImpulse(float impulse, bool wake)
```

Parameters

impulse [float](#) ↗

The magnitude of the angular impulse to apply.

wake [bool](#) ↗

A boolean indicating whether the body should be woken up if it is sleeping.

ApplyForce(Vector2, Vector2, bool)

Applies a force to the rigid body at a specified point.

```
public void ApplyForce(Vector2 force, Vector2 point, bool wake)
```

Parameters

force [Vector2](#) ↗

The force vector to apply.

point [Vector2](#) ↗

The world position where the force is applied.

wake [bool](#) ↗

Indicates whether to wake up the body, if it is sleeping.

ApplyForceToCenter(Vector2, bool)

Applies a force to the center of the rigid body, influencing its velocity.

```
public void ApplyForceToCenter(Vector2 force, bool wake)
```

Parameters

force [Vector2](#)

The two-dimensional vector defining the magnitude and direction of the force to apply.

wake [bool](#)

A boolean indicating whether to wake up the body if it is currently sleeping.

ApplyLinearImpulse(Vector2, Vector2, bool)

Applies a linear impulse to the body at a specific point in world coordinates.

```
public void ApplyLinearImpulse(Vector2 impulse, Vector2 point, bool wake)
```

Parameters

impulse [Vector2](#)

The impulse vector to apply to the body.

point [Vector2](#)

The world position where the impulse is applied.

wake [bool](#)

A value indicating whether to wake the body if it is currently sleeping.

ApplyLinearImpulseToCenter(Vector2, bool)

Applies a linear impulse to the center of mass of the rigid body.

```
public void ApplyLinearImpulseToCenter(Vector2 impulse, bool wake)
```

Parameters

impulse [Vector2](#)

The vector representing the magnitude and direction of the impulse to apply.

wake [bool](#)

A boolean indicating whether to wake the body if it is currently sleeping.

ApplyTorque(float, bool)

Applies a torque to the rigid body, affecting its angular velocity.

```
public void ApplyTorque(float torque, bool wake)
```

Parameters

torque [float](#)

The torque to apply, measured in Newton-meters.

wake [bool](#)

Determines whether to wake the body if it is currently sleeping.

CreateChain(ChainDef)

Creates a new chain shape based on the specified chain definition.

```
public ChainShape CreateChain(ChainDef def)
```

Parameters

def [ChainDef](#)

The chain definition containing the configuration details for the chain shape.

Returns

ChainShape

A Box2D.ChainShape instance representing the created chain shape.

CreateShape(ShapeDef, Capsule)

Creates a new shape and attaches it to the body based on the provided definition and capsule geometry.

```
public Shape CreateShape(ShapeDef def, Capsule capsule)
```

Parameters

def ShapeDef

The shape definition containing attributes such as density, friction, and filter information.

capsule Capsule

The capsule geometry describing the shape's dimensions and orientation.

Returns

Shape

A Box2D.Shape instance representing the newly created shape.

CreateShape(ShapeDef, Circle)

Creates a new shape and attaches it to the body using the specified shape definition and geometry.

```
public Shape CreateShape(ShapeDef def, Circle circle)
```

Parameters

def ShapeDef

The shape definition containing physical and configuration data for the shape.

circle Circle

The circle geometry representing the shape.

Returns

Shape

The created shape instance.

CreateShape(ShapeDef, Polygon)

Creates a new shape and attaches it to the rigid body using the provided shape definition and polygon data.

```
public Shape CreateShape(ShapeDef def, Polygon polygon)
```

Parameters

def ShapeDef

The shape definition containing properties for the new shape.

polygon Polygon

The polygon data defining the geometry of the shape.

Returns

Shape

CreateShape(ShapeDef, Segment)

Creates a new shape for the rigid body using the specified shape definition and segment.

```
public Shape CreateShape(ShapeDef def, Segment segment)
```

Parameters

def ShapeDef

The shape definition that specifies the properties of the shape.

segment Segment

The segment used to define specific geometry of the shape.

Returns

Shape

The created shape instance.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

EnableContactEvents(bool)

Enables or disables contact events for the rigid body.

```
public void EnableContactEvents(bool flags)
```

Parameters

flags [bool](#)

A boolean value indicating whether contact events should be enabled or disabled.

EnableHitEvents(bool)

Enables or disables hit event handling for the rigid body.

```
public void EnableHitEvents(bool flags)
```

Parameters

flags [bool](#)

A boolean value indicating whether hit events should be enabled (true) or disabled (false).

GetLocalPoint(Vector2)

Converts a point in world coordinates to a point in the local space of the body.

```
public Vector2 GetLocalPoint(Vector2 worldPoint)
```

Parameters

worldPoint [Vector2](#)

The point in world coordinates to be transformed to local coordinates.

Returns

[Vector2](#)

The transformed point in the local coordinates of the body.

GetLocalPointVelocity(Vector2)

Calculates the velocity at a given local point on the rigid body.

```
public Vector2 GetLocalPointVelocity(Vector2 localPoint)
```

Parameters

`localPoint` [Vector2](#)

The point in local coordinates for which to calculate the velocity.

Returns

[Vector2](#)

The velocity at the specified local point as a [Vector2](#).

GetLocalVector(Vector2)

Converts a vector from world coordinates to local coordinates relative to this body.

```
public Vector2 GetLocalVector(Vector2 worldVector)
```

Parameters

`worldVector` [Vector2](#)

The vector in world coordinates to be converted to local coordinates.

Returns

[Vector2](#)

The converted vector in local coordinates.

GetWorldPoint(Vector2)

Transforms the specified local point to world coordinates relative to the rigid body.

```
public Vector2 GetWorldPoint(Vector2 localPoint)
```

Parameters

`localPoint` [Vector2](#)

The point in local coordinates to be transformed into world coordinates.

Returns

[Vector2](#)

A [Vector2](#) representing the transformed point in world coordinates.

GetWorldPointVelocity(Vector2)

Calculates the velocity of the rigid body at a specific world point.

```
public Vector2 GetWorldPointVelocity(Vector2 worldPoint)
```

Parameters

[worldPoint](#) [Vector2](#)

The point in world coordinates where velocity is to be calculated.

Returns

[Vector2](#)

The velocity of the rigid body at the specified world point.

GetWorldVector(Vector2)

Converts a given vector from local space to world space.

```
public Vector2 GetWorldVector(Vector2 localVector)
```

Parameters

[localVector](#) [Vector2](#)

The vector defined in the local coordinate system.

Returns

[Vector2](#)

The vector transformed into world space coordinates.

Init()

Initializes the [RigidBody2D](#) component.

```
protected override void Init()
```

SetTargetTransform(Transform, float)

Sets the target transform for the rigid body to interpolate towards over a specified time step.

```
public void SetTargetTransform(Transform target, float timeStep)
```

Parameters

target Transform

The desired target transform for the rigid body.

timeStep float

The time step over which the rigid body interpolates towards the target transform.

Class RigidBody3D

Namespace: [Sparkle.CSharp.Entities.Components](#)

Assembly: Sparkle.dll

```
public class RigidBody3D : Component, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Component](#) ← RigidBody3D

Implements

[IDisposable](#)

Inherited Members

[Component.GraphicsDevice](#) , [Component.Entity](#) , [Component.GlobalPosition](#) ,
[Component.Update\(double\)](#) , [Component.AfterUpdate\(double\)](#) , [Component.FixedUpdate\(double\)](#) ,
[Component.Draw\(GraphicsContext, Framebuffer\)](#) , [Component.Resize\(Rectangle\)](#) ,
[Component.ConflictsWith<T>\(I\)](#) , [Disposable.Dispose\(\)](#) , [Disposable.ThrowIfDisposed\(\)](#) ,
[Disposable.HasDisposed](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

RigidBody3D(RigidBodyShape, bool, MotionType, float, float, bool)

Initializes a new instance of the [RigidBody3D](#) class with a single collision shape.

```
public RigidBody3D(RigidBodyShape shape, bool setMassInertia = true, MotionType motionType = MotionType.Dynamic, float friction = 0.2, float restitution = 0, bool drawDebug = false)
```

Parameters

shape RigidBodyShape

The collision shape to attach to the rigid body.

setMassInertia [bool](#)

Determines whether to automatically calculate mass and inertia from the shape.

motionType MotionType

Specifies whether the rigid body should be dynamic, static, or kinematic in the simulation.

friction [float](#)

The friction coefficient of the rigid body.

restitution [float](#)

The restitution (bounciness) of the rigid body.

drawDebug [bool](#)

Indicates whether debug visualization for the rigid body should be enabled.

RigidBody3D(List<RigidBodyShape>, bool, MotionType, float, float, bool)

Initializes a new instance of the [RigidBody3D](#) class with a collection of collision shapes.

```
public RigidBody3D(List<RigidBodyShape> shapes, bool setMassInertia = true, MotionType motionType = MotionType.Dynamic, float friction = 0.2, float restitution = 0, bool drawDebug = false)
```

Parameters

shapes [List](#)<RigidBodyShape>

The list of collision shapes to attach to the rigid body.

setMassInertia [bool](#)

Determines whether to automatically calculate mass and inertia from the shapes.

motionType MotionType

Specifies whether the rigid body should be dynamic, static, or kinematic in the simulation.

friction [float](#)

The friction coefficient of the rigid body.

restitution [float](#)

The restitution (bounciness) of the rigid body.

drawDebug [bool](#)

Indicates whether debug visualization for the rigid body should be enabled.

Fields

DebugDrawColor

The color used for debugging visualization of the 3D rigid body's properties and behavior.

```
public Color DebugDrawColor
```

Field Value

Color

DrawDebug

Whether debug information, such as visual representations of physics-related elements, should be displayed for the rigid body.

```
public bool DrawDebug
```

Field Value

[bool](#)

Properties

AffectedByGravity

Determines whether the rigid body is affected by gravity.

```
public bool AffectedByGravity { get; set; }
```

Property Value

[bool](#)

AngularVelocity

Gets or sets the angular velocity of the rigid body.

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

Property Value

[Vector3](#)

Body

Gets the underlying main rigid body instance used in the simulation.

```
public RigidBody Body { get; }
```

Property Value

RigidBody

BodyId

Gets the unique ID of the rigid body.

```
public ulong BodyId { get; }
```

Property Value

[ulong](#)

Connections

Gets all bodies connected to this one.

```
public ReadOnlyList<RigidBody> Connections { get; }
```

Property Value

ReadOnlyList<RigidBody>

Constraints

Gets all constraints applied to this rigid body.

```
public ReadOnlyHashSet<Constraint> Constraints { get; }
```

Property Value

ReadOnlyHashSet<Constraint>

Damping

Damping factor for angular and linear motion.

```
public (float angular, float linear) Damping { get; set; }
```

Property Value

([float](#) angular, [float](#) linear)

Data

Gets a reference to the physical data of the rigid body.

```
public ref RigidBodyData Data { get; }
```

Property Value

RigidBodyData

DeactivationThreshold

Sets the angular and linear deactivation thresholds.

```
public (float angular, float linear) DeactivationThreshold { set; }
```

Property Value

([float](#), [angular](#), [float](#), [linear](#))

DeactivationTime

Time before the body is considered inactive when not moving.

```
public TimeSpan DeactivationTime { get; set; }
```

Property Value

[TimeSpan](#)

EnableSpeculativeContacts

Enables or disables speculative contact solving for better collision detection.

```
public bool EnableSpeculativeContacts { get; set; }
```

Property Value

[bool](#)

Force

Gets or sets the applied force on the rigid body.

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

Property Value

[Vector3](#)

Friction

Friction coefficient of the rigid body.

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

Property Value

[float](#)

Handle

Gets the handle to the rigid body data.

```
public JHandle<RigidBodyData> Handle { get; }
```

Property Value

[JHandle<RigidBodyData>](#)

InCompatibleTypes

Gets a list of component types that are incompatible with this component.

```
public override IReadOnlyList<Type> InCompatibleTypes { get; }
```

Property Value

[IReadOnlyList](#)<[Type](#)>

InverseInertia

Gets the inverse inertia tensor of the rigid body.

```
public Matrix4x4 InverseInertia { get; }
```

Property Value

[Matrix4x4](#)

IsActive

Returns whether the rigid body is currently active in the simulation.

```
public bool IsActive { get; }
```

Property Value

[bool](#)

IsStatic

Specifies if the rigid body is static (immovable).

```
[Obsolete("Use the MotionType property instead.")]  
public bool IsStatic { get; set; }
```

Property Value

[bool](#)

Island

Gets the island this rigid body is part of (for sleeping management).

```
public Island Island { get; }
```

Property Value

Island

Mass

Gets the mass of the rigid body.

```
public float Mass { get; }
```

Property Value

[float](#)

MotionType

Gets or sets the motion type of the rigid body, determining its dynamic behavior within the physics simulation (e.g., static, dynamic, or kinematic).

```
public MotionType MotionType { get; set; }
```

Property Value

MotionType

OffsetPosition

The positional offset of this component, always zero for Rigidbody3D.

```
public Vector3 OffsetPosition { get; }
```

Property Value

[Vector3](#)

Orientation

Gets or sets the orientation of the rigid body.

```
public Quaternion Orientation { get; set; }
```

Property Value

[Quaternion](#)

Position

Gets or sets the position of the rigid body.

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

Property Value

[Vector3](#)

Restitution

Restitution (bounciness) of the rigid body.

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

Property Value

[float](#)

Shapes

Gets the shapes associated with the rigid body.

```
public ReadOnlyList<RigidBodyShape> Shapes { get; }
```

Property Value

ReadOnlyList<RigidBodyShape>

Simulation

Gets the current 3D simulation instance used by the scene. Throws an [InvalidOperationException](#) if the simulation is not of type [Simulation3D](#).

```
public Simulation3D Simulation { get; }
```

Property Value

[Simulation3D](#)

Tag

A custom user tag associated with the rigid body.

```
public object? Tag { get; set; }
```

Property Value

[object](#)

Torque

Gets or sets the applied torque on the rigid body.

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

Property Value

[Vector3](#)

Velocity

Gets or sets the linear velocity of the rigid body.

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

Property Value

[Vector3](#)

World

Gets the physics world from the current 3D simulation.

```
public World World { get; }
```

Property Value

World

Methods

AddForce(Vector3)

Applies a force to the rigid body.

```
public void AddForce(Vector3 force)
```

Parameters

force [Vector3](#)

The force vector to apply to the rigid body.

AddForce(Vector3, Vector3)

Applies a force at a specific position on the rigid body.

```
public void AddForce(Vector3 force, Vector3 position)
```

Parameters

force [Vector3](#)

The force vector to apply to the rigid body.

position [Vector3](#)

The position at which the force is applied, relative to the rigid body's center of mass.

AddShape(RigidBodyShape, bool)

Adds a new collision shape to the rigid body.

```
public void AddShape(RigidBodyShape shape, bool setMassInertia = true)
```

Parameters

shape RigidBodyShape

The specific collision shape to be added.

setMassInertia [bool](#)

Indicates whether the mass and inertia should be recalculated after adding the shape.

AddShape(IEnumerable<RigidBodyShape>, bool)

Adds a collision shape to the rigid body.

```
public void AddShape(IEnumerable<RigidBodyShape> shapes, bool setMassInertia = true)
```

Parameters

shapes [IEnumerable](#)<RigidBodyShape>

The collection of shapes to add.

setMassInertia [bool](#)

Determines whether to automatically calculate mass and inertia after adding the shape.

ClearShapes(bool)

Clears all collision shapes attached to the rigid body.

```
[Obsolete("ClearShapes is deprecated, please use RemoveShape instead.")]  
public void ClearShapes(bool setMassInertia = true)
```

Parameters

setMassInertia [bool](#)

Determines whether to recalculate mass and inertia after clearing the shapes.

DebugDraw(IDebugDrawer)

Renders debug information for the rigid body using the specified debug drawer.

```
public void DebugDraw(IDebugDrawer drawer)
```

Parameters

drawer IDebugDrawer

The debug drawer instance used to render the debug information.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Init()

Initializes the rigid body and registers it into the physics world.

```
protected override void Init()
```

RemoveShape(RigidBodyShape, bool)

Removes a collision shape from the rigid body.

```
public void RemoveShape(RigidBodyShape shape, bool setMassInertia = true)
```

Parameters

shape RigidBodyShape

The collision shape to be removed from the rigid body.

setMassInertia [bool](#)

Determines whether to recalculate mass and inertia after removing the shape.

RemoveShape(IEnumerable<RigidBodyShape>, bool)

Removes the specified collision shape from the rigid body.

```
public void RemoveShape(IEnumerable<RigidBodyShape> shapes, bool setMassInertia = true)
```

Parameters

shapes [IEnumerable](#)<RigidBodyShape>

The collection of shapes to remove.

setMassInertia [bool](#)

Determines whether to automatically recalculate mass and inertia after removing the shape.

SetActivationState(bool)

Sets the activation state of the rigid body.

```
public void SetActivationState(bool active)
```

Parameters

active [bool](#)

Indicates whether the rigid body should be activated or deactivated.

SetMassInertia()

Updates the mass and inertia of the rigid body based on its current shapes and associated properties.

```
public void SetMassInertia()
```

SetMassInertia(Matrix4x4, float, bool)

Sets the mass and inertia of the rigid body.

```
public void SetMassInertia(Matrix4x4 inertia, float mass, bool setAsInverse = false)
```

Parameters

inertia [Matrix4x4](#)

The inertia matrix to assign to the rigid body.

mass [float](#)

The mass of the rigid body.

setAsInverse bool ↗

Determines whether the provided inertia matrix should be treated as an inverse matrix.

SetMassInertia(float)

Sets the mass and automatically recalculates the inertia of the rigid body.

```
public void SetMassInertia(float mass)
```

Parameters

mass float ↗

The new mass value to be applied to the rigid body.

Class SoftBody3D

Namespace: [Sparkle.CSharp.Entities.Components](#)

Assembly: Sparkle.dll

```
public class SoftBody3D : Component, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Component](#) ← SoftBody3D

Implements

[IDisposable](#)

Inherited Members

[Component.GraphicsDevice](#) , [Component.Entity](#) , [Component.GlobalPosition](#) ,
[Component.Update\(double\)](#) , [Component.AfterUpdate\(double\)](#) , [Component.FixedUpdate\(double\)](#) ,
[Component.Resize\(Rectangle\)](#) , [Component.ConflictsWith<T>\(T\)](#) , [Disposable.Dispose\(\)](#) ,
[Disposable.ThrowIfDisposed\(\)](#) , [Disposable.HasDisposed](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

SoftBody3D(ISoftBodyFactory, bool, bool)

Initializes a new instance of the [SoftBody3D](#) component.

```
public SoftBody3D(ISoftBodyFactory factory, bool drawMesh = true, bool drawDebug = false)
```

Parameters

factory [ISoftBodyFactory](#)

The factory used to create the soft body instance.

drawMesh [bool](#)

Whether the soft body's mesh should be rendered. Defaults to [true](#).

drawDebug [bool](#)

Indicates whether debug visualization for the soft body should be enabled.

Fields

DebugDrawColor

The color used for debugging visualization of the 3D soft body's properties and behavior.

```
public Color DebugDrawColor
```

Field Value

Color

DrawDebug

Determines whether debug information, such as visual representations of physics-related elements, should be displayed for the soft body.

```
public bool DrawDebug
```

Field Value

[bool](#)

DrawMesh

Indicates whether the mesh associated with the soft body should be drawn.

```
public bool DrawMesh
```

Field Value

[bool](#)

Properties

BoneMatrics

A reference to the bone matrices for skeletal animation, if applicable.

```
public Matrix4x4[]? BoneMatrics { get; }
```

Property Value

[Matrix4x4](#)[]

Center

The center rigid body used to constrain the soft body structure.

```
public RigidBody Center { get; }
```

Property Value

RigidBody

InCompatibleTypes

Gets a list of component types that are incompatible with this component.

```
public override IReadOnlyList<Type> InCompatibleTypes { get; }
```

Property Value

[IReadOnlyList](#)<[Type](#)>

IsActive

Indicates whether the soft body is currently active in the simulation.

```
public bool IsActive { get; }
```

Property Value

[bool](#)

Material

A reference to the material used for rendering the mesh.

```
public ref Material Material { get; }
```

Property Value

Material

Mesh

The mesh used to visually represent the soft body.

```
public Mesh Mesh { get; }
```

Property Value

Mesh

OffsetPosition

The positional offset from the entity's origin. Always returns zero for this component.

```
public Vector3 OffsetPosition { get; }
```

Property Value

[Vector3](#)

Shapes

The geometric shapes representing the volume of the soft body.

```
public List<SoftBodyShape> Shapes { get; }
```

Property Value

[List](#)<SoftBodyShape>

Simulation

Gets the current 3D simulation instance used by the scene. Throws an [InvalidOperationException](#) if the simulation is not of type [Simulation3D](#).

```
public Simulation3D Simulation { get; }
```

Property Value

[Simulation3D](#)

SoftBody

The internal soft body implementation.

```
public SimpleSoftBody SoftBody { get; }
```

Property Value

[SimpleSoftBody](#)

Springs

The constraints (springs) connecting the soft body vertices.

```
public List<Constraint> Springs { get; }
```

Property Value

[List](#)<Constraint>

Vertices

The vertex rigid bodies that make up the soft body.

```
public List<RigidBody> Vertices { get; }
```

Property Value

[List](#)<RigidBody>

World

Gets the physics world from the current 3D simulation.

```
public World World { get; }
```

Property Value

World

Methods

DebugDraw(IDebugDrawer)

Renders debug information for the soft body component using the provided debug drawer.

```
public void DebugDraw(IDebugDrawer drawer)
```

Parameters

drawer IDebugDrawer

The debug drawer used to render the debug information.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

`disposing` [bool](#)

True if called from user code; false if called from a finalizer.

Draw(GraphicsContext, Framebuffer)

Renders the soft body mesh if it falls within the active camera's frustum, applying optional visual configurations.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

`context` [GraphicsContext](#)

The graphics context used for rendering operations.

`framebuffer` [Framebuffer](#)

The framebuffer to which the rendering output is written.

GetLerpedVertexPos(int)

Retrieves the interpolated position of a vertex at the specified index within the soft body.

```
public Vector3 GetLerpedVertexPos(int index)
```

Parameters

`index` [int](#)

The index of the vertex within the soft body.

Returns

[Vector3](#)

The interpolated position of the vertex. If interpolation data is unavailable, the current position of the vertex is returned.

Init()

Initializes the component and creates the soft body instance.

```
protected override void Init()
```

Class Sprite

Namespace: [Sparkle.CSharp.Entities.Components](#)

Assembly: Sparkle.dll

```
public class Sprite : InterpolatedComponent, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Component](#) ← [InterpolatedComponent](#) ← Sprite

Implements

[IDisposable](#)

Inherited Members

[InterpolatedComponent.LerpedPosition](#) , [InterpolatedComponent.LerpedGlobalPosition](#) ,
[InterpolatedComponent.LerpedRotation](#) , [InterpolatedComponent.LerpedScale](#) ,
[InterpolatedComponent.FixedUpdate\(double\)](#) , [Component.GraphicsDevice](#) , [Component.Entity](#) ,
[Component.InCompatibleTypes](#) , [Component.GlobalPosition](#) , [Component.OffsetPosition](#) ,
[Component.Init\(\)](#) , [Component.Update\(double\)](#) , [Component.AfterUpdate\(double\)](#) ,
[Component.Resize\(Rectangle\)](#) , [Component.ConflictsWith<T>\(T\)](#) , [Component.Dispose\(bool\)](#) ,
Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

Sprite(Texture2D, Vector2, Sampler?, Rectangle?, Vector2?, float,
Color?, SpriteFlip, Effect?, BlendStateDescription?,
DepthStencilStateDescription?, RasterizerStateDescription?,
Rectangle?)

Initializes a new instance of the [Sprite](#) class with configurable rendering and transform options.

```
public Sprite(Texture2D texture, Vector2 offsetPos, Sampler? sampler = null, Rectangle?  
sourceRect = null, Vector2? size = null, float layerDepth = 0.5, Color? color = null,  
SpriteFlip flip = SpriteFlip.None, Effect? effect = null, BlendStateDescription? blendState
```

```
= null, DepthStencilStateDescription? depthStencilState = null, RasterizerStateDescription?  
rasterizerState = null, Rectangle? scissorRect = null)
```

Parameters

texture Texture2D

The texture to display for this sprite.

offsetPos [Vector2](#)?

The local position offset of the sprite relative to its entity.

sampler [Sampler](#)?

Optional sampler used to sample the texture.

sourceRect Rectangle?

The subsection of the texture to draw. If null, uses the full texture.

size [Vector2](#)?

The size of the sprite. If null, defaults to the texture's dimensions.

layerDepth float?

The depth layer of the sprite (used for Z-ordering). Lower values are drawn in front.

color Color?

Optional color tint. Defaults to white if null.

flip SpriteFlip

The flip mode for rendering. Defaults to Bliss.CSharp.Graphics.Rendering.Renderers.Batches.Sprites.SpriteFlip.None.

effect Effect

Optional effect (shader) to apply to this sprite.

blendState [BlendStateDescription](#)?

Optional blend state override.

depthStencilState [DepthStencilStateDescription](#)?

Optional depth-stencil state override.

rasterizerState [RasterizerStateDescription](#)?

Optional rasterizer state override.

scissorRect Rectangle?

The rectangle used to define the scissor test area.

Fields

BlendState

Optional blend state override used when drawing the sprite.

```
public BlendStateDescription? BlendState
```

Field Value

[BlendStateDescription](#)?

Color

The tint color applied when rendering the sprite. Defaults to white (no tint).

```
public Color Color
```

Field Value

Color

DepthStencilState

Optional depth-stencil state override used when drawing the sprite.

```
public DepthStencilStateDescription? DepthStencilState
```

Field Value

[DepthStencilStateDescription](#)?

Effect

Optional custom shader effect to apply during rendering.

```
public Effect? Effect
```

Field Value

Effect

Flip

The flip mode of the sprite (e.g., horizontal, vertical, or none).

```
public SpriteFlip Flip
```

Field Value

SpriteFlip

LayerDepth

Specifies the rendering order of the sprite.

```
public float LayerDepth
```

Field Value

[float](#)

RasterizerState

Optional rasterizer state override used when drawing the sprite.

```
public RasterizerStateDescription? RasterizerState
```

Field Value

[RasterizerStateDescription](#)?

Sampler

Optional sampler state override used when sampling the sprite texture.

```
public Sampler? Sampler
```

Field Value

[Sampler](#)

ScissorRect

Optional scissor rectangle to define a clipping region during rendering.

```
public Rectangle? ScissorRect
```

Field Value

[Rectangle](#)?

Size

The size of the sprite in pixels. If not specified, defaults to the texture's dimensions.

```
public Vector2 Size
```

Field Value

[Vector2](#) ↗

SourceRect

The portion of the texture to be used when rendering the sprite.

`public Rectangle? SourceRect`

Field Value

`Rectangle?`

Texture

The texture used to render the sprite.

`public Texture2D Texture`

Field Value

`Texture2D`

Methods

Draw(GraphicsContext, Framebuffer)

Draws the sprite using the provided graphics context and target framebuffer.

`protected override void Draw(GraphicsContext context, Framebuffer framebuffer)`

Parameters

`context` [GraphicsContext](#)

The graphics context used for rendering operations.

framebuffer [Framebuffer](#)

The target framebuffer where the sprite will be drawn.

Namespace Sparkle.CSharp.GUI

Classes

[Gui](#)

[GuiManager](#)

Structs

[BorderInsets](#)

Enums

[Anchor](#)

[ResizeMode](#)

[TextAlignment](#)

Enum Anchor

Namespace: [Sparkle.CSharp.GUI](#)

Assembly: Sparkle.dll

```
public enum Anchor
```

Fields

BottomCenter = 7

Anchors the element to the bottom-center edge.

BottomLeft = 6

Anchors the element to the bottom-left corner.

BottomRight = 8

Anchors the element to the bottom-right corner.

Center = 4

Anchors the element to the center of its bounds.

CenterLeft = 3

Anchors the element to the center-left edge.

CenterRight = 5

Anchors the element to the center-right edge.

TopCenter = 1

Anchors the element to the top-center edge.

TopLeft = 0

Anchors the element to the top-left corner.

TopRight = 2

Anchors the element to the top-right corner.

Struct BorderInsets

Namespace: [Sparkle.CSharp.GUI](#)

Assembly: Sparkle.dll

```
public struct BorderInsets
```

Inherited Members

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

Constructors

BorderInsets(int)

Initializes a new instance of the [BorderInsets](#) struct.

```
public BorderInsets(int uniform)
```

Parameters

uniform [int](#)

The inset value applied equally to all sides.

BorderInsets(int, int, int, int)

Initializes a new instance of the [BorderInsets](#) struct.

```
public BorderInsets(int left, int right, int top, int bottom)
```

Parameters

left [int](#)

The inset value for the left side.

right [int↗](#)

The inset value for the right side.

top [int↗](#)

The inset value for the top side.

bottom [int↗](#)

The inset value for the bottom side.

Fields

Bottom

The size of the bottom border inset.

```
public int Bottom
```

Field Value

[int↗](#)

Left

The size of the left border inset.

```
public int Left
```

Field Value

[int↗](#)

Right

The size of the right border inset.

```
public int Right
```

Field Value

[int↗](#)

Top

The size of the top border inset.

```
public int Top
```

Field Value

[int↗](#)

Zero

Represents a border with all sides set to zero.

```
public static readonly BorderInsets Zero
```

Field Value

[BorderInsets](#)

Class Gui

Namespace: [Sparkle.CSharp.GUI](#)

Assembly: Sparkle.dll

```
public abstract class Gui : Disposable, IDisposable
```

Inheritance

[object](#) ← Disposable ← Gui

Implements

[IDisposable](#)

Derived

[TestGui](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

Gui(string, (int, int)?)

Initializes a new instance of the [Gui](#) class.

```
public Gui(string name, (int, int)? size = null)
```

Parameters

name [string](#)

The name of the GUI.

size ([int](#), [int](#))?

The width and height of the GUI.

Fields

Name

The name of the GUI.

```
public readonly string Name
```

Field Value

[string](#)

Size

The dimensions of the GUI.

```
public readonly (int Width, int Height) Size
```

Field Value

[\(int](#) [Width](#), [int](#) [Height](#))

Properties

ScaleFactor

The scaling factor applied to the GUI.

```
public int ScaleFactor { get; }
```

Property Value

[int](#)

Methods

AddElement(string, GuiElement)

Adds a new GUI element to the current GUI instance.

```
public void AddElement(string name, GuiElement element)
```

Parameters

name [string](#)

The unique name of the GUI element being added.

element [GuiElement](#)

The instance of the [GuiElement](#) to add to the GUI.

Exceptions

[Exception](#)

Thrown if the element with the specified name is already present in the GUI or has been added to another GUI.

AfterUpdate(double)

Performs post-update logic on all enabled elements.

```
protected virtual void AfterUpdate(double delta)
```

Parameters

delta [double](#)

Elapsed time since the last frame in seconds.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Draw(GraphicsContext, Framebuffer)

Draws all enabled elements to the specified framebuffer using the provided graphics context.

```
protected virtual void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The rendering context.

framebuffer [Framebuffer](#)

The framebuffer to draw into.

FixedUpdate(double)

Performs fixed-step updates on all enabled elements.

```
protected virtual void FixedUpdate(double fixedStep)
```

Parameters

fixedStep [double](#)

The fixed timestep interval.

GetElement(string)

Retrieves a GUI element by its name.

```
public GuiElement? GetElement(string name)
```

Parameters

`name` [string](#)

The name of the GUI element to retrieve.

Returns

[GuiElement](#)

The [GuiElement](#) associated with the specified name if found; otherwise, null.

GetElements()

Retrieves all GUI elements associated with the current GUI instance.

```
public IEnumerable<GuiElement> GetElements()
```

Returns

[IEnumerable](#)<[GuiElement](#)>

An enumerable collection of [GuiElement](#) objects belonging to this GUI.

HasElement(string)

Determines whether an element with the specified name exists in the GUI.

```
public bool HasElement(string name)
```

Parameters

`name` [string](#)

The name of the element to search for.

Returns

[bool](#)

True if an element with the specified name exists; otherwise, false.

Init()

Called when the GUI is initialized.

```
protected virtual void Init()
```

RemoveElement(GuiElement)

Removes the specified GUI element from the collection.

```
public void RemoveElement(GuiElement element)
```

Parameters

[element](#) [GuiElement](#)

The GUI element to be removed.

Exceptions

[Exception](#)

Thrown when the element could not be removed or disposed.

RemoveElement(string)

Removes the GUI element with the specified name. Throws an exception if the element cannot be removed.

```
public void RemoveElement(string name)
```

Parameters

name [string](#)

The name of the GUI element to remove.

Exceptions

[Exception](#)

Thrown when the element cannot be removed or disposed of successfully.

Resize(Rectangle)

Resizes all elements in the GUI based on the new window or framebuffer size.

```
protected virtual void Resize(Rectangle rectangle)
```

Parameters

rectangle Rectangle

The new size rectangle.

TryAddElement(string, GuiElement)

Attempts to add a new GUI element to the current GUI.

```
public bool TryAddElement(string name, GuiElement element)
```

Parameters

name [string](#)

The name of the GUI element to add. Must be unique and non-empty.

element [GuiElement](#)

The GUI element to add to the current GUI.

Returns

[bool](#) ↗

true if the element is successfully added; **false** if the name is empty, if an element with the same name already exists, or if the element is already associated with another GUI.

TryGetElement(string, out GuiElement?)

Attempts to retrieve a GUI element by its name.

```
public bool TryGetElement(string name, out GuiElement? element)
```

Parameters

[name](#) [string](#) ↗

The name of the GUI element to retrieve.

[element](#) [GuiElement](#)

When this method returns, contains the GUI element associated with the specified name, if found; otherwise, null.

Returns

[bool](#) ↗

True if the element with the specified name is found; otherwise, false.

TryRemoveElement(GuiElement)

Attempts to remove the specified GUI element from the collection of elements.

```
public bool TryRemoveElement(GuiElement element)
```

Parameters

element [GuiElement](#)

The [GuiElement](#) instance to be removed.

Returns

[bool](#) ↗

true if the element was successfully removed; otherwise, **false**.

TryRemoveElement(string)

Attempts to remove a GUI element by its name.

```
public bool TryRemoveElement(string name)
```

Parameters

name [string](#) ↗

The name of the GUI element to remove.

Returns

[bool](#) ↗

true if the element is successfully removed; otherwise, **false**.

Update(double)

Updates all enabled elements in the GUI.

```
protected virtual void Update(double delta)
```

Parameters

delta [double](#) ↗

Elapsed time since the last frame in seconds.

Class GuiManager

Namespace: [Sparkle.CSharp.GUI](#)

Assembly: Sparkle.dll

```
public static class GuiManager
```

Inheritance

[object](#) ← GuiManager

Inherited Members

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

Fields

Scale

A scaling factor applied to the GUI.

```
public static float Scale
```

Field Value

[float](#)

Properties

ActiveGui

The currently active GUI.

```
public static Gui? ActiveGui { get; }
```

Property Value

Methods

SetGui(Gui?)

Sets the currently active GUI. Disposes of the previous one, if set, and initializes the new GUI.

```
public static void SetGui(Gui? gui)
```

Parameters

gui [Gui](#)

The new GUI to set as active, or null to unset the active GUI.

Enum ResizeMode

Namespace: [Sparkle.CSharp.GUI](#)

Assembly: Sparkle.dll

```
public enum ResizeMode
```

Fields

NineSlice = 1

Uses nine-slice scaling where corners stay fixed and the center is stretched.

None = 0

Renders the texture at its original size without any scaling.

TileCenter = 2

Uses nine-slice scaling where corners stay fixed and the center is tiled.

Enum TextAlignement

Namespace: [Sparkle.CSharp.GUI](#)

Assembly: Sparkle.dll

```
public enum TextAlignement
```

Fields

Center = 1

Aligns text to the center.

Left = 0

Aligns text to the left.

Right = 2

Aligns text to the right.

Namespace Sparkle.CSharp.GUI.Elements

Classes

[GuiElement](#)

[LabelElement](#)

[RectangleButtonElement](#)

[RectangleSlideBarElement](#)

[RectangleTextBoxElement](#)

[TextureButtonElement](#)

[TextureDropDownElement](#)

[TextureSlideBarElement](#)

[TextureTextBoxElement](#)

[ToggleElement](#)

Class GuiElement

Namespace: [Sparkle.CSharp.GUI.Elements](#)

Assembly: Sparkle.dll

```
public abstract class GuiElement
```

Inheritance

[object](#) ← GuiElement

Derived

[LabelElement](#), [RectangleButtonElement](#), [RectangleSlideBarElement](#), [RectangleTextBoxElement](#),
[TextureButtonElement](#), [TextureDropDownElement](#), [TextureSlideBarElement](#), [TextureTextBoxElement](#),
[ToggleElement](#)

Inherited Members

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

Constructors

GuiElement(Anchor, Vector2, Vector2, Vector2?, Vector2?, float, Func<GuiElement, bool>?)

Initializes a new instance of the [GuiElement](#) class.

```
public GuiElement(Anchor anchor, Vector2 offset, Vector2 size, Vector2? scale = null,  
Vector2? origin = null, float rotation = 0, Func<GuiElement, bool>? clickFunc = null)
```

Parameters

anchor [Anchor](#)

The anchor point for positioning the element.

offset [Vector2](#)

The offset from the anchor position.

size [Vector2](#)

The unscaled size of the element.

scale [Vector2](#)?

Optional scaling factor for the element. Defaults to (1, 1).

origin [Vector2](#)?

Optional origin point for rotation/scaling. Defaults to (0, 0).

rotation [float](#)

Optional rotation in radians. Defaults to 0.

clickFunc [Func](#)<[GuiElement](#), [bool](#)>

Optional function that determines what happens on click. Should return true if handled.

Fields

AnchorPoint

The anchor point used for positioning this element relative to the screen.

public [Anchor](#) [AnchorPoint](#)

Field Value

[Anchor](#)

Enabled

Indicates whether this element is active and should be updated and drawn.

public [bool](#) [Enabled](#)

Field Value

[bool](#)

Interactable

Indicates whether the GUI element can respond to user interactions, such as clicks or hover events.

`public bool Interactable`

Field Value

[bool](#)

Offset

The offset from the anchor point used to position this element.

`public Vector2 Offset`

Field Value

[Vector2](#)

Origin

The origin point (pivot) used for rotation and scaling.

`public Vector2 Origin`

Field Value

[Vector2](#)

Rotation

The rotation applied to the element in radians.

```
public float Rotation
```

Field Value

[float](#)

Scale

The scaling factor applied to the element, modifying its size along the X and Y axes.

```
public Vector2 Scale
```

Field Value

[Vector2](#)

Size

The base size of the element before scaling.

```
public Vector2 Size
```

Field Value

[Vector2](#)

Properties

Gui

The GUI instance to which this element belongs.

```
public Gui Gui { get; }
```

Property Value

IsClicked

Indicates whether the element has been clicked during this update cycle.

```
public bool IsClicked { get; }
```

Property Value

[bool](#) ↗

IsHovered

Indicates whether the mouse is currently hovering over this element.

```
public bool IsHovered { get; }
```

Property Value

[bool](#) ↗

IsSelected

Indicates whether this element is currently selected.

```
public bool IsSelected { get; }
```

Property Value

[bool](#) ↗

Name

The unique name of this element within its GUI.

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

Property Value

[string](#)

Position

The on-screen position of the element after anchor and offset are applied.

```
public Vector2 Position { get; }
```

Property Value

[Vector2](#)

ScaledSize

The size of the element after scaling is applied.

```
public Vector2 ScaledSize { get; }
```

Property Value

[Vector2](#)

Methods

AfterUpdate(double)

Invoked after the update process to perform additional operations specific to the element.

```
protected virtual void AfterUpdate(double delta)
```

Parameters

`delta` [double](#)

The time elapsed since the last update, typically used for time-dependent operations.

CalculatePos()

Calculates the position of the GUI element based on its anchor point, scaled size, offset, and origin.

`protected virtual Vector2 CalculatePos()`

Returns

[Vector2](#)

A [Vector2](#) representing the calculated position of the element on the screen.

CalculateSize()

Calculates the scaled size of the GUI element based on the base size and the current scale factor.

`protected virtual Vector2 CalculateSize()`

Returns

[Vector2](#)

The scaled size of the GUI element as a [Vector2](#).

Draw(GraphicsContext, Framebuffer)

Renders the graphical representation of the GUI element to the specified framebuffer.

`protected abstract void Draw(GraphicsContext context, Framebuffer framebuffer)`

Parameters

`context` [GraphicsContext](#)

The graphics context used for rendering the element.

framebuffer [Framebuffer](#)

The framebuffer where the element will be drawn.

FixedUpdate(double)

Executes logic that requires a constant fixed time step for this element.

```
protected virtual void FixedUpdate(double fixedStep)
```

Parameters

fixedStep [double](#)

The fixed time step duration, typically used for physics or other time-dependent updates.

Resize(Rectangle)

Optional method for responding to window or viewport resize events.

```
protected virtual void Resize(Rectangle rectangle)
```

Parameters

rectangle Rectangle

The new size and bounds of the rendering area.

Update(double, ref bool)

Updates the state of the GuiElement during each frame with the given time delta.

```
protected virtual void Update(double delta, ref bool interactionHandled)
```

Parameters

delta double ↗

The time elapsed between the current and the previous frame, in seconds.

interactionHandled bool ↗

A reference to a boolean tracking whether interaction has already been handled by another element.

Class LabelElement

Namespace: [Sparkle.CSharp.GUI.Elements](#)

Assembly: Sparkle.dll

```
public class LabelElement : GuiElement
```

Inheritance

[object](#) ← [GuiElement](#) ← LabelElement

Inherited Members

[GuiElement.Gui](#) , [GuiElement.Name](#) , [GuiElement.Enabled](#) , [GuiElement.Interactable](#) ,
[GuiElement.AnchorPoint](#) , [GuiElement.Offset](#) , [GuiElement.Size](#) , [GuiElement.Scale](#) , [GuiElement.Origin](#) ,
[GuiElement.Rotation](#) , [GuiElement.IsHovered](#) , [GuiElement.IsClicked](#) , [GuiElement.IsSelected](#) ,
[GuiElement.Position](#) , [GuiElement.ScaledSize](#) , [GuiElement.AfterUpdate\(double\)](#) ,
[GuiElement.FixedUpdate\(double\)](#) , [GuiElement.Resize\(Rectangle\)](#) , [GuiElement.CalculatePos\(\)](#) ,
[GuiElement.CalculateSize\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

LabelElement(LabelData, Anchor, Vector2, Vector2?, Vector2?, float, Func<GuiElement, bool>?)

Initializes a new instance of the [LabelElement](#) class.

```
public LabelElement(LabelData data, Anchor anchor, Vector2 offset, Vector2? scale = null,  
Vector2? origin = null, float rotation = 0, Func<GuiElement, bool>? clickFunc = null)
```

Parameters

data [LabelData](#)

The label data containing text and rendering settings.

anchor [Anchor](#)

The anchor point that determines the label's alignment.

offset [Vector2](#)?

The offset from the anchor point.

scale [Vector2](#)?

Optional scaling factor for the element. Defaults to (1, 1).

origin [Vector2](#)?

Optional origin point used for rotation and scaling (default is null).

rotation [float](#)

The rotation of the label in radians (default is 0).

clickFunc [Func](#)<[GuiElement](#), [bool](#)>

Optional function to be called when the label is clicked (default is null).

Properties

Data

The data used to render the label.

```
public LabelData Data { get; }
```

Property Value

[LabelData](#)

Methods

Draw(GraphicsContext, Framebuffer)

Draws the LabelElement on the specified framebuffer using the provided graphics context.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering the LabelElement.

framebuffer [Framebuffer](#)

The framebuffer in which the LabelElement will be drawn.

Update(double, ref bool)

Updates the state of the LabelElement during each frame with the given time delta.

```
protected override void Update(double delta, ref bool interactionHandled)
```

Parameters

delta [double](#)

The time elapsed between the current and the previous frame, in seconds.

interactionHandled [bool](#)

A reference to a boolean tracking whether interaction has already been handled by another element.

Class RectangleButtonElement

Namespace: [Sparkle.CSharp.GUI.Elements](#)

Assembly: Sparkle.dll

```
public class RectangleButtonElement : GuiElement
```

Inheritance

[object](#) ← [GuiElement](#) ← RectangleButtonElement

Inherited Members

[GuiElement.Gui](#) , [GuiElement.Name](#) , [GuiElement.Enabled](#) , [GuiElement.Interactable](#) ,
[GuiElement.AnchorPoint](#) , [GuiElement.Offset](#) , [GuiElement.Size](#) , [GuiElement.Scale](#) , [GuiElement.Origin](#) ,
[GuiElement.Rotation](#) , [GuiElement.IsHovered](#) , [GuiElement.IsClicked](#) , [GuiElement.IsSelected](#) ,
[GuiElement.Position](#) , [GuiElement.ScaledSize](#) , [GuiElement.Update\(double, ref bool\)](#) ,
[GuiElement.AfterUpdate\(double\)](#) , [GuiElement.FixedUpdate\(double\)](#) , [GuiElement.Resize\(Rectangle\)](#) ,
[GuiElement.CalculatePos\(\)](#) , [GuiElement.CalculateSize\(\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

RectangleButtonElement(RectangleButtonData, LabelData, Anchor, Vector2, Vector2, Vector2?, TextAlign, Vector2?, Vector2?, float, Func<GuiElement, bool>?)

Initializes a new instance of the [RectangleButtonElement](#) class.

```
public RectangleButtonElement(RectangleButtonData buttonData, LabelData labelData, Anchor  
anchor, Vector2 offset, Vector2 size, Vector2? scale = null, TextAlign alignment =  
TextAlign.Center, Vector2? textOffset = null, Vector2? origin = null, float rotation =  
0, Func<GuiElement, bool>? clickFunc = null)
```

Parameters

buttonData [RectangleButtonData](#)

The visual configuration of the rectangle button.

labelData [LabelData](#)

The text label data displayed on the button.

anchor [Anchor](#)

The anchor point for positioning the element.

offset [Vector2](#)

The offset from the anchor position.

size [Vector2](#)

The size of the button.

scale [Vector2](#)?

The scale applied to the button.

textAlignment [TextAlignment](#)

The alignment of text within a GUI element.

textOffset [Vector2](#)?

The offset of the text relative to its position.

origin [Vector2](#)?

The origin point for rotation and alignment. Defaults to (0, 0).

rotation [float](#)

The rotation of the button in radians. Defaults to 0.

clickFunc [Func](#)<[GuiElement](#), [bool](#)>

Optional function to invoke when the button is clicked. Returns true if handled.

Fields

TextAlignment

The alignment of text within a GUI element.

```
public TextAlignement TextAlignement
```

Field Value

[TextAlignment](#)

TextOffset

The offset of the text relative to its position.

```
public Vector2 TextOffset
```

Field Value

[Vector2](#)

Properties

ButtonData

Gets the button's visual configuration, including fill color, hover color, and outline properties.

```
public RectangleButtonData ButtonData { get; }
```

Property Value

[RectangleButtonData](#)

LabelData

Gets the label data used to render text over the button.

```
public LabelData LabelData { get; }
```

Property Value

Methods

Draw(GraphicsContext, Framebuffer)

Renders the rectangle button element on the given framebuffer using the provided graphics context.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering primitives and text.

framebuffer [Framebuffer](#)

The framebuffer target for rendering.

Class RectangleSlideBarElement

Namespace: [Sparkle.CSharp.GUI.Elements](#)

Assembly: Sparkle.dll

```
public class RectangleSlideBarElement : GuiElement
```

Inheritance

[object](#) ← [GuiElement](#) ← RectangleSlideBarElement

Inherited Members

[GuiElement.Gui](#) , [GuiElement.Name](#) , [GuiElement.Enabled](#) , [GuiElement.Interactable](#) ,
[GuiElement.AnchorPoint](#) , [GuiElement.Offset](#) , [GuiElement.Size](#) , [GuiElement.Scale](#) , [GuiElement.Origin](#) ,
[GuiElement.Rotation](#) , [GuiElement.IsHovered](#) , [GuiElement.IsClicked](#) , [GuiElement.IsSelected](#) ,
[GuiElement.Position](#) , [GuiElement.ScaledSize](#) , [GuiElement.AfterUpdate\(double\)](#) ,
[GuiElement.FixedUpdate\(double\)](#) , [GuiElement.Resize\(Rectangle\)](#) , [GuiElement.CalculatePos\(\)](#) ,
[GuiElement.CalculateSize\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

RectangleSlideBarElement(RectangleSlideBarData, Anchor, Vector2, Vector2, float, float, float, bool, Vector2?, Vector2?, float, Func<GuiElement, bool>?)

Initializes a new instance of the [RectangleSlideBarElement](#) class.

```
public RectangleSlideBarElement(RectangleSlideBarData data, Anchor anchor, Vector2 offset,  
Vector2 size, float minValue, float maxValue, float value = 0, bool wholeNumbers = false,  
Vector2? scale = null, Vector2? origin = null, float rotation = 0, Func<GuiElement, bool>  
clickFunc = null)
```

Parameters

data [RectangleSlideBarData](#)

The visual and styling data used to render the slide bar.

anchor [Anchor](#)

The anchor point used to position the element.

offset [Vector2](#)

The offset from the anchor position.

size [Vector2](#)

The size of the slide bar.

minValue [float](#)

The minimum value of the slide bar.

maxValue [float](#)

The maximum value of the slide bar.

value [float](#)

The initial value of the slide bar.

wholeNumbers [bool](#)

Indicates whether the value should be restricted to whole numbers.

scale [Vector2](#)?

Optional scale applied to the element.

origin [Vector2](#)?

Optional origin point used for rotation and alignment.

rotation [float](#)

The rotation of the element in degrees.

clickFunc [Func](#)<[GuiElement](#), [bool](#)>

Optional function invoked when the element is clicked.

Fields

MaxValue

The maximum allowable value for the slider bar, defining the upper limit of the slider's range.

```
public float MaxValue
```

Field Value

[float](#)

MinValue

The minimum allowable value for the slider bar, defining the lower limit of the slider's range.

```
public float MinValue
```

Field Value

[float](#)

WholeNumbers

The slider value should be rounded to the nearest whole number.

```
public bool WholeNumbers
```

Field Value

[bool](#)

Properties

Data

The data used to render the slide bar.

```
public RectangleSlideBarData Data { get; }
```

Property Value

[RectangleSlideBarData](#)

Value

The current value of the slider bar, clamped between the minimum and maximum values.

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

Property Value

[float](#)

Methods

Draw(GraphicsContext, Framebuffer)

Renders the rectangle slide bar element onto the specified framebuffer using the provided graphics context.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering operations, including primitive drawing.

framebuffer [Framebuffer](#)

The framebuffer target where the slide bar will be rendered.

Update(double, ref bool)

Updates the visual state and properties of the rectangle slide bar element based on time delta and user interactions.

```
protected override void Update(double delta, ref bool interactionHandled)
```

Parameters

delta [double](#)

The elapsed time, in seconds, since the last update, used for time-based calculations or animations.

interactionHandled [bool](#)

A reference to a boolean tracking whether interaction has already been handled by another element.

Events

ValueUpdated

An event that triggers when the slider value is updated. The new value of the slider is passed as a parameter to the attached event handlers.

```
public event Action<float>? ValueUpdated
```

Event Type

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

Class RectangleTextBoxElement

Namespace: [Sparkle.CSharp.GUI.Elements](#)

Assembly: Sparkle.dll

```
public class RectangleTextBoxElement : GuiElement
```

Inheritance

[object](#) ← [GuiElement](#) ← RectangleTextBoxElement

Inherited Members

[GuiElement.Gui](#) , [GuiElement.Name](#) , [GuiElement.Enabled](#) , [GuiElement.Interactable](#) ,
[GuiElement.AnchorPoint](#) , [GuiElement.Offset](#) , [GuiElement.Size](#) , [GuiElement.Scale](#) , [GuiElement.Origin](#) ,
[GuiElement.Rotation](#) , [GuiElement.IsHovered](#) , [GuiElement.IsClicked](#) , [GuiElement.IsSelected](#) ,
[GuiElement.Position](#) , [GuiElement.ScaledSize](#) , [GuiElement.AfterUpdate\(double\)](#) ,
[GuiElement.FixedUpdate\(double\)](#) , [GuiElement.Resize\(Rectangle\)](#) , [GuiElement.CalculatePos\(\)](#) ,
[GuiElement.CalculateSize\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

RectangleTextBoxElement(RectangleTextBoxData, LabelData, LabelData, Anchor, Vector2, Vector2, int, Vector2?,
TextAlignment, Vector2?, (float Left, float Right)?, Vector2?, float, Func<GuiElement, bool>?)

Initializes a new instance of the [RectangleButtonElement](#) class.

```
public RectangleTextBoxElement(RectangleTextBoxData textBoxData, LabelData labelData,  
LabelData hintLabelData, Anchor anchor, Vector2 offset, Vector2 size, int maxTextLength,  
Vector2? scale = null, TextAlignment textAlignment = TextAlignment.Left, Vector2? textOffset  
= null, (float Left, float Right)? textEdgeOffset = null, Vector2? origin = null, float  
rotation = 0, Func<GuiElement, boolnull)
```

Parameters

textBoxData [RectangleTextBoxData](#)

Configuration for the rectangle and visual appearance of the textbox.

labelData [LabelData](#)

Configuration for the primary text label.

hintLabelData [LabelData](#)

Configuration for the hint text label.

anchor [Anchor](#)

Defines the anchor point for positioning the element.

offset [Vector2](#)?

Position offset relative to the anchor.

size [Vector2](#)?

Size of the textbox.

maxTextLength [int](#)?

Maximum number of allowed characters in the textbox.

scale [Vector2](#)?

Optional parameter that defines the scaling factor for the element. Default is null.

textAlignment [TextAlignment](#)

Text alignment within the textbox. Default is Left.

textOffset [Vector2](#)?

The offset of the text relative to its position.

textEdgeOffset ([float](#) [Left](#), [float](#) [Right](#))?

Optional offsets for the left and right edges of the text. Default is (0.0F, 0.0F).

origin [Vector2](#)?

Optional origin point for rotation and scaling. Default is the center.

`rotation` [float](#)

Optional rotation angle (in radians). Default is 0.

`clickFunc` [Func](#)<[GuiElement](#), [bool](#)>

Optional custom function to execute on click. Returns true if the event is consumed.

Fields

TextAlignment

Specifies the alignment of text within the textbox (e.g., Left, Center, Right).

```
public TextAlignment TextAlignment
```

Field Value

[TextAlignment](#)

TextEdgeOffset

The horizontal offsets to be applied to the text edges within the textbox.

```
public (float Left, float Right) TextEdgeOffset
```

Field Value

([float](#) [angular](#), [float](#) [linear](#))

TextOffset

The offset of the text relative to its position.

```
public Vector2 TextOffset
```

Field Value

Properties

HintLabelData

Holds the configuration for the hint label text, shown when the textbox is empty.

```
public LabelData HintLabelData { get; }
```

Property Value

[LabelData](#)

LabelData

Holds the configuration for the label text displayed within the textbox.

```
public LabelData LabelData { get; }
```

Property Value

[LabelData](#)

MaxTextLength

The maximum number of characters allowed within the textbox.

```
public int MaxTextLength { get; }
```

Property Value

[int](#)

TextBoxData

Holds the configuration for the rectangle-based textbox element.

```
public RectangleTextBoxData TextBoxData { get; }
```

Property Value

[RectangleTextBoxData](#)

Methods

Draw(GraphicsContext, Framebuffer)

Draws the GUI element on the specified framebuffer using the provided graphics context.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering.

framebuffer [Framebuffer](#)

The framebuffer to which the element is rendered.

Update(double, ref bool)

Updates the state of the [RectangleButtonElement](#).

```
protected override void Update(double delta, ref bool interactionHandled)
```

Parameters

delta [double](#)

Elapsed time in seconds since the last update.

interactionHandled [bool](#)

A reference to a boolean tracking whether interaction has already been handled by another element.

Class TextureButtonElement

Namespace: [Sparkle.CSharp.GUI.Elements](#)

Assembly: Sparkle.dll

```
public class TextureButtonElement : GuiElement
```

Inheritance

[object](#) ← [GuiElement](#) ← TextureButtonElement

Inherited Members

[GuiElement.Gui](#) , [GuiElement.Name](#) , [GuiElement.Enabled](#) , [GuiElement.Interactable](#) ,
[GuiElement.AnchorPoint](#) , [GuiElement.Offset](#) , [GuiElement.Size](#) , [GuiElement.Scale](#) , [GuiElement.Origin](#) ,
[GuiElement.Rotation](#) , [GuiElement.IsHovered](#) , [GuiElement.IsClicked](#) , [GuiElement.IsSelected](#) ,
[GuiElement.Position](#) , [GuiElement.ScaledSize](#) , [GuiElement.Update\(double, ref bool\)](#) ,
[GuiElement.AfterUpdate\(double\)](#) , [GuiElement.FixedUpdate\(double\)](#) , [GuiElement.Resize\(Rectangle\)](#) ,
[GuiElement.CalculatePos\(\)](#) , [GuiElement.CalculateSize\(\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

TextureButtonElement(TextureButtonData, LabelData, Anchor, Vector2, TextAlignment, Vector2?, Vector2?, Vector2?, Vector2?, float, Func<GuiElement, bool>?)

Initializes a new instance of the [TextureButtonElement](#) class.

```
public TextureButtonElement(TextureButtonData buttonData, LabelData labelData, Anchor  
anchor, Vector2 offset, TextAlignment textAlignment = TextAlignment.Center, Vector2?  
textOffset = null, Vector2? size = null, Vector2? scale = null, Vector2? origin = null,  
float rotation = 0, Func<GuiElement, bool>? clickFunc = null)
```

Parameters

buttonData [TextureButtonData](#)

The texture and visual configuration for the button.

labelData [LabelData](#)

The label configuration to be drawn over the button.

anchor [Anchor](#)

The anchor point determining the element's relative position.

offset [Vector2](#)?

The offset from the anchor point.

textAlignment [TextAlignment](#)

The alignment of text within a GUI element.

textOffset [Vector2](#)?

The offset of the text relative to its position.

size [Vector2](#)?

Optional override for the size. If not provided, defaults to the texture size.

scale [Vector2](#)?

The scale applied to the button.

origin [Vector2](#)?

Optional origin point for transformations like rotation and scaling.

rotation [float](#)

Optional rotation angle in radians.

clickFunc [Func](#)<[GuiElement](#), [bool](#)>

Optional function to execute when the button is clicked. Should return true if handled.

Fields

TextAlignment

The alignment of text within a GUI element.

```
public TextAlignment TextAlignment
```

Field Value

[TextAlignment](#)

TextOffset

The offset of the text relative to its position.

```
public Vector2 TextOffset
```

Field Value

[Vector2](#)

Properties

ButtonData

The associated data for a texture-based button in the GUI.

```
public TextureButtonData ButtonData { get; }
```

Property Value

[TextureButtonData](#)

LabelData

The data and properties necessary for rendering and handling text on a GUI element.

```
public LabelData LabelData { get; }
```

Property Value

Methods

Draw(GraphicsContext, Framebuffer)

Draws the texture button element and its label onto the specified framebuffer.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering operations.

framebuffer [Framebuffer](#)

The framebuffer where the element will be drawn.

Class TextureDropDownElement

Namespace: [Sparkle.CSharp.GUI.Elements](#)

Assembly: Sparkle.dll

```
public class TextureDropDownElement : GuiElement
```

Inheritance

[object](#) ← [GuiElement](#) ← TextureDropDownElement

Inherited Members

[GuiElement.Gui](#), [GuiElement.Name](#), [GuiElement.Enabled](#), [GuiElement.Interactable](#),
[GuiElement.AnchorPoint](#), [GuiElement.Offset](#), [GuiElement.Size](#), [GuiElement.Scale](#), [GuiElement.Origin](#),
[GuiElement.Rotation](#), [GuiElement.IsHovered](#), [GuiElement.IsClicked](#), [GuiElement.IsSelected](#),
[GuiElement.Position](#), [GuiElement.ScaledSize](#), [GuiElement.AfterUpdate\(double\)](#),
[GuiElement.FixedUpdate\(double\)](#), [GuiElement.Resize\(Rectangle\)](#), [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Constructors

TextureDropDownElement(TextureDropDownData, List<LabelData>, int, Anchor, Vector2, TextAlignment, TextAlignment, Vector2?, Vector2?, Vector2?, Vector2?, Vector2?, (float Top, float Bottom)?, float, float, Vector2?, Vector2?, Vector2?, Vector2?, float, Func<GuiElement, bool>?)

Initializes a new instance of the [TextureDropDownElement](#) class.

```
public TextureDropDownElement(TextureDropDownData dropDownData, List<LabelData> options, int maxVisibleOptions, Anchor anchor, Vector2 offset, TextAlignment fieldTextAlignment = TextAlignment.Left, TextAlignment menuTextAlignment = TextAlignment.Left, Vector2? fieldTextOffset = null, Vector2? menuTextOffset = null, Vector2? sliderOffset = null, Vector2? arrowOffset = null, (float Top, float Bottom)? scrollMaskInsets = null, float scrollSensitivity = 0.1, float scrollLerpSpeed = 10, Vector2? size = null, Vector2? scale = null, Vector2? origin = null, float rotation = 0, Func<GuiElement, bool>? clickFunc = null)
```

Parameters

dropDownData [TextureDropDownData](#)

The visual and texture data used to render the dropdown field and menu.

options [List](#)<[LabelData](#)>

The list of label data entries representing selectable options.

maxVisibleOptions [int](#)

The maximum number of options visible in the dropdown menu at a time.

anchor [Anchor](#)

The anchor point used to position the dropdown element.

offset [Vector2](#)

The offset from the anchor position.

fieldTextAlignment [TextAlignment](#)

The text alignment used for the selected value displayed in the field.

menuTextAlignment [TextAlignment](#)

The text alignment used for options displayed in the dropdown menu.

fieldTextOffset [Vector2](#)?

The offset applied to the field text.

menuTextOffset [Vector2](#)?

The offset applied to the menu option text.

sliderOffset [Vector2](#)?

An optional offset applied to the position of the slider within the dropdown menu.

arrowOffset [Vector2](#)?

The offset applied to the dropdown arrow indicator.

scrollMaskInsets ([float](#) [Left](#), [float](#) [Right](#))?

Defines top and bottom padding for the scroll text clipping mask.

scrollSensitivity [float](#)

Indicates how sensitive the dropdown menu scrolling is to user input.

scrollLerpSpeed [float](#)

Specifies the speed at which the dropdown menu's scroll position interpolates to the target position.

size [Vector2](#)?

The size of the dropdown element.

scale [Vector2](#)?

Optional scale applied to the dropdown element.

origin [Vector2](#)?

Optional origin point used for rotation and alignment.

rotation [float](#)

The rotation of the dropdown element in radians.

clickFunc [Func](#)<[GuiElement](#), [bool](#)>

Optional function invoked when the dropdown field is clicked.

Fields

ArrowOffset

The offset applied to the dropdown arrow indicator.

```
public Vector2 ArrowOffset
```

Field Value

[Vector2](#)

FieldTextAlignment

The text alignment used for the selected value displayed in the dropdown field.

```
public TextAlignement FieldTextAlignment
```

Field Value

[TextAlignment](#)

FieldTextOffset

The offset applied to the text displayed in the dropdown field.

```
public Vector2 FieldTextOffset
```

Field Value

[Vector2](#)

MenuTextAlignment

The text alignment used for options displayed in the dropdown menu.

```
public TextAlignement MenuTextAlignment
```

Field Value

[TextAlignment](#)

MenuTextOffset

The offset applied to the text displayed in the dropdown menu items.

```
public Vector2 MenuTextOffset
```

Field Value

[Vector2](#)

ScrollLerpSpeed

The interpolation speed at which the menu scrolls to a target position.

`public float ScrollLerpSpeed`

Field Value

[float](#)

ScrollMaskInsets

The insets applied to the scroll mask for controlling the visible region of the dropdown menu's scrollable content.

`public (float Top, float Bottom) ScrollMaskInsets`

Field Value

[\(float angular, float linear\)](#)

ScrollSensitivity

The sensitivity of the dropdown scroll when navigating through options.

`public float ScrollSensitivity`

Field Value

[float](#)

SliderOffset

The offset applied to the slider element used within the dropdown component.

```
public Vector2 SliderOffset
```

Field Value

[Vector2](#)

Properties

DropDownData

The visual and texture configuration used to render the dropdown field and menu.

```
public TextureDropDownData DropDownData { get; }
```

Property Value

[TextureDropDownData](#)

IsMenuOpen

The dropdown menu is currently open.

```
public bool IsMenuOpen { get; }
```

Property Value

[bool](#)

MaxVisibleOptions

The maximum number of dropdown menu options visible at once.

```
public int MaxVisibleOptions { get; }
```

Property Value

[int](#)

Options

The list of selectable options displayed in the dropdown menu.

```
public List<LabelData> Options { get; }
```

Property Value

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

SelectedOption

The currently selected option from the dropdown menu.

```
public LabelData? SelectedOption { get; }
```

Property Value

[LabelData](#)

Methods

CalculatePos()

Calculates the position of the texture drop-down element based on its anchor point, offset, scale, and other transformation factors. If the drop-down menu is closed, the position is determined using the base implementation.

```
protected override Vector2 CalculatePos()
```

Returns

[Vector2](#)

A [Vector2](#) representing the calculated position of the element on the screen.

CalculateSize()

Calculates the size of the texture drop-down element, taking into account whether the drop-down menu is open or closed.

```
protected override Vector2 CalculateSize()
```

Returns

[Vector2](#)

A [Vector2](#) representing the size of the element. If the menu is open, the size is adjusted to include the additional height for the drop-down options.

Draw(GraphicsContext, Framebuffer)

Renders the texture-based dropdown element, including its field, menu, arrow, and text components, based on the current state and interaction details.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering operations, including sprite batching.

framebuffer [Framebuffer](#)

The target framebuffer where the dropdown element will be drawn.

Update(double, ref bool)

Updates the state of the texture-based dropdown element, including handling user interactions such as clicking and selecting options, and managing the visibility of the dropdown menu.

```
protected override void Update(double delta, ref bool interactionHandled)
```

Parameters

delta [double](#)

The time elapsed since the last update, in seconds, used for timing-related logic.

interactionHandled [bool](#)

A reference to a boolean tracking whether interaction has already been handled by another element.

Events

MenuToggled

Event triggered when the dropdown menu is toggled open or closed.

```
public event Action<bool>? MenuToggled
```

Event Type

[Action](#) <[bool](#)>

OptionChanged

Triggered when the selected option in the dropdown menu changes.

```
public event Action<LabelData>? OptionChanged
```

Event Type

[Action](#) <[LabelData](#)>

Class TextureSlideBarElement

Namespace: [Sparkle.CSharp.GUI.Elements](#)

Assembly: Sparkle.dll

```
public class TextureSlideBarElement : GuiElement
```

Inheritance

[object](#) ← [GuiElement](#) ← TextureSlideBarElement

Inherited Members

[GuiElement.Gui](#) , [GuiElement.Name](#) , [GuiElement.Enabled](#) , [GuiElement.Interactable](#) ,
[GuiElement.AnchorPoint](#) , [GuiElement.Offset](#) , [GuiElement.Size](#) , [GuiElement.Scale](#) , [GuiElement.Origin](#) ,
[GuiElement.Rotation](#) , [GuiElement.IsHovered](#) , [GuiElement.IsClicked](#) , [GuiElement.IsSelected](#) ,
[GuiElement.Position](#) , [GuiElement.ScaledSize](#) , [GuiElement.AfterUpdate\(double\)](#) ,
[GuiElement.FixedUpdate\(double\)](#) , [GuiElement.Resize\(Rectangle\)](#) , [GuiElement.CalculatePos\(\)](#) ,
[GuiElement.CalculateSize\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

TextureSlideBarElement(TextureSlideBarData, Anchor, Vector2, float, float, float, bool, Vector2?, Vector2?, Vector2?, float, Func<GuiElement, bool>?)

Initializes a new instance of the [TextureSlideBarElement](#) class.

```
public TextureSlideBarElement(TextureSlideBarData data, Anchor anchor, Vector2 offset, float minValue, float maxValue, float value = 0, bool wholeNumbers = false, Vector2? size = null, Vector2? scale = null, Vector2? origin = null, float rotation = 0, Func<GuiElement, bool>? clickFunc = null)
```

Parameters

data [TextureSlideBarData](#)

Defines the texture and styling data for the slider bar.

anchor [Anchor](#)

Specifies the alignment of the slider bar to its parent container.

offset [Vector2](#)

Defines the positional offset of the slider bar in relation to its anchor.

minValue [float](#)

The minimum value the slider bar represents.

maxValue [float](#)

The maximum value the slider bar represents.

value [float](#)

The initial value of the slider bar, which can be changed by user interaction.

wholeNumbers [bool](#)

Indicates whether the slider should operate in whole number (integer) intervals or support floating-point values.

size [Vector2](#)?

Specifies the size of the slider bar. Defaults to the size defined by the texture data if not provided.

scale [Vector2](#)?

Specifies the scale factor for the slider bar's dimensions.

origin [Vector2](#)?

Defines the origin point for transformations applied to the slider bar.

rotation [float](#)

The rotation angle (in degrees) to apply to the slider bar.

clickFunc [Func](#)<[GuiElement](#), [bool](#)>

A function that is invoked when the slider bar is clicked, returning a boolean to indicate success or state change.

Fields

MaxValue

The maximum allowable value for the slider bar, defining the upper limit of the slider's range.

```
public float MaxValue
```

Field Value

[float](#) ↗

MinValue

The minimum allowable value for the slider bar, defining the lower limit of the slider's range.

```
public float MinValue
```

Field Value

[float](#) ↗

WholeNumbers

The slider value should be rounded to the nearest whole number.

```
public bool WholeNumbers
```

Field Value

[bool](#) ↗

Properties

Data

The data used to render the slide bar.

```
public TextureSlideBarData Data { get; }
```

Property Value

[TextureSlideBarData](#)

Value

The current value of the slider bar, clamped between the minimum and maximum values.

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

Property Value

[float](#) ↗

Methods

Draw(GraphicsContext, Framebuffer)

Draws the texture button element and its label onto the specified framebuffer.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering operations.

framebuffer [Framebuffer](#) ↗

The framebuffer where the element will be drawn.

Update(double, ref bool)

Updates the state of the texture slider bar element, including checking for interaction, handling dragging behavior, and updating the current value based on user input.

```
protected override void Update(double delta, ref bool interactionHandled)
```

Parameters

delta [double](#)

The time elapsed since the last update, in seconds.

interactionHandled [bool](#)

A reference to a boolean tracking whether interaction has already been handled by another element.

Events

ValueUpdated

An event that triggers when the slider value is updated. The new value of the slider is passed as a parameter to the attached event handlers.

```
public event Action<float>? ValueUpdated
```

Event Type

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

Class TextureTextBoxElement

Namespace: [Sparkle.CSharp.GUI.Elements](#)

Assembly: Sparkle.dll

```
public class TextureTextBoxElement : GuiElement
```

Inheritance

[object](#) ← [GuiElement](#) ← TextureTextBoxElement

Inherited Members

[GuiElement.Gui](#), [GuiElement.Name](#), [GuiElement.Enabled](#), [GuiElement.Interactable](#),
[GuiElement.AnchorPoint](#), [GuiElement.Offset](#), [GuiElement.Size](#), [GuiElement.Scale](#), [GuiElement.Origin](#),
[GuiElement.Rotation](#), [GuiElement.IsHovered](#), [GuiElement.IsClicked](#), [GuiElement.IsSelected](#),
[GuiElement.Position](#), [GuiElement.ScaledSize](#), [GuiElement.AfterUpdate\(double\)](#),
[GuiElement.FixedUpdate\(double\)](#), [GuiElement.Resize\(Rectangle\)](#), [GuiElement.CalculatePos\(\)](#),
[GuiElement.CalculateSize\(\)](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),
[object.GetHashCode\(\)](#), [object.GetType\(\)](#), [object.MemberwiseClone\(\)](#),
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Constructors

TextureTextBoxElement(TextureTextBoxData, LabelData, LabelData, Anchor, Vector2, int, TextAlignment, Vector2?, (float Left, float Right)?, Vector2?, Vector2?, Vector2?, float, Func<GuiElement, bool>?)

Initializes a new instance of the [TextureTextBoxElement](#) class.

```
public TextureTextBoxElement(TextureTextBoxData textBoxData, LabelData labelData, LabelData hintLabelData, Anchor anchor, Vector2 offset, int maxTextLength, TextAlignment textAlignment = TextAlignment.Left, Vector2? textOffset = null, (float Left, float Right)? textEdgeOffset = null, Vector2? size = null, Vector2? scale = null, Vector2? origin = null, float rotation = 0, Func<GuiElement, bool>? clickFunc = null)
```

Parameters

`textBoxData` [TextureTextBoxData](#)

Configuration for the texture and visual appearance of the textbox.

`labelData` [LabelData](#)

Configuration for the primary text label.

`hintLabelData` [LabelData](#)

Configuration for the hint text label.

`anchor` [Anchor](#)

Defines the anchor point for positioning the element.

`offset` [Vector2](#)?

Position offset relative to the anchor.

`maxTextLength` [int](#)

Maximum number of allowed characters in the textbox.

`textAlignment` [TextAlignment](#)

Text alignment within the textbox. Default is Left.

`textOffset` [Vector2](#)?

The offset of the text relative to its position.

`textEdgeOffset` ([float](#) [Left](#), [float](#) [Right](#))?

Optional offsets for the left and right edges of the text. Default is (0.0F, 0.0F).

`size` [Vector2](#)?

Optional size of the textbox. Defaults to the texture dimensions.

`scale` [Vector2](#)?

Optional scale factor for the element.

`origin` [Vector2](#)?

Optional origin point for rotation and scaling. Default is the center.

`rotation` [float](#)

Optional rotation angle (in radians). Default is 0.

`clickFunc` [Func](#)<[GuiElement](#), [bool](#)>

Optional custom function to execute on click. Returns true if the event is consumed.

Fields

TextAlignment

Specifies the alignment of text within the textbox (e.g., Left, Center, Right).

```
public TextAlignment TextAlignment
```

Field Value

[TextAlignment](#)

TextEdgeOffset

The horizontal offsets to be applied to the text edges within the textbox.

```
public (float Left, float Right) TextEdgeOffset
```

Field Value

([float](#) [angular](#), [float](#) [linear](#))

TextOffset

The offset of the text relative to its position.

```
public Vector2 TextOffset
```

Field Value

Properties

HintLabelData

Holds the configuration for the hint label text, shown when the textbox is empty.

```
public LabelData HintLabelData { get; }
```

Property Value

[LabelData](#)

LabelData

Holds the configuration for the label text displayed within the textbox.

```
public LabelData LabelData { get; }
```

Property Value

[LabelData](#)

MaxTextLength

The maximum number of characters allowed within the textbox.

```
public int MaxTextLength { get; }
```

Property Value

[int](#)

TextBoxData

Holds the configuration for the texture-based textbox element.

```
public TextureTextBoxData TextBoxData { get; }
```

Property Value

[TextureTextBoxData](#)

Methods

Draw(GraphicsContext, Framebuffer)

Draws the GUI element on the specified framebuffer using the provided graphics context.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering.

framebuffer [Framebuffer](#)

The framebuffer to which the element is rendered.

Update(double, ref bool)

Updates the state of the [TextureTextBoxElement](#).

```
protected override void Update(double delta, ref bool interactionHandled)
```

Parameters

delta [double](#)

Elapsed time in seconds since the last update.

interactionHandled [bool](#)

A reference to a boolean tracking whether interaction has already been handled by another element.

Class ToggleElement

Namespace: [Sparkle.CSharp.GUI.Elements](#)

Assembly: Sparkle.dll

```
public class ToggleElement : GuiElement
```

Inheritance

[object](#) ← [GuiElement](#) ← ToggleElement

Inherited Members

[GuiElement.Gui](#) , [GuiElement.Name](#) , [GuiElement.Enabled](#) , [GuiElement.Interactable](#) ,
[GuiElement.AnchorPoint](#) , [GuiElement.Offset](#) , [GuiElement.Size](#) , [GuiElement.Scale](#) , [GuiElement.Origin](#) ,
[GuiElement.Rotation](#) , [GuiElement.IsHovered](#) , [GuiElement.IsClicked](#) , [GuiElement.IsSelected](#) ,
[GuiElement.Position](#) , [GuiElement.ScaledSize](#) , [GuiElement.AfterUpdate\(double\)](#) ,
[GuiElement.FixedUpdate\(double\)](#) , [GuiElement.Resize\(Rectangle\)](#) , [GuiElement.CalculatePos\(\)](#) ,
[GuiElement.CalculateSize\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

ToggleElement(ToggleData, LabelData, Anchor, Vector2, float, bool, Vector2?, Vector2?, float, Func<GuiElement, bool>?)

Initializes a new instance of the [ToggleElement](#) class.

```
public ToggleElement(ToggleData toggleData, LabelData labelData, Anchor anchor, Vector2 offset, float boxTextSpacing, bool toggleState = false, Vector2? scale = null, Vector2? origin = null, float rotation = 0, Func<GuiElement, bool>? clickFunc = null)
```

Parameters

toggleData [ToggleData](#)

The toggle rendering data.

labelData [LabelData](#)

The label rendering data.

anchor [Anchor](#)

Anchor position of the element.

offset [Vector2](#)

Offset from the anchor.

boxTextSpacing [float](#)

Spacing between the toggle box and the label text.

toggleState [bool](#)

Initial toggle state.

scale [Vector2](#)?

The scale applied to the toggle element.

origin [Vector2](#)?

Optional custom origin.

rotation [float](#)

Optional rotation angle.

clickFunc [Func](#)<[GuiElement](#), [bool](#)>

Optional function determining click behavior.

Fields

ToggleState

Indicates whether the toggle is currently in the 'off' or 'on' state.

```
public bool ToggleState
```

Field Value

[bool](#)

Properties

BoxTextSpacing

The spacing between the toggle box and the label text.

```
public float BoxTextSpacing { get; }
```

Property Value

[float](#)

LabelData

Holds the configuration for the label text displayed within the toggle.

```
public LabelData LabelData { get; }
```

Property Value

[LabelData](#)

ToggleData

Holds the configuration for the texture-based toggle element.

```
public ToggleData ToggleData { get; }
```

Property Value

[ToggleData](#)

Methods

Draw(GraphicsContext, Framebuffer)

Draws the GUI element on the specified framebuffer using the provided graphics context.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering.

framebuffer [Framebuffer](#)

The framebuffer to which the element is rendered.

Update(double, ref bool)

Updates the toggle state each frame.

```
protected override void Update(double delta, ref bool interactionHandled)
```

Parameters

delta [double](#)

The elapsed time since the last frame.

interactionHandled [bool](#)

A reference to a boolean tracking whether interaction has already been handled by another element.

Events

Toggled

Event invoked when the toggle state changes.

```
public event Action<bool>? Toggled
```

Event Type

[Action](#) <[bool](#)>

Namespace Sparkle.CSharp.GUI.Elements.Data Classes

[LabelData](#)

[RectangleButtonData](#)

[RectangleSlideBarData](#)

[RectangleTextBoxData](#)

[TextureButtonData](#)

[TextureDropDownData](#)

[TextureSlideBarData](#)

[TextureTextBoxData](#)

[ToggleData](#)

Class LabelData

Namespace: [Sparkle.CSharp.GUI.Elements.Data](#)

Assembly: Sparkle.dll

```
public class LabelData
```

Inheritance

[object](#) ← LabelData

Inherited Members

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

Constructors

LabelData(Font, string, float, float, float, Color?, Color?, Color?,
TextStyle, FontSystemEffect, int, Sampler?)

Constructs a new instance of [LabelData](#) with full customization for rendering text.

```
public LabelData(Font font, string text, float size, float characterSpacing = 0, float  
lineSpacing = 0, Color? color = null, Color? hoverColor = null, Color? disabledColor = null,  
TextStyle style = TextStyle.None, FontSystemEffect effect = FontSystemEffect.None, int  
effectAmount = 0, Sampler? sampler = null)
```

Parameters

font Font

The font to use for rendering the text.

text [string](#)

The text content to display.

size [float](#)

The font size of the text.

characterSpacing [float](#)

Spacing between characters. Default is 0.

lineSpacing [float](#)

Spacing between lines. Default is 0.

color Color?

Optional color of the text. Defaults to white.

hoverColor Color?

Optional color of the text. Defaults the same as the color parameter.

disabledColor Color?

The color of the text when the label is in a disabled state. Defaults to gray if not specified.

style TextStyle

The style of the text. Default is `TextStyle.None`.

effect FontSystemEffect

The font effect applied. Default is `FontSystemEffect.None`.

effectAmount [int](#)

The intensity of the font effect. Default is 0.

sampler [Sampler](#)

The texture sampler used for rendering operations in the label.

Fields

CharacterSpacing

The spacing between individual characters.

public float CharacterSpacing

Field Value

[float](#) ↗

Color

The color of the rendered text.

```
public Color Color
```

Field Value

Color

DisabledColor

The color applied when the toggle is in an inactive or disabled state.

```
public Color DisabledColor
```

Field Value

Color

Effect

The special visual effect applied to the font (e.g., shadow, outline).

```
public FontSystemEffect Effect
```

Field Value

FontSystemEffect

EffectAmount

The intensity or size of the font effect.

```
public int EffectAmount
```

Field Value

[int](#)

Font

The font used to render the text.

```
public Font Font
```

Field Value

Font

HoverColor

The color of the text when the label is hovered over.

```
public Color HoverColor
```

Field Value

Color

LineSpacing

The spacing between lines of text.

```
public float LineSpacing
```

Field Value

[float](#)

Sampler

The texture sampler used for rendering operations in the label.

```
public Sampler? Sampler
```

Field Value

[Sampler](#)

Size

The font size of the text.

```
public float Size
```

Field Value

[float](#)

Style

The style of the text, such as bold or italic.

```
public TextStyle Style
```

Field Value

TextStyle

Text

The text to be displayed.

```
public string Text
```

Field Value

[string](#) ↗

Class RectangleButtonData

Namespace: [Sparkle.CSharp.GUI.Elements.Data](#)

Assembly: Sparkle.dll

```
public class RectangleButtonData
```

Inheritance

[object](#) ← RectangleButtonData

Inherited Members

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

Constructors

RectangleButtonData(Color?, Color?, Color?, float, Color?, Color?, Color?)

Initializes a new instance of the [RectangleButtonData](#) class.

```
public RectangleButtonData(Color? color = null, Color? hoverColor = null, Color?  
disabledColor = null, float outlineThickness = 0, Color? outlineColor = null, Color?  
outlineHoverColor = null, Color? disabledOutlineColor = null)
```

Parameters

color Color?

The base fill color of the button. Defaults to white.

hoverColor Color?

The fill color when the button is hovered. Defaults to the base color.

disabledColor Color?

The fill color of the button when it is disabled. Defaults to gray.

outlineThickness [float](#)

The thickness of the button outline. Defaults to 0 (no outline).

outlineColor Color?

The color of the outline. Defaults to white.

outlineHoverColor Color?

The outline color when hovered. Defaults to the regular outline color.

disabledOutlineColor Color?

The outline color when the button is disabled. Defaults to dark gray.

Fields

Color

The base fill color of the button.

```
public Color Color
```

Field Value

Color

DisabledColor

The color applied when the toggle is in an inactive or disabled state.

```
public Color DisabledColor
```

Field Value

Color

DisabledOutlineColor

The color applied when the toggle is in an inactive or disabled state.

```
public Color DisabledOutlineColor
```

Field Value

Color

HoverColor

The fill color of the button when it is hovered.

```
public Color HoverColor
```

Field Value

Color

OutlineColor

The color of the outline.

```
public Color OutlineColor
```

Field Value

Color

OutlineHoverColor

The color of the outline when the button is hovered.

```
public Color OutlineHoverColor
```

Field Value

Color

OutlineThickness

The thickness of the button's outline. Set to 0 to disable outlining.

```
public float OutlineThickness
```

Field Value

[float](#)

Class RectangleSlideBarData

Namespace: [Sparkle.CSharp.GUI.Elements.Data](#)

Assembly: Sparkle.dll

```
public class RectangleSlideBarData
```

Inheritance

[object](#) ← RectangleSlideBarData

Inherited Members

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

Constructors

RectangleSlideBarData(Color?, Color?, Color?, float, Color?,
Color?, Color?, Color?, Color?, Color?, float, Color?, Color?,
Color?, Color?, Color?, float, Color?, Color?, Color?)

Initializes a new instance of the [RectangleSlideBarData](#) class.

```
public RectangleSlideBarData(Color? barColor = null, Color? barHoverColor = null, Color?  
disabledBarColor = null, float barOutlineThickness = 0, Color? barOutlineColor = null,  
Color? barOutlineHoverColor = null, Color? disabledBarOutlineColor = null, Color?  
filledBarColor = null, Color? filledBarHoverColor = null, Color? disabledFilledBarColor =  
null, float filledBarOutlineThickness = 0, Color? filledBarOutlineColor = null, Color?  
filledBarOutlineHoverColor = null, Color? disabledFilledBarOutlineColor = null, Color?  
sliderColor = null, Color? sliderHoverColor = null, Color? disabledSliderColor = null, float  
sliderOutlineThickness = 0, Color? sliderOutlineColor = null, Color? sliderOutlineHoverColor  
= null, Color? disabledSliderOutlineColor = null)
```

Parameters

barColor Color?

The base color used to render the slide bar background.

barHoverColor Color?

The color applied to the slide bar background when hovered.

disabledBarColor Color?

The color applied to the slide bar background when disabled.

barOutlineThickness [float ↗](#)

The thickness of the outline drawn around the slide bar background.

barOutlineColor Color?

The color of the slide bar background outline.

barOutlineHoverColor Color?

The color of the slide bar background outline when hovered.

disabledBarOutlineColor Color?

The color of the slide bar background outline when disabled.

filledBarColor Color?

The base color used to render the filled portion of the slide bar.

filledBarHoverColor Color?

The color applied to the filled portion when hovered.

disabledFilledBarColor Color?

The color applied to the filled portion when disabled.

filledBarOutlineThickness [float ↗](#)

The thickness of the outline drawn around the filled portion.

filledBarOutlineColor Color?

The color of the filled portion outline.

filledBarOutlineHoverColor Color?

The color of the filled portion outline when hovered.

disabledFilledBarOutlineColor Color?

The color of the filled portion outline when disabled.

sliderColor Color?

The base color used to render the slider handle.

sliderHoverColor Color?

The color applied to the slider handle when hovered.

disabledSliderColor Color?

The color applied to the slider handle when disabled.

sliderOutlineThickness [float ↗](#)

The thickness of the outline drawn around the slider handle.

sliderOutlineColor Color?

The color of the slider handle outline.

sliderOutlineHoverColor Color?

The color of the slider handle outline when hovered.

disabledSliderOutlineColor Color?

The color of the slider handle outline when disabled.

Fields

BarColor

The base color used to render the slide bar background.

public Color BarColor

Field Value

Color

BarHoverColor

The color applied to the slide bar background when hovered.

```
public Color BarHoverColor
```

Field Value

Color

BarOutlineColor

The color of the slide bar background outline.

```
public Color BarOutlineColor
```

Field Value

Color

BarOutlineHoverColor

The color of the slide bar background outline when hovered.

```
public Color BarOutlineHoverColor
```

Field Value

Color

BarOutlineThickness

The thickness of the outline drawn around the slide bar background.

```
public float BarOutlineThickness
```

Field Value

[float](#)

DisabledBarColor

The color applied to the slide bar background when disabled.

`public Color DisabledBarColor`

Field Value

Color

DisabledBarOutlineColor

The color of the slide bar background outline when disabled.

`public Color DisabledBarOutlineColor`

Field Value

Color

DisabledFilledBarColor

The color applied to the filled portion of the slide bar when disabled.

`public Color DisabledFilledBarColor`

Field Value

Color

DisabledFilledBarOutlineColor

The color of the filled portion outline when disabled.

```
public Color DisabledFilledBarOutlineColor
```

Field Value

Color

DisabledSliderColor

The color applied to the slider handle when disabled.

```
public Color DisabledSliderColor
```

Field Value

Color

DisabledSliderOutlineColor

The color of the slider handle outline when disabled.

```
public Color DisabledSliderOutlineColor
```

Field Value

Color

FilledBarColor

The base color used to render the filled portion of the slide bar.

```
public Color FilledBarColor
```

Field Value

Color

FilledBarHoverColor

The color applied to the filled portion of the slide bar when hovered.

```
public Color FilledBarHoverColor
```

Field Value

Color

FilledBarOutlineColor

The color of the filled portion outline.

```
public Color FilledBarOutlineColor
```

Field Value

Color

FilledBarOutlineHoverColor

The color of the filled portion outline when hovered.

```
public Color FilledBarOutlineHoverColor
```

Field Value

Color

FilledBarOutlineThickness

The thickness of the outline drawn around the filled portion of the slide bar.

```
public float FilledBarOutlineThickness
```

Field Value

[float ↗](#)

SliderColor

The base color used to render the slider handle.

```
public Color SliderColor
```

Field Value

Color

SliderHoverColor

The color applied to the slider handle when hovered.

```
public Color SliderHoverColor
```

Field Value

Color

SliderOutlineColor

The color of the slider handle outline.

```
public Color SliderOutlineColor
```

Field Value

Color

SliderOutlineHoverColor

The color of the slider handle outline when hovered.

```
public Color SliderOutlineHoverColor
```

Field Value

Color

SliderOutlineThickness

The thickness of the outline drawn around the slider handle.

```
public float SliderOutlineThickness
```

Field Value

[float](#)

SliderSize

The optional size override for the slider handle.

```
public Vector2? SliderSize
```

Field Value

[Vector2](#)?

Class RectangleTextBoxData

Namespace: [Sparkle.CSharp.GUI.Elements.Data](#)

Assembly: Sparkle.dll

```
public class RectangleTextBoxData
```

Inheritance

[object](#) ← RectangleTextBoxData

Inherited Members

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

Constructors

RectangleTextBoxData(Color?, Color?, Color?, float, Color?,
Color?, Color?, Color?)

Initializes a new instance of the [RectangleButtonData](#) class.

```
public RectangleTextBoxData(Color? color = null, Color? hoverColor = null, Color?  
disabledColor = null, float outlineThickness = 0, Color? outlineColor = null, Color?  
outlineHoverColor = null, Color? highlightColor = null, Color? disabledOutlineColor = null)
```

Parameters

color Color?

The base fill color of the text box. Defaults to white.

hoverColor Color?

The fill color when the text box is hovered. Defaults to the base color.

disabledColor Color?

The fill color when the text box is disabled. Defaults to gray.

outlineThickness [float](#)

The thickness of the outline of the text box. Defaults to 0 (no outline).

outlineColor Color?

The color of the text box outline. Defaults to white.

outlineHoverColor Color?

The color of the outline when the text box is hovered. Defaults to the regular outline color.

highlightColor Color?

The color used for the text box highlight. Defaults to a semi-transparent blue.

disabledOutlineColor Color?

The color of the outline when the text box is disabled. Defaults to dark gray.

Fields

Color

The base fill color of the text box.

```
public Color Color
```

Field Value

Color

DisabledColor

The color applied when the toggle is in an inactive or disabled state.

```
public Color DisabledColor
```

Field Value

Color

DisabledOutlineColor

The color applied when the toggle is in an inactive or disabled state.

```
public Color DisabledOutlineColor
```

Field Value

Color

HighlightColor

The color for the highlight.

```
public Color HighlightColor
```

Field Value

Color

HoverColor

The fill color of the text box when it is hovered.

```
public Color HoverColor
```

Field Value

Color

OutlineColor

The color of the outline.

```
public Color OutlineColor
```

Field Value

Color

OutlineHoverColor

The color of the outline when the text box is hovered.

```
public Color OutlineHoverColor
```

Field Value

Color

OutlineThickness

The thickness of the text box's outline. Set to 0 to disable outlining.

```
public float OutlineThickness
```

Field Value

[float](#)

Class TextureButtonData

Namespace: [Sparkle.CSharp.GUI.Elements.Data](#)

Assembly: Sparkle.dll

```
public class TextureButtonData
```

Inheritance

[object](#) ← TextureButtonData

Inherited Members

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

Constructors

TextureButtonData(Texture2D, Sampler?, Rectangle?,
ResizeMode, BorderInsets?, Color?, Color?, Color?, SpriteFlip)

Initializes a new instance of the [TextureButtonData](#) class.

```
public TextureButtonData(Texture2D texture, Sampler? sampler = null, Rectangle? sourceRect  
= null, ResizeMode resizeMode = ResizeMode.None, BorderInsets? borderInsets = null, Color?  
color = null, Color? hoverColor = null, Color? disabledColor = null, SpriteFlip flip  
= SpriteFlip.None)
```

Parameters

texture Texture2D

The texture used for the button.

sampler [Sampler](#)

The sampler used to control how the texture is sampled.

sourceRect Rectangle?

The section of the texture to be displayed.

resizeMode [ResizeMode](#)

The mode used to resize the texture.

borderInsets [BorderInsets?](#)

The border insets used for nine-slice resizing.

color Color?

The primary color.

hoverColor Color?

The color gets applied when the mouse is hover over.

disabledColor Color?

The color of the button when it is disabled. Defaults to gray if not specified.

flip SpriteFlip

The flip mode for the texture (none, horizontal, vertical, both).

Fields

BorderInsets

The border insets used for nine-slice resizing.

```
public BorderInsets BorderInsets
```

Field Value

[BorderInsets](#)

Color

The primary color.

```
public Color Color
```

Field Value

Color

DisabledColor

The color applied when the toggle is in an inactive or disabled state.

```
public Color DisabledColor
```

Field Value

Color

Flip

The flip mode for the texture (none, horizontal, vertical, both).

```
public SpriteFlip Flip
```

Field Value

SpriteFlip

HoverColor

The color get's applied when the mouse is hover over.

```
public Color HoverColor
```

Field Value

Color

ResizeMode

The mode used to resize the texture.

```
public ResizeMode ResizeMode
```

Field Value

[ResizeMode](#)

Sampler

The sampler used to control how the texture is sampled.

```
public Sampler? Sampler
```

Field Value

[Sampler](#)

SourceRect

The section of the texture to be displayed.

```
public Rectangle SourceRect
```

Field Value

Rectangle

Texture

The texture used for the button.

```
public Texture2D Texture
```

Field Value

Class TextureDropDownData

Namespace: [Sparkle.CSharp.GUI.Elements.Data](#)

Assembly: Sparkle.dll

```
public class TextureDropDownData
```

Inheritance

[object](#) ← TextureDropDownData

Inherited Members

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

Constructors

TextureDropDownData(Texture2D, Texture2D, Texture2D, Texture2D, Texture2D?, Sampler?, Sampler?, Sampler?, Sampler?, Sampler?, Sampler?, Rectangle?, Rectangle?, Rectangle?, Rectangle?, Rectangle?, Rectangle?, Rectangle?, ResizeMode, ResizeMode, ResizeMode, BorderInsets?, BorderInsets?, BorderInsets?, Color?, SpriteFlip, SpriteFlip, SpriteFlip, SpriteFlip, int)

Initializes a new instance of the [TextureDropDownData](#) class, defining all visual, layout, and interaction states for a texture-based dropdown.

```
public TextureDropDownData(Texture2D fieldTexture, Texture2D menuTexture, Texture2D sliderBarTexture, Texture2D sliderTexture, Texture2D? arrowTexture, Sampler? fieldSampler = null, Sampler? menuSampler = null, Sampler? sliderBarSampler = null, Sampler? sliderSampler = null, Sampler? arrowSampler = null, Rectangle? fieldSourceRect = null, Rectangle? menuSourceRect = null, Rectangle? sliderBarSourceRect = null, Rectangle? sliderSourceRect = null, Rectangle? arrowSourceRect = null, ResizeMode fieldResizeMode = ResizeMode.None, ResizeMode menuResizeMode = ResizeMode.None, ResizeMode sliderBarResizeMode = ResizeMode.None, BorderInsets? fieldBorderInsets = null, BorderInsets? menuBorderInsets = null, BorderInsets? sliderBarBorderInsets = null, Color? fieldColor = null, Color? menuColor = null)
```

```
= null, Color? sliderBarColor = null, Color? sliderColor = null, Color? arrowColor = null,
Color? fieldHoverColor = null, Color? menuHoverColor = null, Color? sliderBarHoverColor =
null, Color? sliderHoverColor = null, Color? arrowHoverColor = null, Color? highlightColor =
null, Color? disabledFieldColor = null, Color? disabledMenuColor = null, Color?
disabledSliderBarColor = null, Color? disabledSliderColor = null, Color? disabledArrowColor
= null, SpriteFlip fieldFlip = SpriteFlip.None, SpriteFlip menuFlip = SpriteFlip.None,
SpriteFlip sliderBarFlip = SpriteFlip.None, SpriteFlip sliderFlip = SpriteFlip.None,
SpriteFlip arrowFlip = SpriteFlip.None, int scrollBarWidth = 16)
```

Parameters

fieldTexture Texture2D

The texture used to render the dropdown field in its collapsed state.

menuTexture Texture2D

The texture used to render the dropdown menu background.

sliderBarTexture Texture2D

Optional texture for the dropdown menu's scrollbar track.

sliderTexture Texture2D

Optional texture for the dropdown menu's scrollbar handle.

arrowTexture Texture2D

The texture used to render the dropdown arrow indicator.

fieldSampler [Sampler](#)

Optional sampler used for sampling the field texture.

menuSampler [Sampler](#)

Optional sampler used for sampling the menu texture.

sliderBarSampler [Sampler](#)

Optional sampler used for sampling the slider bar texture.

sliderSampler [Sampler](#)

Optional sampler used for sampling the slider texture.

arrowSampler [Sampler](#)

Optional sampler used for sampling the arrow texture.

fieldSourceRect Rectangle?

Optional source rectangle for the field texture. Defaults to the full texture.

menuSourceRect Rectangle?

Optional source rectangle for the menu texture. Defaults to the full texture.

sliderBarSourceRect Rectangle?

Optional source rectangle for the slider bar texture. Defaults to the full texture.

sliderSourceRect Rectangle?

Optional source rectangle for the slider texture. Defaults to the full texture.

arrowSourceRect Rectangle?

Optional source rectangle for the arrow texture. Defaults to the full texture.

fieldResizeMode [ResizeMode](#)

Resize mode applied to the field texture.

menuResizeMode [ResizeMode](#)

Resize mode applied to the menu texture.

sliderBarResizeMode [ResizeMode](#)

Resize mode applied to the slider bar texture.

fieldBorderInsets [BorderInsets](#)?

Border insets used for nine-slice resizing of the field texture.

menuBorderInsets [BorderInsets](#)?

Border insets used for nine-slice resizing of the menu texture.

sliderBarBorderInsets [BorderInsets](#)?

Border insets used for nine-slice resizing of the slider bar texture.

fieldColor Color?

Base color applied to the dropdown field.

menuColor Color?

Base color applied to the dropdown menu.

sliderBarColor Color?

Base color applied to the scrollbar bar.

sliderColor Color?

Base color applied to the scrollbar handle.

arrowColor Color?

Base color applied to the dropdown arrow.

fieldHoverColor Color?

Color applied to the field when hovered.

menuHoverColor Color?

Color applied to the menu when hovered.

sliderBarHoverColor Color?

Color applied to the scrollbar bar when hovered.

sliderHoverColor Color?

Color applied to the scrollbar handle when hovered.

arrowHoverColor Color?

Color applied to the arrow when hovered.

highlightColor Color?

Color used to highlight hovered or selected menu entries.

disabledFieldColor Color?

Color applied to the field when the dropdown is disabled.

disabledMenuColor Color?

Color applied to the menu when the dropdown is disabled.

disabledSliderBarColor Color?

Color applied to the scrollbar bar when the dropdown is disabled.

disabledSliderColor Color?

Color applied to the scrollbar handle when the dropdown is disabled.

disabledArrowColor Color?

Color applied to the arrow when the dropdown is disabled.

fieldFlip SpriteFlip

Flip mode applied to the field texture.

menuFlip SpriteFlip

Flip mode applied to the menu texture.

sliderBarFlip SpriteFlip

Flip mode applied to the slider bar texture.

sliderFlip SpriteFlip

Flip mode applied to the slider texture.

arrowFlip SpriteFlip

Flip mode applied to the arrow texture.

scrollBarWidth int ↴

The width of the scrollbar in the dropdown menu.

Fields

ArrowColor

The base color applied to the dropdown arrow.

```
public Color ArrowColor
```

Field Value

Color

ArrowFlip

The flip mode applied to the dropdown arrow texture.

```
public SpriteFlip ArrowFlip
```

Field Value

SpriteFlip

ArrowHoverColor

The color applied to the dropdown arrow when hovered.

```
public Color ArrowHoverColor
```

Field Value

Color

ArrowSampler

The sampler used for sampling the dropdown arrow texture.

```
public Sampler? ArrowSampler
```

Field Value

[Sampler](#)

ArrowSourceRect

The source rectangle defining the region of the arrow texture to draw.

```
public Rectangle ArrowSourceRect
```

Field Value

Rectangle

ArrowTexture

The texture used to render the dropdown arrow indicator.

```
public Texture2D? ArrowTexture
```

Field Value

Texture2D

DisabledArrowColor

The color applied to the dropdown arrow when disabled.

```
public Color DisabledArrowColor
```

Field Value

Color

DisabledFieldColor

The color applied to the dropdown field when disabled.

```
public Color DisabledFieldColor
```

Field Value

Color

DisabledMenuColor

The color applied to the dropdown menu when disabled.

```
public Color DisabledMenuColor
```

Field Value

Color

DisabledSliderBarColor

The color applied to the scrollbar track when the dropdown is disabled.

```
public Color DisabledSliderBarColor
```

Field Value

Color

DisabledSliderColor

The color applied to the scrollbar handle when the dropdown is disabled.

```
public Color DisabledSliderColor
```

Field Value

Color

FieldBorderInsets

The border insets used for nine-slice resizing of the field texture.

```
public BorderInsets FieldBorderInsets
```

Field Value

[BorderInsets](#)

FieldColor

The base color applied to the dropdown field.

```
public Color FieldColor
```

Field Value

Color

FieldFlip

The flip mode applied to the dropdown field texture.

```
public SpriteFlip FieldFlip
```

Field Value

SpriteFlip

FieldHoverColor

The color applied to the dropdown field when hovered.

```
public Color FieldHoverColor
```

Field Value

Color

FieldResizeMode

The resize mode applied to the dropdown field texture.

```
public ResizeMode FieldResizeMode
```

Field Value

[ResizeMode](#)

FieldSampler

The sampler used for sampling the dropdown field texture.

```
public Sampler? FieldSampler
```

Field Value

[Sampler](#)

FieldSourceRect

The source rectangle defining the region of the field texture to draw.

```
public Rectangle FieldSourceRect
```

Field Value

Rectangle

FieldTexture

The texture used to render the dropdown field (collapsed state).

```
public Texture2D FieldTexture
```

Field Value

Texture2D

HighlightColor

The color used to highlight hovered or selected menu items.

```
public Color HighlightColor
```

Field Value

Color

MenuBorderInsets

The border insets used for nine-slice resizing of the menu texture.

```
public BorderInsets MenuBorderInsets
```

Field Value

[BorderInsets](#)

MenuColor

The base color applied to the dropdown menu.

```
public Color MenuColor
```

Field Value

Color

MenuFlip

The flip mode applied to the dropdown menu texture.

```
public SpriteFlip MenuFlip
```

Field Value

SpriteFlip

MenuHoverColor

The color applied to the dropdown menu when hovered.

```
public Color MenuHoverColor
```

Field Value

Color

MenuResizeMode

The resize mode applied to the dropdown menu texture.

```
public ResizeMode MenuResizeMode
```

Field Value

[ResizeMode](#)

MenuSampler

The sampler used for sampling the dropdown menu texture.

```
public Sampler? MenuSampler
```

Field Value

[Sampler](#)

MenuSourceRect

The source rectangle defining the region of the menu texture to draw.

```
public Rectangle MenuSourceRect
```

Field Value

Rectangle

MenuTexture

The texture used to render the dropdown menu background.

```
public Texture2D MenuTexture
```

Field Value

Texture2D

ScrollBarWidth

The width of the scrollbar rendered inside the dropdown menu.

```
public int ScrollBarWidth
```

Field Value

[int](#)

SliderBarBorderInsets

The border insets used for nine-slice resizing of the scrollbar track texture.

```
public BorderInsets SliderBarBorderInsets
```

Field Value

[BorderInsets](#)

SliderBarColor

The base color applied to the scrollbar track.

```
public Color SliderBarColor
```

Field Value

Color

SliderBarFlip

The flip mode applied to the scrollbar track texture.

```
public SpriteFlip SliderBarFlip
```

Field Value

SpriteFlip

SliderBarHoverColor

The color applied to the scrollbar track when hovered.

```
public Color SliderBarHoverColor
```

Field Value

Color

SliderBarResizeMode

The resize mode applied to the scrollbar track texture.

```
public ResizeMode SliderBarResizeMode
```

Field Value

[ResizeMode](#)

SliderBarSampler

The sampler used for sampling the scrollbar track texture.

```
public Sampler? SliderBarSampler
```

Field Value

[Sampler](#)

SliderBarSourceRect

The source rectangle defining the region of the scrollbar track texture to draw.

```
public Rectangle SliderBarSourceRect
```

Field Value

Rectangle

SliderBarTexture

The texture used to render the scrollbar track inside the dropdown menu.

```
public Texture2D SliderBarTexture
```

Field Value

Texture2D

SliderColor

The base color applied to the scrollbar handle.

```
public Color SliderColor
```

Field Value

Color

SliderFlip

The flip mode applied to the scrollbar handle texture.

```
public SpriteFlip SliderFlip
```

Field Value

SpriteFlip

SliderHoverColor

The color applied to the scrollbar handle when hovered.

```
public Color SliderHoverColor
```

Field Value

Color

SliderSampler

The sampler used for sampling the scrollbar handle texture.

```
public Sampler? SliderSampler
```

Field Value

[Sampler](#)

SliderSourceRect

The source rectangle defining the region of the scrollbar handle texture to draw.

```
public Rectangle SliderSourceRect
```

Field Value

Rectangle

SliderTexture

The texture used to render the scrollbar handle inside the dropdown menu.

```
public Texture2D SliderTexture
```

Field Value

Texture2D

Class TextureSlideBarData

Namespace: [Sparkle.CSharp.GUI.Elements.Data](#)

Assembly: Sparkle.dll

```
public class TextureSlideBarData
```

Inheritance

[object](#) ← TextureSlideBarData

Inherited Members

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

Constructors

TextureSlideBarData(Texture2D, Texture2D?, Texture2D?,
Sampler?, Sampler?, Sampler?, Rectangle?, Rectangle?,
Rectangle?, ResizeMode, ResizeMode, BorderInsets?,
BorderInsets?, Color?, Color?, Color?, Color?, Color?,
Color?, Color?, Color?, SpriteFlip, SpriteFlip, SpriteFlip)

Initializes a new instance of the [TextureSlideBarData](#) class.

```
public TextureSlideBarData(Texture2D barTexture, Texture2D? filledBarTexture, Texture2D?  
sliderTexture, Sampler? barSampler = null, Sampler? filledBarSampler = null, Sampler?  
sliderSampler = null, Rectangle? barSourceRect = null, Rectangle? filledBarSourceRect =  
null, Rectangle? sliderSourceRect = null, ResizeMode barResizeMode = ResizeMode.None,  
ResizeMode filledBarResizeMode = ResizeMode.None, BorderInsets? barBorderInsets = null,  
BorderInsets? filledBarBorderInsets = null, Color? barColor = null, Color? filledBarColor =  
null, Color? sliderColor = null, Color? barHoverColor = null, Color? filledBarHoverColor =  
null, Color? sliderHoverColor = null, Color? disabledBarColor = null, Color?  
disabledFilledBarColor = null, Color? disabledSliderColor = null, SpriteFlip barFlip  
= SpriteFlip.None, SpriteFlip filledBarFlip = SpriteFlip.None, SpriteFlip sliderFlip  
= SpriteFlip.None)
```

Parameters

barTexture Texture2D

The texture used to render the slide bar background.

filledBarTexture Texture2D

The optional texture used to render the filled portion of the slide bar.

sliderTexture Texture2D

The optional texture used to render the slider handle.

barSampler [Sampler](#)

The sampler used for sampling the bar texture.

filledBarSampler [Sampler](#)

The sampler used for sampling the filled bar texture.

sliderSampler [Sampler](#)

The sampler used for sampling the slider texture.

barSourceRect Rectangle?

The source rectangle defining the visible region of the bar texture.

filledBarSourceRect Rectangle?

The source rectangle defining the visible region of the filled bar texture.

sliderSourceRect Rectangle?

The source rectangle defining the visible region of the slider texture.

barResizeMode [ResizeMode](#)

The resize mode applied to the bar texture.

filledBarResizeMode [ResizeMode](#)

The resize mode applied to the filled bar texture.

barBorderInsets [BorderInsets](#)?

The border insets used for nine-slice resizing of the bar texture.

filledBarBorderInsets BorderInsets?

The border insets used for nine-slice resizing of the filled bar texture.

barColor Color?

The base color applied to the bar texture.

filledBarColor Color?

The base color applied to the filled bar texture.

sliderColor Color?

The base color applied to the slider texture.

barHoverColor Color?

The color applied to the bar texture when hovered.

filledBarHoverColor Color?

The color applied to the filled bar texture when hovered.

sliderHoverColor Color?

The color applied to the slider texture when hovered.

disabledBarColor Color?

The color applied to the bar texture when disabled.

disabledFilledBarColor Color?

The color applied to the filled bar texture when disabled.

disabledSliderColor Color?

The color applied to the slider texture when disabled.

barFlip SpriteFlip

The flip mode applied to the bar texture.

filledBarFlip SpriteFlip

The flip mode applied to the filled bar texture.

sliderFlip SpriteFlip

The flip mode applied to the slider texture.

Fields

BarBorderInsets

The border insets used for nine-slice resizing of the bar texture.

```
public BorderInsets BarBorderInsets
```

Field Value

[BorderInsets](#)

BarColor

The base color applied to the bar texture.

```
public Color BarColor
```

Field Value

Color

BarFlip

The flip mode applied to the bar texture.

```
public SpriteFlip BarFlip
```

Field Value

SpriteFlip

BarHoverColor

The color applied to the bar texture when hovered.

```
public Color BarHoverColor
```

Field Value

Color

BarResizeMode

The resize mode applied to the bar texture.

```
public ResizeMode BarResizeMode
```

Field Value

[ResizeMode](#)

BarSampler

The sampler used for sampling the bar texture.

```
public Sampler? BarSampler
```

Field Value

[Sampler](#)

BarSourceRect

The source rectangle defining the visible region of the bar texture.

```
public Rectangle BarSourceRect
```

Field Value

Rectangle

BarTexture

The texture used to render the slide bar background.

```
public Texture2D BarTexture
```

Field Value

Texture2D

DisabledBarColor

The color applied to the entire slide bar when disabled.

```
public Color DisabledBarColor
```

Field Value

Color

DisabledFilledBarColor

The color applied to the filled bar texture when the slide bar is disabled.

```
public Color DisabledFilledBarColor
```

Field Value

Color

DisabledSliderColor

The color used to render the slider when it is in a disabled state.

```
public Color DisabledSliderColor
```

Field Value

Color

FilledBarBorderInsets

The border insets used for nine-slice resizing of the filled bar texture.

```
public BorderInsets FilledBarBorderInsets
```

Field Value

[BorderInsets](#)

FilledBarColor

The base color applied to the filled bar texture.

```
public Color FilledBarColor
```

Field Value

Color

FilledBarFlip

The flip mode applied to the filled bar texture.

```
public SpriteFlip FilledBarFlip
```

Field Value

FilledBarHoverColor

The color applied to the filled bar texture when hovered.

```
public Color FilledBarHoverColor
```

Field Value

Color

FilledBarResizeMode

The resize mode applied to the filled bar texture.

```
public ResizeMode FilledBarResizeMode
```

Field Value

[ResizeMode](#)

FilledBarSampler

The sampler used for sampling the filled bar texture.

```
public Sampler? FilledBarSampler
```

Field Value

[Sampler](#)

FilledBarSourceRect

The source rectangle defining the visible region of the filled bar texture.

```
public Rectangle FilledBarSourceRect
```

Field Value

Rectangle

FilledBarTexture

The optional texture used to render the filled portion of the slide bar.

```
public Texture2D? FilledBarTexture
```

Field Value

Texture2D

SliderColor

The base color applied to the slider texture.

```
public Color SliderColor
```

Field Value

Color

SliderFlip

The flip mode applied to the slider texture.

```
public SpriteFlip SliderFlip
```

Field Value

SpriteFlip

SliderHoverColor

The color applied to the slider texture when hovered.

```
public Color SliderHoverColor
```

Field Value

Color

SliderSampler

The sampler used for sampling the slider texture.

```
public Sampler? SliderSampler
```

Field Value

[Sampler](#)

SliderSourceRect

The source rectangle defining the visible region of the slider texture.

```
public Rectangle SliderSourceRect
```

Field Value

Rectangle

SliderTexture

The optional texture used to render the slider handle.

```
public Texture2D? SliderTexture
```

Field Value

Texture2D

Class TextureTextBoxData

Namespace: [Sparkle.CSharp.GUI.Elements.Data](#)

Assembly: Sparkle.dll

```
public class TextureTextBoxData
```

Inheritance

[object](#) ← TextureTextBoxData

Inherited Members

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

Constructors

TextureTextBoxData(Texture2D, Sampler?, Rectangle?,
ResizeMode, BorderInsets?, Color?, Color?, Color?, Color?,
SpriteFlip)

Initializes a new instance of the [TextureTextBoxData](#) class.

```
public TextureTextBoxData(Texture2D texture, Sampler? sampler = null, Rectangle? sourceRect  
= null, ResizeMode resizeMode = ResizeMode.None, BorderInsets? borderInsets = null, Color?  
color = null, Color? hoverColor = null, Color? highlightColor = null, Color? disabledColor =  
null, SpriteFlip flip = SpriteFlip.None)
```

Parameters

texture Texture2D

The texture used for the text box.

sampler [Sampler](#)

The sampler used to control how the texture is sampled.

sourceRect Rectangle?

The section of the texture to be displayed.

resizeMode [ResizeMode](#)

The mode used to resize the texture.

borderInsets [BorderInsets?](#)

The border insets used for nine-slice resizing.

color Color?

The primary color.

hoverColor Color?

The color get's applied when the mouse is hover over.

highlightColor Color?

The color for the highlight.

disabledColor Color?

The color displayed when the text box is disabled.

flip SpriteFlip

The flip mode for the texture (none, horizontal, vertical, both).

Fields

BorderInsets

The border insets used for nine-slice resizing.

```
public BorderInsets BorderInsets
```

Field Value

[BorderInsets](#)

Color

The primary color.

```
public Color Color
```

Field Value

Color

DisabledColor

The color applied when the toggle is in an inactive or disabled state.

```
public Color DisabledColor
```

Field Value

Color

Flip

The flip mode for the texture (none, horizontal, vertical, both).

```
public SpriteFlip Flip
```

Field Value

SpriteFlip

HighlightColor

The color for the highlight.

```
public Color HighlightColor
```

Field Value

Color

HoverColor

The color get's applied when the mouse is hover over.

```
public Color HoverColor
```

Field Value

Color

SizeMode

The mode used to resize the texture.

```
public ResizeMode ResizeMode
```

Field Value

[ResizeMode](#)

Sampler

The sampler used to control how the texture is sampled.

```
public Sampler? Sampler
```

Field Value

[Sampler](#) ↗

SourceRect

The section of the texture to be displayed.

```
public Rectangle SourceRect
```

Field Value

Rectangle

Texture

The texture used for the text box.

```
public Texture2D Texture
```

Field Value

Texture2D

Class ToggleData

Namespace: [Sparkle.CSharp.GUI.Elements.Data](#)

Assembly: Sparkle.dll

```
public class ToggleData
```

Inheritance

[object](#) ← ToggleData

Inherited Members

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

Constructors

ToggleData(Texture2D, Texture2D, Sampler?, Sampler?, Rectangle?, Rectangle?, Color?, Color?, Color?, Color?, Color?, SpriteFlip, SpriteFlip)

Initializes a new instance of the [ToggleData](#) class with optional customization.

```
public ToggleData(Texture2D checkboxTexture, Texture2D checkmarkTexture, Sampler?  
checkboxSampler = null, Sampler? checkmarkSampler = null, Rectangle? checkboxSourceRect =  
null, Rectangle? checkmarkSourceRect = null, Color? checkboxColor = null, Color?  
checkmarkColor = null, Color? checkboxHoverColor = null, Color? checkmarkHoverColor = null,  
Color? disabledColor = null, SpriteFlip checkboxFlip = SpriteFlip.None, SpriteFlip  
checkmarkFlip = SpriteFlip.None)
```

Parameters

checkboxTexture Texture2D

The texture for the toggle Checkbox.

checkmarkTexture Texture2D

The texture for the toggle checkmark.

checkboxSampler [Sampler](#)

Optional sampler for the Checkbox texture.

checkmarkSampler [Sampler](#)

Optional sampler for the checkmark texture.

checkboxSourceRect Rectangle?

Optional source rectangle for the Checkbox texture. Defaults to full texture.

checkmarkSourceRect Rectangle?

Optional source rectangle for the checkmark texture. Defaults to full texture.

checkboxColor Color?

Optional base color for the Checkbox. Defaults to white.

checkmarkColor Color?

Optional base color for the checkmark. Defaults to white.

checkboxHoverColor Color?

Optional hover color for the Checkbox. Defaults to `checkboxColor`.

checkmarkHoverColor Color?

Optional hover color for the checkmark. Defaults to `checkmarkColor`.

disabledColor Color?

Optional color for the toggle when disabled. Defaults to gray.

checkboxFlip SpriteFlip

Optional flip setting for the Checkbox texture. Defaults to none.

checkmarkFlip SpriteFlip

Optional flip setting for the checkmark texture. Defaults to none.

Fields

CheckboxColor

The base color applied to the checkbox when not hovered.

```
public Color CheckboxColor
```

Field Value

Color

CheckboxFlip

The flip transformation applied to the checkbox texture.

```
public SpriteFlip CheckboxFlip
```

Field Value

SpriteFlip

CheckboxHoverColor

The color applied to the checkbox when the toggle is hovered.

```
public Color CheckboxHoverColor
```

Field Value

Color

CheckboxSampler

The sampler used when rendering the checkbox texture.

```
public Sampler? CheckboxSampler
```

Field Value

[Sampler](#)

CheckboxSourceRect

The source rectangle of the checkbox texture.

```
public Rectangle CheckboxSourceRect
```

Field Value

Rectangle

CheckboxTexture

The texture used for the checkbox of the toggle box.

```
public Texture2D CheckboxTexture
```

Field Value

Texture2D

CheckmarkColor

The base color applied to the checkmark when not hovered.

```
public Color CheckmarkColor
```

Field Value

Color

CheckmarkFlip

The flip transformation applied to the checkmark texture.

```
public SpriteFlip CheckmarkFlip
```

Field Value

SpriteFlip

CheckmarkHoverColor

The color applied to the checkmark when the toggle is hovered.

```
public Color CheckmarkHoverColor
```

Field Value

Color

CheckmarkSampler

The sampler used when rendering the checkmark texture.

```
public Sampler? CheckmarkSampler
```

Field Value

[Sampler](#)

CheckmarkSourceRect

The source rectangle of the checkmark texture.

```
public Rectangle CheckmarkSourceRect
```

Field Value

Rectangle

CheckmarkTexture

The texture used for the checkmark when the toggle is active.

```
public Texture2D CheckmarkTexture
```

Field Value

Texture2D

DisabledColor

The color applied when the toggle is in an inactive or disabled state.

```
public Color DisabledColor
```

Field Value

Color

Namespace Sparkle.CSharp.Graphics

Classes

[GlobalGraphicsAssets](#)

[GraphicsContext](#)

Class GlobalGraphicsAssets

Namespace: [Sparkle.CSharp.Graphics](#)

Assembly: Sparkle.dll

```
public static class GlobalGraphicsAssets
```

Inheritance

[object](#) ← GlobalGraphicsAssets

Inherited Members

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

Properties

BloomEffect

The shader effect used to apply a bloom filter.

```
public static BloomEffect BloomEffect { get; }
```

Property Value

[BloomEffect](#)

BlurEffect

The shader effect used to apply a blur filter.

```
public static BlurEffect BlurEffect { get; }
```

Property Value

[BlurEffect](#)

FxaaEffect

The FXAA (Fast Approximate Anti-Aliasing) effect used for post-processing rendering.

```
public static FxaaEffect FxaaEffect { get; }
```

Property Value

[FxaaEffect](#)

GraphicsDevice

The graphics device used for rendering.

```
public static GraphicsDevice GraphicsDevice { get; }
```

Property Value

[GraphicsDevice](#)

GrayScaleEffect

The shader effect used to apply a grayscale filter.

```
public static Effect GrayScaleEffect { get; }
```

Property Value

Effect

PhysicsDebugEffect

The shader effect used for rendering physics debug visuals.

```
public static Effect PhysicsDebugEffect { get; }
```

Property Value

Effect

PixelizerEffect

The shader effect used to apply a pixelizer filter.

```
public static PixelizerEffect PixelizerEffect { get; }
```

Property Value

[PixelizerEffect](#)

PosterizationEffect

The shader effect used to apply a posterization filter.

```
public static PosterizationEffect PosterizationEffect { get; }
```

Property Value

[PosterizationEffect](#)

PredatorEffect

The shader effect used to apply a predator filter.

```
public static PredatorEffect PredatorEffect { get; }
```

Property Value

[PredatorEffect](#)

SkyboxEffect

The shader effect used for rendering the skybox.

```
public static Effect SkyboxEffect { get; }
```

Property Value

Effect

SobelEffect

The shader effect used to apply a sobel filter.

```
public static SobelEffect SobelEffect { get; }
```

Property Value

[SobelEffect](#)

Window

The window used for rendering.

```
public static IWindow Window { get; }
```

Property Value

IWindow

Class GraphicsContext

Namespace: [Sparkle.CSharp.Graphics](#)

Assembly: Sparkle.dll

```
public class GraphicsContext
```

Inheritance

[object](#) ← GraphicsContext

Inherited Members

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

Constructors

**GraphicsContext(GraphicsDevice, CommandList,
FullScreenRenderer, SpriteBatch, PrimitiveBatch,
ImmediateRenderer)**

Initializes a new instance of the [GraphicsContext](#) class.

```
public GraphicsContext(GraphicsDevice graphicsDevice, CommandList commandList,  
FullScreenRenderer fullScreenRenderer, SpriteBatch spriteBatch, PrimitiveBatch  
primitiveBatch, ImmediateRenderer immediateRenderer)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device to use.

commandList [CommandList](#)

The command list for issuing rendering commands.

fullScreenRenderer FullScreenRenderer

The render pass used for full-screen rendering operations.

spriteBatch SpriteBatch

The sprite batch for rendering sprites.

primitiveBatch PrimitiveBatch

The primitive batch for rendering primitive shapes.

immediateRenderer ImmediateRenderer

The immediate renderer for performing low-level custom rendering.

Properties

CommandList

The command list for submitting rendering commands.

```
public CommandList CommandList { get; }
```

Property Value

[CommandList](#)

FullScreenRenderer

The full-screen render pass used for post-processing.

```
public FullScreenRenderer FullScreenRenderer { get; }
```

Property Value

FullScreenRenderer

GraphicsDevice

The graphics device used for rendering.

```
public GraphicsDevice GraphicsDevice { get; }
```

Property Value

[GraphicsDevice](#) ↗

ImmediateRenderer

The immediate renderer for low-level rendering operations.

```
public ImmediateRenderer ImmediateRenderer { get; }
```

Property Value

ImmediateRenderer

PrimitiveBatch

The primitive batch for rendering basic geometric shapes.

```
public PrimitiveBatch PrimitiveBatch { get; }
```

Property Value

PrimitiveBatch

SpriteBatch

The sprite batch used for efficient 2D sprite rendering.

```
public SpriteBatch SpriteBatch { get; }
```

Property Value

SpriteBatch

Namespace Sparkle.CSharp.Graphics.Rendering Classes

[MultiInstanceRenderer](#)

[Physics3DDebugDrawer](#)

[SkyBox](#)

Class MultiInstanceRenderer

Namespace: [Sparkle.CSharp.Graphics.Rendering](#)

Assembly: Sparkle.dll

```
public class MultiInstanceRenderer
```

Inheritance

[object](#) ← MultiInstanceRenderer

Inherited Members

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

Constructors

MultiInstanceRenderer(Mesh, Material)

Initializes a new instance of the [MultiInstanceRenderer](#) class using a single mesh and material.

```
public MultiInstanceRenderer(Mesh mesh, Material material)
```

Parameters

mesh Mesh

The mesh to render using instancing.

material Material

The material used for rendering the mesh.

MultiInstanceRenderer(Mesh, bool)

Initializes a new instance of the [MultiInstanceRenderer](#) class using a single mesh.

```
public MultiInstanceRenderer(Mesh mesh, bool copyMeshMaterial = false)
```

Parameters

mesh Mesh

The mesh to render using instancing.

copyMeshMaterial [bool](#)

If true, clones the mesh material instead of using the shared reference.

MultilInstanceRenderer(Model, bool)

Initializes a new instance of the [MultilInstanceRenderer](#) class using all meshes from a model.

```
public MultiInstanceRenderer(Model model, bool copyModelMaterials = false)
```

Parameters

model Model

The model whose meshes will be rendered using instancing.

copyModelMaterials [bool](#)

If true, clones each mesh material to allow independent modification.

Properties

Meshes

Gets the meshes managed by this multi-instance renderer.

```
public Mesh[] Meshes { get; }
```

Property Value

Mesh[]

Methods

GetRenderableBoneMatricesByMesh(Mesh)

Retrieves the bone matrix array associated with the renderable for the specified mesh.

```
public Matrix4x4[]? GetRenderableBoneMatricesByMesh(Mesh mesh)
```

Parameters

mesh Mesh

The mesh whose bone matrices are requested.

Returns

[Matrix4x4](#)[]

The bone matrix array if the mesh is skinned, otherwise null.

GetRenderableMaterialByMesh(Mesh)

Retrieves a reference to the material used by the renderable associated with the specified mesh.

```
public ref Material GetRenderableMaterialByMesh(Mesh mesh)
```

Parameters

mesh Mesh

The mesh whose renderable material is requested.

Returns

Material

A reference to the material assigned to the mesh renderable.

Class Physics3DDrawDrawer

Namespace: [Sparkle.CSharp.Graphics.Rendering](#)

Assembly: Sparkle.dll

```
public class Physics3DDrawDrawer : Disposable, IDisposable, IDebugDrawer
```

Inheritance

[object](#) ← Disposable ← Physics3DDrawDrawer

Implements

[IDisposable](#), IDebugDrawer

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

Physics3DDrawDrawer(GraphicsDevice, IWindow, uint)

Initializes a new instance of the [Physics3DDrawDrawer](#) class with the specified graphics device, window, and optional capacity.

```
public Physics3DDrawDrawer(GraphicsDevice graphicsDevice, IWindow window, uint capacity  
= 30720)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for rendering.

window IWindow

The window associated with the rendering context.

capacity [uint](#)

The maximum number of vertices that can be rendered in a single batch. Default is 30720.

Properties

Capacity

Gets the maximum number of vertices the drawer can store.

```
public uint Capacity { get; }
```

Property Value

[uint](#)

DrawCallCount

Gets the number of draw calls performed during the current frame.

```
public int DrawCallCount { get; }
```

Property Value

[int](#)

GraphicsDevice

Gets the graphics device used by the debug drawer.

```
public GraphicsDevice GraphicsDevice { get; }
```

Property Value

[GraphicsDevice](#)

Window

Gets the window associated with the debug drawer.

```
public IWindow Window { get; }
```

Property Value

IWindow

Methods

**Begin(CommandList, OutputDescription,
BlendStateDescription?, DepthStencilStateDescription?,
RasterizerStateDescription?, Color?)**

Begins a new rendering session with the specified command list and output description.

```
public void Begin(CommandList commandList, OutputDescription output, BlendStateDescription?  
blendState = null, DepthStencilStateDescription? depthStencilState = null,  
RasterizerStateDescription? rasterizerState = null, Color? color = null)
```

Parameters

commandList [CommandList](#)

The command list used for issuing rendering commands.

output [OutputDescription](#)

The output description for rendering.

blendState [BlendStateDescription](#)?

Optional blend state description. If null, a default is used.

depthStencilState [DepthStencilStateDescription](#)?

Optional depth-stencil state description. If null, a default is used.

rasterizerState [RasterizerStateDescription](#)?

Optional rasterizer state description. If null, a default is used.

color Color?

Optional color for rendering debug visuals. If null, white is used.

Exceptions

Exception ↗

Thrown if a rendering session has already begun.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing bool ↗

True if called from user code; false if called from a finalizer.

DrawPoint(in JVector)

Draws a point at the specified position.

```
public void DrawPoint(in JVector p)
```

Parameters

p JVector

The Jitter2.LinearMath.JVector representing the position of the point in 3D space.

DrawSegment(in JVector, in JVector)

Draws a line segment using the specified start and end points in 3D space.

```
public void DrawSegment(in JVector pA, in JVector pB)
```

Parameters

pA JVector

The Jitter2.LinearMath.JVector representing the starting point of the segment.

pB JVector

The Jitter2.LinearMath.JVector representing the ending point of the segment.

DrawTriangle(in JVector, in JVector, in JVector)

Draws a triangle using the specified vertices in 3D space.

```
public void DrawTriangle(in JVector pA, in JVector pB, in JVector pC)
```

Parameters

pA JVector

The Jitter2.LinearMath.JVector representing the first vertex of the triangle.

pB JVector

The Jitter2.LinearMath.JVector representing the second vertex of the triangle.

pC JVector

The Jitter2.LinearMath.JVector representing the third vertex of the triangle.

End()

Ends the current rendering session and flushes any remaining draw calls.

```
public void End()
```

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

GetCurrentBlendState()

Gets the current blend state description being used for rendering.

```
public BlendStateDescription GetCurrentBlendState()
```

Returns

[BlendStateDescription](#)

The current [BlendStateDescription](#).

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

GetCurrentColor()

Gets the current color used for rendering debug shapes.

```
public Color GetCurrentColor()
```

Returns

Color

The current color used for rendering.

Exceptions

[Exception](#)

Thrown if the drawer has not begun rendering yet.

GetCurrentDepthStencilState()

Gets the current depth-stencil state being used in rendering.

```
public DepthStencilStateDescription GetCurrentDepthStencilState()
```

Returns

[DepthStencilStateDescription](#)

The current depth-stencil state.

Exceptions

[Exception](#)

Thrown if the drawer has not begun rendering yet.

GetCurrentOutput()

Gets the current output description being used for rendering.

```
public OutputDescription GetCurrentOutput()
```

Returns

[OutputDescription](#)

The current [OutputDescription](#).

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

GetCurrentRasterizerState()

Gets the current rasterizer state being used in rendering.

```
public RasterizerStateDescription GetCurrentRasterizerState()
```

Returns

[RasterizerStateDescription](#)

The current rasterizer state.

Exceptions

[Exception](#)

Thrown if the drawer has not begun rendering yet.

PopBlendState()

Restores the blend state back to the one set at [Begin\(CommandList, OutputDescription, BlendStateDescription?, DepthStencilStateDescription?, RasterizerStateDescription?, Color?\)](#).

```
public void PopBlendState()
```

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

PopColor()

Restores the debug shape color back to the one set at [Begin\(CommandList, OutputDescription, BlendStateDescription?, DepthStencilStateDescription?, RasterizerStateDescription?, Color?\)](#).

```
public void PopColor()
```

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

PopDepthStencilState()

Restores the depth-stencil state back to the one set at [Begin\(CommandList, OutputDescription, BlendStateDescription?, DepthStencilStateDescription?, RasterizerStateDescription?, Color?\)](#).

```
public void PopDepthStencilState()
```

Exceptions

[Exception ↗](#)

Thrown if a rendering session has not begun.

PopOutput()

Restores the output description back to the one set at [Begin\(CommandList, OutputDescription, BlendStateDescription?, DepthStencilStateDescription?, RasterizerStateDescription?, Color?\)](#).

```
public void PopOutput()
```

Exceptions

[Exception ↗](#)

Thrown if a rendering session has not begun.

PopRasterizerState()

Restores the rasterizer state back to the one set at [Begin\(CommandList, OutputDescription, BlendStateDescription?, DepthStencilStateDescription?, RasterizerStateDescription?, Color?\)](#).

```
public void PopRasterizerState()
```

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

PushBlendState(BlendStateDescription)

Pushes a new blend state to override the current state.

```
public void PushBlendState(BlendStateDescription blendState)
```

Parameters

blendState [BlendStateDescription](#)

The new [BlendStateDescription](#) to apply.

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

PushColor(Color)

Pushes a new color to override the current debug shape color.

```
public void PushColor(Color color)
```

Parameters

color Color

The new Bliss.CSharp.Colors.Color to apply.

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

PushDepthStencilState(DepthStencilStateDescription)

Pushes a new depth-stencil state to override the current state.

```
public void PushDepthStencilState(DepthStencilStateDescription depthStencilState)
```

Parameters

`depthStencilState` [DepthStencilStateDescription](#)

The new [DepthStencilStateDescription](#) to apply.

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

PushOutput(OutputDescription)

Pushes a new output description to override the current rendering target.

```
public void PushOutput(OutputDescription output)
```

Parameters

`output` [OutputDescription](#)

The new [OutputDescription](#) to apply.

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

PushRasterizerState(RasterizerStateDescription)

Pushes a new rasterizer state to override the current state.

```
public void PushRasterizerState(RasterizerStateDescription rasterizerState)
```

Parameters

rasterizerState [RasterizerStateDescription](#)

The new [RasterizerStateDescription](#) to apply.

Exceptions

[Exception](#)

Thrown if a rendering session has not begun.

Class SkyBox

Namespace: [Sparkle.CSharp.Graphics.Rendering](#)

Assembly: Sparkle.dll

```
public class SkyBox : Disposable, IDisposable
```

Inheritance

[object](#) ← Disposable ← SkyBox

Implements

[IDisposable](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

SkyBox(GraphicsDevice, Cubemap, Sampler?, Color?)

Initializes a new instance of the [SkyBox](#) class.

```
public SkyBox(GraphicsDevice graphicsDevice, Cubemap cubemap, Sampler? sampler = null,  
Color? color = null)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for rendering.

cubemap Cubemap

The cubemap texture used for the skybox.

sampler [Sampler](#)

The optional texture sampler.

color Color?

The optional color tint applied to the skybox.

Fields

Color

The color tint applied to the skybox.

```
public Color Color
```

Field Value

Color

Cubemap

The cubemap texture used for the skybox.

```
public Cubemap Cubemap
```

Field Value

Cubemap

Sampler

The sampler used for texture sampling.

```
public Sampler Sampler
```

Field Value

[Sampler](#)

Properties

GraphicsDevice

The graphics device used for rendering.

```
public GraphicsDevice GraphicsDevice { get; }
```

Property Value

[GraphicsDevice](#)

Methods

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Namespace Sparkle.CSharp.Graphics. Rendering.Sprites

Classes

[SpriteRenderer](#)

Structs

[SpriteData](#)

Struct SpriteData

Namespace: [Sparkle.CSharp.Graphics.Rendering.Sprites](#)

Assembly: Sparkle.dll

```
public struct SpriteData
```

Inherited Members

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

Constructors

SpriteData(Texture2D, Sampler?, Vector2, float, Rectangle, Vector2, Vector2, float, Color, SpriteFlip, Effect?, BlendStateDescription?, DepthStencilStateDescription?, RasterizerStateDescription?, Rectangle?)

Initializes a new [SpriteData](#) instance with the specified rendering configuration.

```
public SpriteData(Texture2D texture, Sampler? sampler, Vector2 position, float layerDepth, Rectangle sourceRect, Vector2 scale, Vector2 origin, float rotation, Color color, SpriteFlip flip, Effect? effect, BlendStateDescription? blendState, DepthStencilStateDescription? depthStencilState, RasterizerStateDescription? rasterizerState, Rectangle? scissorRect)
```

Parameters

texture Texture2D

The texture to draw.

sampler [Sampler](#)

Optional sampler for texture sampling.

position [Vector2](#)

The world position where the sprite will be rendered.

layerDepth [float](#)?

The Z-order value controlling render order.

sourceRect Rectangle

The portion of the texture to draw.

scale [Vector2](#)?

The scale factor for the sprite.

origin [Vector2](#)?

The pivot point for rotation and scaling.

rotation [float](#)?

The rotation angle in radians.

color Color

A color tint to blend with the sprite texture.

flip SpriteFlip

The flip mode to apply when rendering the sprite.

effect Effect

Optional rendering effect (shader).

blendState [BlendStateDescription](#)?

Optional custom blend state.

depthStencilState [DepthStencilStateDescription](#)?

Optional depth-stencil state.

rasterizerState [RasterizerStateDescription](#)?

Optional rasterizer state.

scissorRect Rectangle?

Optional scissor rectangle to define a restricted rendering area.

Fields

BlendState

Optional blend state override used for this sprite.

```
public BlendStateDescription? BlendState
```

Field Value

[BlendStateDescription](#)?

Color

The color tint to apply to the sprite.

```
public Color Color
```

Field Value

Color

DepthStencilState

Optional depth-stencil state override used for this sprite.

```
public DepthStencilStateDescription? DepthStencilState
```

Field Value

[DepthStencilStateDescription](#)?

Effect

Optional custom effect (shader) applied when rendering this sprite.

```
public Effect? Effect
```

Field Value

Effect

Flip

The sprite flip mode.

```
public SpriteFlip Flip
```

Field Value

SpriteFlip

LayerDepth

The layer depth used to determine draw order.

```
public float LayerDepth
```

Field Value

[float](#)

Origin

The origin point for scaling and rotation, typically the center or a corner of the sprite.

```
public Vector2 Origin
```

Field Value

[Vector2](#)

Position

The world-space position where the sprite should be rendered.

```
public Vector2 Position
```

Field Value

[Vector2](#)

RasterizerState

Optional rasterizer state override used for this sprite.

```
public RasterizerStateDescription? RasterizerState
```

Field Value

[RasterizerStateDescription](#)?

Rotation

The rotation to apply when rendering the sprite.

```
public float Rotation
```

Field Value

[float](#)

Sampler

Optional sampler to control how the texture is sampled.

```
public Sampler? Sampler
```

Field Value

[Sampler](#) ↗

Scale

The scale factor to apply when rendering the sprite.

```
public Vector2 Scale
```

Field Value

[Vector2](#) ↗

ScissorRect

Optional scissor rectangle to define a restricted rendering area for this sprite.

```
public Rectangle? ScissorRect
```

Field Value

Rectangle?

SourceRect

The source rectangle defining the portion of the texture to use.

```
public Rectangle SourceRect
```

Field Value

Rectangle

Texture

The texture to render for this sprite.

```
public Texture2D Texture
```

Field Value

Texture2D

Class SpriteRenderer

Namespace: [Sparkle.CSharp.Graphics.Rendering.Sprites](#)

Assembly: Sparkle.dll

```
public class SpriteRenderer
```

Inheritance

[object](#) ← SpriteRenderer

Inherited Members

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

Constructors

SpriteRenderer()

Initializes a new instance of the [SpriteRenderer](#) class.

```
public SpriteRenderer()
```

Methods

Draw(GraphicsContext, Framebuffer)

Draws all queued sprites onto the specified framebuffer using the provided graphics context.

```
protected void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering operations.

framebuffer [Framebuffer](#)

The framebuffer where the sprites will be drawn.

DrawSprite(Texture2D, Sampler?, Vector2, float, Rectangle, Vector2, Vector2, float, Color, SpriteFlip, Effect?, BlendStateDescription?, DepthStencilStateDescription?, RasterizerStateDescription?, Rectangle?)

Queues a sprite for rendering with specified properties.

```
public void DrawSprite(Texture2D texture, Sampler? sampler, Vector2 position, float layerDepth, Rectangle sourceRect, Vector2 scale, Vector2 origin, float rotation, Color color, SpriteFlip flip, Effect? effect, BlendStateDescription? blendState, DepthStencilStateDescription? depthStencilState, RasterizerStateDescription? rasterizerState, Rectangle? scissorRect)
```

Parameters

texture Texture2D

The texture of the sprite to be drawn.

sampler [Sampler](#)

The sampler specifying how the texture will be sampled.

position [Vector2](#)

The position where the sprite will be rendered.

layerDepth [float](#)

The depth value for layer sorting of the sprite.

sourceRect Rectangle

The rectangle defining the portion of the texture to use.

scale [Vector2](#)

The scaling factor applied to the sprite.

origin [Vector2](#)?

The origin point used for rotation and scaling transformations.

rotation float?

The rotation angle in radians applied to the sprite.

color Color

The color applied to blend with the sprite texture.

flip SpriteFlip

The sprite flipping mode indicating horizontal or vertical flipping.

effect Effect

The effect to be applied to the sprite during rendering, if any.

blendState [BlendStateDescription](#)?

The blend state to be used for rendering, or null for the default state.

depthStencilState [DepthStencilStateDescription](#)?

The depth-stencil state to be used for rendering, or null for the default state.

rasterizerState [RasterizerStateDescription](#)?

The rasterizer state to be used for rendering, or null for the default state.

scissorRect Rectangle?

The rectangle used to define the scissor test area for rendering, or null for none.

Namespace Sparkle.CSharp.Graphics.Vertex Types

Structs

[PhysicsDebugVertex3D](#)

Struct PhysicsDebugVertex3D

Namespace: [Sparkle.CSharp.Graphics.VertexTypes](#)

Assembly: Sparkle.dll

```
public struct PhysicsDebugVertex3D
```

Inherited Members

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

Constructors

PhysicsDebugVertex3D(Vector3, Vector4)

Initializes a new instance of the [PhysicsDebugVertex3D](#) struct with the specified position and color.

```
public PhysicsDebugVertex3D(Vector3 position, Vector4 color)
```

Parameters

position [Vector3](#)

The position of the vertex in 3D space.

color [Vector4](#)

The color of the vertex, including alpha transparency.

Fields

Color

The color of the vertex, including alpha transparency.

```
public Vector4 Color
```

Field Value

[Vector4](#) ↗

Position

The position of the vertex in 3D space.

```
public Vector3 Position
```

Field Value

[Vector3](#) ↗

VertexLayout

The layout description used by the graphics pipeline to interpret this vertex's data.

```
public static VertexLayoutDescription VertexLayout
```

Field Value

[VertexLayoutDescription](#) ↗

Namespace Sparkle.CSharp.IO

Classes

[CryptoProvider](#)

[FileAccessor](#)

Class CryptoProvider

Namespace: [Sparkle.CSharp.IO](#)

Assembly: Sparkle.dll

```
public static class CryptoProvider
```

Inheritance

[object](#) ← CryptoProvider

Inherited Members

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

Methods

Decrypt(string, string)

Decrypts a Base64-encoded string using AES decryption with the provided key.

```
public static string Decrypt(string text, string key)
```

Parameters

text [string](#)

The encrypted text to be decrypted.

key [string](#)

The decryption key.

Returns

[string](#)

The decrypted text.

Encrypt(string, string)

Encrypts a string using AES encryption with the provided key.

```
public static string Encrypt(string text, string key)
```

Parameters

text [string](#)

The text to be encrypted.

key [string](#)

The encryption key.

Returns

[string](#)

The encrypted text as a Base64-encoded string.

Class FileAccessor

Namespace: [Sparkle.CSharp.IO](#)

Assembly: Sparkle.dll

```
public static class FileAccessor
```

Inheritance

[object](#) ← FileAccessor

Inherited Members

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

Methods

Clear(string)

Clears the content of the file at the specified path.

```
public static void Clear(string path)
```

Parameters

path [string](#)

The full path to the file.

Create(string, string, bool)

Creates a new file with the specified name and directory.

```
public static void Create(string directory, string name, bool replace = false)
```

Parameters

directory [string](#)

The directory where the file should be created.

name [string](#)

The name of the file to be created.

replace [bool](#)

Specifies whether to replace an existing file with the same name in the directory. Default is false.

GetPath(string, string)

Combines the specified file name and directory path to create a full file path.

```
public static string GetPath(string directory, string name)
```

Parameters

directory [string](#)

The directory where the file is located.

name [string](#)

The name of the file.

Returns

[string](#)

The full path to the file.

Read(string)

Reads all text from the file at the specified path.

```
public static string Read(string path)
```

Parameters

path [string](#)

The full path to the file.

Returns

[string](#)

A string containing the text read from the file.

ReadAll(string)

Reads all text from the file at the specified path.

```
public static string ReadAll(string path)
```

Parameters

path [string](#)

The full path to the file.

Returns

[string](#)

The text content of the file.

ReadAllLines(string)

Reads all lines from the file at the specified path.

```
public static string[] ReadAllLines(string path)
```

Parameters

path [string](#)

The full path to the file.

Returns

[string](#)[]

An array of strings containing all the lines from the file.

ReadLine(string, int)

Retrieves a specific line of text from a file at the given path.

```
public static string ReadLine(string path, int index)
```

Parameters

path [string](#)

The full path to the file.

index [int](#)

The index of the line to retrieve.

Returns

[string](#)

The specified line of text from the file.

Write(string, string)

Writes the specified text to the file at the given path.

```
public static void Write(string path, string text)
```

Parameters

path [string](#)

The full path to the file.

text [string](#)

The text to be written to the file.

WriteAll(string, string)

Writes the specified text to the file at the given path, replacing any existing content.

```
public static void WriteAll(string path, string text)
```

Parameters

path [string](#)

The full path to the file.

text [string](#)

The text to be written to the file.

WriteAllLines(string, string[])

Writes all lines of text to the file at the specified path.

```
public static void WriteAllLines(string path, string[] text)
```

Parameters

path [string](#)

The full path to the file.

text [string](#)[]

An array of strings containing the lines of text to write to the file.

WriteLine(string, string)

Writes the specified text followed by a new line to the file at the given path.

```
public static void WriteLine(string path, string text)
```

Parameters

path [string](#)

The full path to the file.

text [string](#)

The text to be written to the file.

Namespace Sparkle.CSharp.IO.Configs.Json

Classes

[JsonConfig](#)

[JsonConfigBuilder](#)

Class JsonConfig

Namespace: [Sparkle.CSharp.IO.Configs.Json](#)

Assembly: Sparkle.dll

```
public class JsonConfig
```

Inheritance

[object](#) ← JsonConfig

Inherited Members

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

Constructors

JsonConfig(string, string, Dictionary<string, object>, string)

Represents a JSON configuration file.

```
public JsonConfig(string directory, string name, Dictionary<string, object> defaultValues,  
string encryptKey = "")
```

Parameters

directory [string](#)

name [string](#)

defaultValues [Dictionary](#)<[string](#), [object](#)>

encryptKey [string](#)

Properties

Directory

Gets the directory where the JSON configuration file is stored.

```
public string Directory { get; }
```

Property Value

[string](#)

Name

Gets the name of the JSON configuration file, excluding the extension.

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

Property Value

[string](#)

Path

Gets the full file path of the JSON configuration file.

```
public string Path { get; }
```

Property Value

[string](#)

Methods

AddValue<T>(string, T)

Adds a value to the JSON configuration file.

```
public void AddValue<T>(string key, T value)
```

Parameters

key [string](#) ↗

The key associated with the value.

value [T](#)

The value to be added.

Type Parameters

[T](#)

The type of the value.

GetAllValues()

Retrieves all values from the JSON configuration file.

```
public JObject GetAllValues()
```

Returns

JObject

A JObject containing all the values from the JSON configuration file.

Exceptions

[Exception](#) ↗

Thrown if unable to retrieve the values from the file.

GetValue<T>(string)

Retrieves the value associated with the specified key from the JSON configuration file.

```
public T? GetValue<T>(string key)
```

Parameters

key [string](#) ↗

The key of the value to retrieve.

Returns

T

The value associated with the specified key, or default(T) if the key is not found.

Type Parameters

T

The type of the value to retrieve.

IsValid()

Checks if the JSON configuration file is valid.

```
public bool IsValid()
```

Returns

[bool](#) ↗

True if the JSON configuration file is valid, otherwise false.

RemoveValue(string)

Removes a value from the JSON configuration file.

```
public void RemoveValue(string key)
```

Parameters

key [string](#) ↗

The key of the value to be removed.

SetValue<T>(string, T)

Sets the value of a key in the JSON configuration file.

```
public void SetValue<T>(string key, T value)
```

Parameters

key [string](#)

The key associated with the value.

value T

The new value to set.

Type Parameters

T

The type of the value.

Class JsonConfigBuilder

Namespace: [Sparkle.CSharp.IO.Configs.Json](#)

Assembly: Sparkle.dll

```
public class JsonConfigBuilder
```

Inheritance

[object](#) ← JsonConfigBuilder

Inherited Members

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

Constructors

JsonConfigBuilder(string, string, string)

Represents a builder for creating JSON configurations.

```
public JsonConfigBuilder(string directory, string name, string encryptKey = "")
```

Parameters

directory [string](#)

name [string](#)

encryptKey [string](#)

Methods

Add<T>(string, T)

Adds a key-value pair to the JSON configuration builder.

```
public JsonConfigBuilder Add<T>(string key, T value)
```

Parameters

key [string](#) ↗

The key of the value.

value [T](#)

The value to be added.

Returns

[JsonConfigBuilder](#)

The updated JSON configuration builder.

Type Parameters

[T](#)

The type of the value.

Build()

Builds a JSON configuration file using the values added to the builder.

```
public JsonConfig Build()
```

Returns

[JsonConfig](#)

The created JSON configuration file.

Namespace Sparkle.CSharp.Logging

Classes

[LogFileWriter](#)

[LogJitter](#)

Class LogFileWriter

Namespace: [Sparkle.CSharp.Logging](#)

Assembly: Sparkle.dll

```
public class LogFileWriter
```

Inheritance

[object](#) ← LogFileWriter

Inherited Members

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

Constructors

LogFileWriter(string)

Initializes a new instance of the [LogFileWriter](#) class, creating a new log file in the specified directory.

```
public LogFileWriter(string directory)
```

Parameters

directory [string](#)

The directory where log files should be saved.

Properties

FilePath

The full path of the log file where messages are written.

```
public string FilePath { get; }
```

Property Value

[string](#)

Methods

WriteFileMsg(LogType, string, ConsoleColor, string, string, int)

Writes a log message to the log file with the specified log type, message, color, and source information.

```
public bool WriteFileMsg(LogType type, string msg, ConsoleColor color, string  
sourceFilePath, string memberName, int sourceLineNumber)
```

Parameters

type LogType

The type of the log (e.g., Debug, Info, Warn, Error, Fatal).

msg [string](#)

The log message to write.

color [ConsoleColor](#)

The console color associated with the message.

sourceFilePath [string](#)

The file path of the source code that emitted the message.

memberName [string](#)

The name of the method or member that emitted the message.

sourceLineNumber [int](#)

The line number in the source where the message originated.

Returns

[bool](#)

Returns **true** if the log message was successfully written, otherwise **false**.

Class LogJitter

Namespace: [Sparkle.CSharp.Logging](#)

Assembly: Sparkle.dll

```
public class LogJitter
```

Inheritance

[object](#) ← LogJitter

Inherited Members

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

Methods

Log(LogLevel, string)

Logs a message with the specified log level.

```
public void Log(Logger.LogLevel level, string msg)
```

Parameters

level Logger.LogLevel

The log level used to classify the log message (Information, Warning, or Error).

msg [string](#)

The message to log.

Namespace Sparkle.CSharp.Overlays

Classes

[Overlay](#)

[OverlayManager](#)

Class Overlay

Namespace: [Sparkle.CSharp.Overlays](#)

Assembly: Sparkle.dll

```
public abstract class Overlay
```

Inheritance

[object](#) ← Overlay

Derived

[TestOverlay](#)

Inherited Members

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

Constructors

Overlay(string, bool)

Initializes a new instance of the [Overlay](#) class.

```
protected Overlay(string name, bool enabled = false)
```

Parameters

name [string](#)

The name of the overlay.

enabled [bool](#)

Whether the overlay is initially enabled (default is false).

Fields

Enabled

Determines whether the overlay is enabled.

```
public bool Enabled
```

Field Value

[bool](#)

Properties

Name

Gets the name of the overlay.

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

Property Value

[string](#)

Methods

AfterUpdate(double)

Executes logic after the update step.

```
protected virtual void AfterUpdate(double delta)
```

Parameters

[delta](#) [double](#)

The time delta since the last update.

Draw(GraphicsContext, Framebuffer)

Draws the overlay.

```
protected abstract void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering.

framebuffer [Framebuffer](#)

The framebuffer to which the overlay is rendered.

FixedUpdate(double)

Executes fixed-step updates for the overlay.

```
protected virtual void FixedUpdate(double timeStep)
```

Parameters

timeStep [double](#)

The fixed time step interval for the update.

Resize(Rectangle)

Executes when the window is resized.

```
protected virtual void Resize(Rectangle rectangle)
```

Parameters

rectangle Rectangle

The rectangle specifying the window's updated size.

Update(double)

Updates the overlay each frame.

```
protected virtual void Update(double delta)
```

Parameters

delta [double](#)

The time delta since the last update.

Class OverlayManager

Namespace: [Sparkle.CSharp.Overlays](#)

Assembly: Sparkle.dll

```
public static class OverlayManager
```

Inheritance

[object](#) ← OverlayManager

Inherited Members

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

Methods

AddOverlay(Overlay)

Adds an overlay to the manager.

```
public static void AddOverlay(Overlay overlay)
```

Parameters

overlay [Overlay](#)

The overlay to add.

GetOverlays()

Retrieves the list of active overlays managed by the [OverlayManager](#).

```
public static IEnumerable<Overlay> GetOverlays()
```

Returns

[IEnumerable](#) <[Overlay](#)>

A collection of overlays currently managed by the [OverlayManager](#).

RemoveOverlay(Overlay)

Removes an overlay from the manager.

```
public static void RemoveOverlay(Overlay overlay)
```

Parameters

`overlay` [Overlay](#)

The overlay to remove.

Namespace Sparkle.CSharp.Physics

Classes

[Simulation](#)

Class Simulation

Namespace: [Sparkle.CSharp.Physics](#)

Assembly: Sparkle.dll

```
public abstract class Simulation : Disposable, IDisposable
```

Inheritance

[object](#) ← Disposable ← Simulation

Implements

[IDisposable](#)

Derived

[Simulation2D](#), [Simulation3D](#)

Inherited Members

Disposable.Dispose() , [Disposable.Dispose\(bool\)](#) , Disposable.ThrowIfDisposed() ,
Disposable.HasDisposed , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Step(double)

Executes a simulation step in the physics engine with a fixed time step.

```
protected abstract void Step(double fixedStep)
```

Parameters

fixedStep [double](#)

The fixed time interval for the simulation step.

Namespace Sparkle.CSharp.Physics.Dim2

Classes

[Simulation2D](#)

Structs

[PhysicsSettings2D](#)

Struct PhysicsSettings2D

Namespace: [Sparkle.CSharp.Physics.Dim2](#)

Assembly: Sparkle.dll

```
public struct PhysicsSettings2D
```

Inherited Members

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

Constructors

PhysicsSettings2D()

Initializes a new instance of the [PhysicsSettings2D](#) struct with default values.

```
public PhysicsSettings2D()
```

Fields

SubStepCount

The number of substeps used for physics simulations in a single timestep.

```
public int SubStepCount
```

Field Value

[int](#)

WorldDef

The definition of the physics world, including its properties and settings.

```
public WorldDef WorldDef
```

Field Value

WorldDef

Class Simulation2D

Namespace: [Sparkle.CSharp.Physics.Dim2](#)

Assembly: Sparkle.dll

```
public class Simulation2D : Simulation, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Simulation](#) ← Simulation2D

Implements

[IDisposable](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

Simulation2D(PhysicsSettings2D)

Constructor for creating a Simulation2D object.

```
public Simulation2D(PhysicsSettings2D settings)
```

Parameters

settings [PhysicsSettings2D](#)

The physics settings for the 2D simulation.

Fields

World

The world in which the 2D physics simulation takes place.

```
public readonly World World
```

Field Value

World

Methods

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

`disposing` [bool](#)

True if called from user code; false if called from a finalizer.

Step(double)

Advances the simulation by a fixed time step.

```
protected override void Step(double fixedStep)
```

Parameters

`fixedStep` [double](#)

The time step duration in seconds to advance the simulation.

Events

BodyMoved

Triggered when a body has moved during the physics simulation step.

```
public event Action<BodyMoveEvent>? BodyMoved
```

Event Type

[Action](#) <BodyMoveEvent>

ContactBeginTouch

Triggered when contact between two objects begins.

```
public event Action<ContactBeginTouchEvent>? ContactBeginTouch
```

Event Type

[Action](#) <ContactBeginTouchEvent>

ContactEndTouch

Triggered when contact between two objects ends.

```
public event Action<ContactEndTouchEvent>? ContactEndTouch
```

Event Type

[Action](#) <ContactEndTouchEvent>

ContactHit

Triggered when a contact hit occurs between two colliding bodies.

```
public event Action<ContactHitEvent>? ContactHit
```

Event Type

[Action](#) <ContactHitEvent>

SensorBeginTouch

Triggered when a sensor begins touching another object.

```
public event Action<SensorBeginTouchEvent>? SensorBeginTouch
```

Event Type

[Action](#) <SensorBeginTouchEvent>

SensorEndTouch

Triggered when a sensor stops touching another object.

```
public event Action<SensorEndTouchEvent>? SensorEndTouch
```

Event Type

[Action](#) <SensorEndTouchEvent>

Namespace Sparkle.CSharp.Physics.Dim2.Def

Structs

[BodyDefinition](#)

Struct BodyDefinition

Namespace: [Sparkle.CSharp.Physics.Dim2.Def](#)

Assembly: Sparkle.dll

```
public struct BodyDefinition
```

Inherited Members

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

Constructors

BodyDefinition()

Initializes a new instance of the [BodyDefinition](#) struct.

```
public BodyDefinition()
```

Fields

AllowFastRotation

Whether fast rotation is allowed, possibly sacrificing stability for performance.

```
public bool AllowFastRotation
```

Field Value

[bool](#)

AngularDamping

The angular damping factor, which reduces the angular velocity over time.

```
public float AngularDamping
```

Field Value

[float](#) ↗

AngularVelocity

The initial angular velocity of the body in radians per second.

```
public float AngularVelocity
```

Field Value

[float](#) ↗

EnableSleep

Whether the body is allowed to sleep to conserve resources.

```
public bool EnableSleep
```

Field Value

[bool](#) ↗

FixedRotation

Whether the body should have a fixed rotation, preventing it from rotating.

```
public bool FixedRotation
```

Field Value

[bool](#) ↗

GravityScale

Scale factor for the gravity applied to this body.

```
public float GravityScale
```

Field Value

[float](#) ↗

IsAwake

Whether the body should start in the awake state.

```
public bool IsAwake
```

Field Value

[bool](#) ↗

IsBullet

Whether the body is a bullet (i.e., uses continuous collision detection).

```
public bool IsBullet
```

Field Value

[bool](#) ↗

IsEnabled

Whether the body is enabled and participates in simulation and collisions.

```
public bool IsEnabled
```

Field Value

[bool](#) ↗

LinearDamping

The linear damping factor, which reduces the linear velocity over time.

`public float LinearDamping`

Field Value

[float](#) ↗

LinearVelocity

The initial linear velocity of the body in world units per second.

`public Vector2 LinearVelocity`

Field Value

[Vector2](#) ↗

Name

An optional name identifier for the body.

`public string? Name`

Field Value

[string](#) ↗

SleepThreshold

The minimum speed under which the body can enter the sleep state.

```
public float SleepThreshold
```

Field Value

[float](#)

Type

The type of the body (e.g., Static, Dynamic, or Kinematic).

```
public BodyType Type
```

Field Value

BodyType

UserData

Optional user-defined data associated with the body.

```
public object? UserData
```

Field Value

[object](#)

Namespace Sparkle.CSharp.Physics.Dim2.Shapes

Classes

[CapsuleShape2D](#)

[ChainShape2D](#)

[CircleShape2D](#)

[PolygonShape2D](#)

[SegmentShape2D](#)

Interfaces

[IShape2D](#)

Class CapsuleShape2D

Namespace: [Sparkle.CSharp.Physics.Dim2.Shapes](#)

Assembly: Sparkle.dll

```
public class CapsuleShape2D : IShape2D
```

Inheritance

[object](#) ← CapsuleShape2D

Implements

[IShape2D](#)

Inherited Members

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

Constructors

CapsuleShape2D(Capsule, ShapeDef)

Initializes a new instance of the [CapsuleShape2D](#) class.

```
public CapsuleShape2D(Capsule capsule, ShapeDef def)
```

Parameters

capsule Capsule

The capsule shape data.

def ShapeDef

The definition for the shape's physical properties.

Methods

CreateShape(Body)

Creates the capsule shape and attaches it to the specified physics body.

```
public void CreateShape(Body body)
```

Parameters

body Body

The body to attach the shape to.

Class ChainShape2D

Namespace: [Sparkle.CSharp.Physics.Dim2.Shapes](#)

Assembly: Sparkle.dll

```
public class ChainShape2D : IShape2D
```

Inheritance

[object](#) ← ChainShape2D

Implements

[IShape2D](#)

Inherited Members

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

Constructors

ChainShape2D(ChainDef)

Initializes a new instance of the [ChainShape2D](#) class.

```
public ChainShape2D(ChainDef def)
```

Parameters

def ChainDef

The definition for the chain shape's structure and behavior.

Methods

CreateShape(Body)

Creates the chain shape and attaches it to the specified physics body.

```
public void CreateShape(Body body)
```

Parameters

body Body

The body to which the chain shape will be attached.

Class CircleShape2D

Namespace: [Sparkle.CSharp.Physics.Dim2.Shapes](#)

Assembly: Sparkle.dll

```
public class CircleShape2D : IShape2D
```

Inheritance

[object](#) ← CircleShape2D

Implements

[IShape2D](#)

Inherited Members

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

Constructors

CircleShape2D(Circle, ShapeDef)

Initializes a new instance of the [CircleShape2D](#) class.

```
public CircleShape2D(Circle circle, ShapeDef def)
```

Parameters

circle Circle

The circle geometry defining the shape's radius and center.

def ShapeDef

The definition of the shape's physical properties.

Methods

CreateShape(Body)

Creates the circle shape and attaches it to the specified physics body.

```
public void CreateShape(Body body)
```

Parameters

body Body

The body to which the circle shape will be attached.

Interface IShape2D

Namespace: [Sparkle.CSharp.Physics.Dim2.Shapes](#)

Assembly: Sparkle.dll

```
public interface IShape2D
```

Methods

CreateShape(Body)

Creates and attaches a shape to the specified physics body.

```
void CreateShape(Body body)
```

Parameters

body Body

The physics body to which the shape will be attached.

Class PolygonShape2D

Namespace: [Sparkle.CSharp.Physics.Dim2.Shapes](#)

Assembly: Sparkle.dll

```
public class PolygonShape2D : IShape2D
```

Inheritance

[object](#) ← PolygonShape2D

Implements

[IShape2D](#)

Inherited Members

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

Constructors

PolygonShape2D(Polygon, ShapeDef)

Initializes a new instance of the [PolygonShape2D](#) class.

```
public PolygonShape2D(Polygon polygon, ShapeDef def)
```

Parameters

polygon Polygon

The polygon geometry defining the shape's structure.

def ShapeDef

The definition of the shape's physical properties.

Methods

CreateShape(Body)

Creates the polygon shape and attaches it to the specified physics body.

```
public void CreateShape(Body body)
```

Parameters

body Body

The body to which the polygon shape will be attached.

Class SegmentShape2D

Namespace: [Sparkle.CSharp.Physics.Dim2.Shapes](#)

Assembly: Sparkle.dll

```
public class SegmentShape2D : IShape2D
```

Inheritance

[object](#) ← SegmentShape2D

Implements

[IShape2D](#)

Inherited Members

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

Constructors

SegmentShape2D(Segment, ShapeDef)

Initializes a new instance of the [SegmentShape2D](#) class.

```
public SegmentShape2D(Segment segment, ShapeDef def)
```

Parameters

segment Segment

The segment geometry defining the shape's structure.

def ShapeDef

The definition of the shape's physical properties.

Methods

CreateShape(Body)

Creates the segment shape and attaches it to the specified physics body.

```
public void CreateShape(Body body)
```

Parameters

body Body

The body to which the segment shape will be attached.

Namespace Sparkle.CSharp.Physics.Dim3

Classes

[Simulation3D](#)

Structs

[PhysicsSettings3D](#)

Struct PhysicsSettings3D

Namespace: [Sparkle.CSharp.Physics.Dim3](#)

Assembly: Sparkle.dll

```
public struct PhysicsSettings3D
```

Inherited Members

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

Constructors

PhysicsSettings3D()

Represents the settings for the physics engine.

```
public PhysicsSettings3D()
```

Fields

Gravity

The gravitational force applied in the simulation. Default is (0, -9.81, 0) representing Earth's gravity.

```
public Vector3 Gravity
```

Field Value

[Vector3](#)

MultiThreaded

Specifies whether the physics simulation runs using multiple threads.

```
public bool MultiThreaded
```

Field Value

[bool ↗](#)

Class Simulation3D

Namespace: [Sparkle.CSharp.Physics.Dim3](#)

Assembly: Sparkle.dll

```
public class Simulation3D : Simulation, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Simulation](#) ← Simulation3D

Implements

[IDisposable](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

Simulation3D(PhysicsSettings3D)

Constructor for creating a Simulation3D object.

```
public Simulation3D(PhysicsSettings3D settings)
```

Parameters

settings [PhysicsSettings3D](#)

The physics settings for the 3D simulation.

Fields

World

The physics world that manages all physical objects and interactions.

```
public readonly World World
```

Field Value

World

Methods

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

`disposing` [bool](#)

True if called from user code; false if called from a finalizer.

Step(double)

Executes a simulation step for the 3D physics world.

```
protected override void Step(double fixedStep)
```

Parameters

`fixedStep` [double](#)

The fixed time interval for the simulation step.

Events

BodyMoved

Triggered when a body has moved during the physics simulation step.

```
public event Action<RigidBody>? BodyMoved
```

Event Type

[Action](#) <RigidBody>

Namespace Sparkle.CSharp.Physics.Dim3.Extensions

Classes

[Matrix4X4Extensions](#)

Class Matrix4X4Extensions

Namespace: [Sparkle.CSharp.Physics.Dim3.Extensions](#)

Assembly: Sparkle.dll

```
public static class Matrix4X4Extensions
```

Inheritance

[object](#) ← Matrix4X4Extensions

Inherited Members

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

Methods

ToJMatrix(Matrix4x4)

Converts a [Matrix4x4](#) instance to a Jitter2.LinearMath.JMatrix.

```
public static JMatrix ToJMatrix(this Matrix4x4 matrix)
```

Parameters

matrix [Matrix4x4](#)

The Matrix4x4 instance to convert.

Returns

JMatrix

A JMatrix containing the equivalent transformation values from the given Matrix4x4.

ToMatrix4X4(JMatrix)

Converts a Jitter2.LinearMath.JMatrix object to a [Matrix4x4](#) object.

```
public static Matrix4x4 ToMatrix4X4(this JMatrix jMatrix)
```

Parameters

jMatrix JMatrix

The JMatrix object to convert.

Returns

[Matrix4x4](#)

A Matrix4x4 object representing the same transformation as the input JMatrix.

Namespace Sparkle.CSharp.Physics.Dim3. Mappables

Classes

[UnitSphere](#)

Class UnitSphere

Namespace: [Sparkle.CSharp.Physics.Dim3.Mappables](#)

Assembly: Sparkle.dll

```
public class UnitSphere : ISupportMappable
```

Inheritance

[object](#) ← UnitSphere

Implements

ISupportMappable

Inherited Members

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

Methods

GetCenter(out JVector)

Gets the center point of the unit sphere.

```
public void GetCenter(out JVector point)
```

Parameters

point JVector

The output parameter that will store the center point of the unit sphere.

SupportMap(in JVector, out JVector)

Provides the support mapping for the unit sphere in a specified direction.

```
public void SupportMap(in JVector direction, out JVector result)
```

Parameters

direction JVector

The direction vector used for the support mapping calculation.

result JVector

The output parameter that will store the support point on the unit sphere.

Namespace Sparkle.CSharp.Physics.Dim3.SoftBodies.Factories

Classes

[SoftBodyClothFactory](#)

[SoftBodyCubeFactory](#)

[SoftBodySphereFactory](#)

Interfaces

[ISoftBodyFactory](#)

Interface ISoftBodyFactory

Namespace: [Sparkle.CSharp.Physics.Dim3.SoftBodies.Factories](#)

Assembly: Sparkle.dll

```
public interface ISoftBodyFactory
```

Methods

CreateSoftBody(GraphicsDevice, World, Vector3, Quaternion)

Creates a soft body instance within the specified world.

```
SimpleSoftBody CreateSoftBody(GraphicsDevice graphicsDevice, World world, Vector3 position,  
Quaternion rotation)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used to create material and meshes for the soft body.

world [World](#)

The physics world where the soft body will be created.

position [Vector3](#)

The initial position for the soft body.

rotation [Quaternion](#)

The initial rotation for the soft body.

Returns

[SimpleSoftBody](#)

A new instance of a soft body.

Class SoftBodyClothFactory

Namespace: [Sparkle.CSharp.Physics.Dim3.SoftBodies.Factories](#)

Assembly: Sparkle.dll

```
public class SoftBodyClothFactory : ISoftBodyFactory
```

Inheritance

[object](#) ← SoftBodyClothFactory

Implements

[ISoftBodyFactory](#)

Inherited Members

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

Constructors

SoftBodyClothFactory(int, int, Vector2, float, float, float, float, bool, bool)

Initializes a new instance of the [SoftBodyClothFactory](#) class with the specified cloth simulation parameters.

```
public SoftBodyClothFactory(int girdWidth, int girdHeight, Vector2 vertexSpacing, float  
vertexMass = 10, float centerMass = 1, float centerInertia = 0.05, float softness = 0.2,  
bool useDynamicCenterVertexSelection = true, bool useGridTexture = false)
```

Parameters

girdWidth [int](#)

The number of vertices along the width of the cloth grid.

girdHeight [int](#)

The number of vertices along the height of the cloth grid.

vertexSpacing [Vector2](#)

The spacing between adjacent vertices in the cloth grid.

vertexMass [float](#)

The mass of each vertex in the cloth grid.

centerMass [float](#)

The mass of the center vertex (used to stabilize the cloth).

centerInertia [float](#)

The inertia of the center vertex.

softness [float](#)

A factor that determines the elasticity or softness of the cloth.

useDynamicCenterVertexSelection [bool](#)

Determines whether the center vertex should be dynamically selected based on vertex positions.

useGridTexture [bool](#)

Whether to use a tiled grid texture layout for UV coordinates.

Properties

CenterInertia

The inertia of the center of the cloth.

```
public float CenterInertia { get; }
```

Property Value

[float](#)

CenterMass

The mass of the center of the cloth.

```
public float CenterMass { get; }
```

Property Value

[float ↗](#)

GirdHeight

The height of the cloth grid.

```
public int GirdHeight { get; }
```

Property Value

[int ↗](#)

GirdWidth

The width of the cloth grid.

```
public int GirdWidth { get; }
```

Property Value

[int ↗](#)

Softness

The softness of the cloth simulation.

```
public float Softness { get; }
```

Property Value

[float](#)

UseDynamicCenterVertexSelection

Indicates whether to dynamically select the center vertex for the cloth simulation.

```
public bool UseDynamicCenterVertexSelection { get; }
```

Property Value

[bool](#)

UseGridTexture

Indicates whether to use a tiled grid texture layout for UV coordinates.

```
public bool UseGridTexture { get; }
```

Property Value

[bool](#)

VertexMass

The mass of each vertex in the cloth simulation.

```
public float VertexMass { get; }
```

Property Value

[float](#)

VertexSpacing

The vertex spacing of the cloth grid.

```
public Vector2 VertexSpacing { get; }
```

Property Value

[Vector2](#)

Methods

CreateSoftBody(GraphicsDevice, World, Vector3, Quaternion)

Creates a new soft body cloth simulation in the given world with specified parameters.

```
public SimpleSoftBody CreateSoftBody(GraphicsDevice graphicsDevice, World world, Vector3  
position, Quaternion rotation)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for rendering.

world [World](#)

The simulation world where the soft body will be created.

position [Vector3](#)

The position where the cloth will be placed.

rotation [Quaternion](#)

The rotation of the cloth in the world.

Returns

[SimpleSoftBody](#)

A new instance of [SimpleSoftBody](#) representing the soft body cloth.

Class SoftBodyCubeFactory

Namespace: [Sparkle.CSharp.Physics.Dim3.SoftBodies.Factories](#)

Assembly: Sparkle.dll

```
public class SoftBodyCubeFactory : ISoftBodyFactory
```

Inheritance

[object](#) ← SoftBodyCubeFactory

Implements

[ISoftBodyFactory](#)

Inherited Members

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

Constructors

SoftBodyCubeFactory(Vector3, Vector3?, float, float, float, float)

Initializes a new instance of the [SoftBodyCubeFactory](#) class with the given physical parameters.

```
public SoftBodyCubeFactory(Vector3 size, Vector3? scale = null, float vertexMass = 5, float  
centerMass = 0.1, float centerInertia = 0.05, float softness = 1)
```

Parameters

size [Vector3](#)

The size of the cube.

scale [Vector3](#)?

The scale of the cube.

vertexMass [float](#)

Mass of each vertex (default is 5.0).

centerMass [float](#)

Mass of the cube's center (default is 0.1).

centerInertia [float](#)

Inertia of the cube's center (default is 0.05).

softness [float](#)

Softness factor of the body (default is 1.0).

Properties

CenterInertia

The inertia value applied to the center of the cube.

```
public float CenterInertia { get; }
```

Property Value

[float](#)

CenterMass

The mass assigned to the center of the cube.

```
public float CenterMass { get; }
```

Property Value

[float](#)

Scale

The scale factor applied to the soft body dimensions during creation.

```
public Vector3 Scale { get; }
```

Property Value

[Vector3](#)

Size

The size dimensions of the soft body cube.

```
public Vector3 Size { get; }
```

Property Value

[Vector3](#)

Softness

Controls how soft or rigid the body behaves.

```
public float Softness { get; }
```

Property Value

[float](#)

VertexMass

The mass assigned to each vertex of the cube.

```
public float VertexMass { get; }
```

Property Value

[float](#)

Methods

CreateSoftBody(GraphicsDevice, World, Vector3, Quaternion)

Creates a new soft body instance in the given world with specified parameters.

```
public SimpleSoftBody CreateSoftBody(GraphicsDevice graphicsDevice, World world, Vector3 position, Quaternion rotation)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used to render the soft body.

world [World](#)

The simulation world to which the soft body will be added.

position [Vector3](#)

The position where the soft body will be placed.

rotation [Quaternion](#)

The rotation of the soft body in the world.

Returns

[SimpleSoftBody](#)

A new instance of a [SimpleSoftBody](#) initialized with the specified parameters.

Class SoftBodySphereFactory

Namespace: [Sparkle.CSharp.Physics.Dim3.SoftBodies.Factories](#)

Assembly: Sparkle.dll

```
public class SoftBodySphereFactory : ISoftBodyFactory
```

Inheritance

[object](#) ← SoftBodySphereFactory

Implements

[ISoftBodyFactory](#)

Inherited Members

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

Constructors

SoftBodySphereFactory(Vector3, Vector3?, int, float, float, float, float)

Initializes a new instance of the [SoftBodySphereFactory](#) class with parameters used to generate soft-body spheres.

```
public SoftBodySphereFactory(Vector3 size, Vector3? scale = null, int subdivisions = 4,  
    float vertexMass = 200, float centerMass = 1, float centerInertia = 0.05, float softness  
    = 0.75)
```

Parameters

size [Vector3](#)

The base size of the soft-body sphere.

scale [Vector3](#)?

Optional scaling factor to apply to the sphere. If null, defaults to [One](#).

subdivisions [int](#)

The number of recursive subdivisions used to refine the unit sphere's mesh.

vertexMass [float](#)

The mass to assign to each vertex in the generated soft body. Default is 200.0.

centerMass [float](#)

The mass to assign to the central rigid body. Default is 1.0.

centerInertia [float](#)

The inertia tensor scale to use for the central body. Default is 0.05.

softness [float](#)

The softness value used for spring constraints between the center and vertex bodies. Default is 0.75.

Properties

CenterInertia

The inertia factor of the central body.

```
public float CenterInertia { get; }
```

Property Value

[float](#)

CenterMass

The mass of the central body in the soft-body structure.

```
public float CenterMass { get; }
```

Property Value

[float](#)

Scale

The scale applied to the sphere's geometry.

```
public Vector3 Scale { get; }
```

Property Value

[Vector3](#)

Size

The size of the soft-body sphere before scaling.

```
public Vector3 Size { get; }
```

Property Value

[Vector3](#)

Softness

The softness value used for spring constraints.

```
public float Softness { get; }
```

Property Value

[float](#)

Subdivisions

The number of recursive subdivisions used to refine the sphere's geometry.

```
public int Subdivisions { get; }
```

Property Value

[int ↗](#)

VertexMass

The mass of each individual vertex in the soft-body mesh.

```
public float VertexMass { get; }
```

Property Value

[float ↗](#)

Methods

CreateSoftBody(GraphicsDevice, World, Vector3, Quaternion)

Creates a soft body sphere instance using the specified parameters and factory settings.

```
public SimpleSoftBody CreateSoftBody(GraphicsDevice graphicsDevice, World world, Vector3  
position, Quaternion rotation)
```

Parameters

graphicsDevice [GraphicsDevice ↗](#)

The graphics device used for rendering the soft body sphere.

world [World](#)

The physics world where the soft body sphere will reside.

position [Vector3 ↗](#)

The initial position of the soft body sphere.

`rotation` [Quaternion](#)

The initial rotation of the soft body sphere.

Returns

[SimpleSoftBody](#)

A new instance of [SimpleSoftBody](#) representing the soft body sphere.

Namespace Sparkle.CSharp.Physics.Dim3.SoftBodies.Types

Classes

[SimpleSoftBody](#)

[SoftBodyCloth](#)

[SoftBodyCube](#)

[SoftBodySphere](#)

Class SimpleSoftBody

Namespace: [Sparkle.CSharp.Physics.Dim3.SoftBodies.Types](#)

Assembly: Sparkle.dll

```
public abstract class SimpleSoftBody : SoftBody, IDebugDrawable
```

Inheritance

[object](#) ← SoftBody ← SimpleSoftBody

Implements

IDebugDrawable

Derived

[SoftBodyCloth](#), [SoftBodyCube](#), [SoftBodySphere](#)

Inherited Members

SoftBody.AddShape(SoftBodyShape), SoftBody.AddSpring(Constraint), SoftBody.Vertices,
SoftBody.Springs, SoftBody.Shapes, SoftBody.World, SoftBody.IsActive, [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Constructors

SimpleSoftBody(GraphicsDevice, World)

Initializes a new instance of the [SimpleSoftBody](#) class.

```
protected SimpleSoftBody(GraphicsDevice graphicsDevice, World world)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for rendering the mesh.

world World

The physics world this soft body is part of.

Properties

Center

The central rigid body representing the core of the soft body structure.

```
public abstract Rigidbody Center { get; protected set; }
```

Property Value

Rigidbody

GraphicsDevice

The graphics device used to create and manage GPU resources.

```
public GraphicsDevice GraphicsDevice { get; }
```

Property Value

[GraphicsDevice](#) ↗

Mesh

The GPU mesh associated with the soft body. Created lazily on first access.

```
public Mesh Mesh { get; }
```

Property Value

Mesh

Methods

CreateMesh(GraphicsDevice)

Creates the mesh representation of this soft body using the specified graphics device.

```
protected abstract Mesh CreateMesh(GraphicsDevice graphicsDevice)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used to create the mesh.

Returns

Mesh

A new [Mesh](#) instance representing the soft body.

DebugDraw(IDebugDrawer)

Draws debug visuals for this soft body using the specified debug drawer.

```
public abstract void DebugDraw(IDebugDrawer drawer)
```

Parameters

drawer IDebugDrawer

The debug drawer to use for rendering visual helpers.

Destroy()

Destroys the soft body, removing all its components from the simulation world.

```
public override void Destroy()
```

GetLerpedVertexPos(int)

Gets the interpolated position of a vertex at the specified index based on its history.

```
public Vector3 GetLerpedVertexPos(int index)
```

Parameters

index [int](#)

The index of the vertex.

Returns

[Vector3](#)

The interpolated position of the vertex. If interpolation data is not available, the current position of the vertex is returned.

UpdateMesh(CommandList)

Updates the mesh with the latest simulation and rendering data.

```
protected abstract void UpdateMesh(CommandList commandList)
```

Parameters

commandList [CommandList](#)

The command list used to issue rendering commands to the GPU.

WorldOnPostStep(float)

Executes actions required after the simulation step has completed.

```
protected override void WorldOnPostStep(float dt)
```

Parameters

dt [float](#)

The time step duration since the last simulation update.

Class SoftBodyCloth

Namespace: [Sparkle.CSharp.Physics.Dim3.SoftBodies.Types](#)

Assembly: Sparkle.dll

```
public class SoftBodyCloth : SimpleSoftBody, IDebugDrawable
```

Inheritance

[object](#) ← SoftBody ← [SimpleSoftBody](#) ← SoftBodyCloth

Implements

IDebugDrawable

Inherited Members

[SimpleSoftBody.GraphicsDevice](#) , [SimpleSoftBody.Mesh](#) , [SimpleSoftBody.WorldOnPostStep\(float\)](#) ,
[SimpleSoftBody.GetLerpedVertexPos\(int\)](#) , [SimpleSoftBody.Destroy\(\)](#) ,
SoftBody.AddShape(SoftBodyShape) , SoftBody.AddSpring(Constraint) , SoftBody.Vertices ,
SoftBody.Springs , SoftBody.Shapes , SoftBody.World , SoftBody.IsActive , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

SoftBodyCloth(GraphicsDevice, World, Vector3, Quaternion, int, int, Vector2, float, float, float, float, bool, bool)

Initializes a new instance of the [SoftBodyCloth](#) class.

```
public SoftBodyCloth(GraphicsDevice graphicsDevice, World world, Vector3 position,  
Quaternion rotation, int gridWidth, int gridHeight, Vector2 vertexSpacing, float vertexMass  
= 10, float centerMass = 1, float centerInertia = 0.05, float softness = 0.2, bool  
useDynamicCenterVertexSelection = true, bool useGridTexture = false)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for mesh creation.

world [World](#)

The physics world in which the cloth is simulated.

position [Vector3](#)

The position of the cloth in world space.

rotation [Quaternion](#)

The rotation applied to the cloth's initial layout.

gridWidth [int](#)

The number of columns in the cloth grid.

gridHeight [int](#)

The number of rows in the cloth grid.

vertexSpacing [Vector2](#)

The spacing between vertices in the grid.

vertexMass [float](#)

The mass of each individual vertex.

centerMass [float](#)

The mass of the central rigid body.

centerInertia [float](#)

The inertia of the central rigid body.

softness [float](#)

The softness factor applied to the spring constraints.

useDynamicCenterVertexSelection [bool](#)

Whether to automatically choose multiple closest vertices to connect to the center.

useGridTexture [bool](#)

Whether to use a tiled grid texture layout for UV coordinates.

Properties

Center

The central rigid body of the soft body.

```
public override sealed Rigidbody Center { get; protected set; }
```

Property Value

Rigidbody

Methods

CreateMesh(GraphicsDevice)

Creates a mesh representation for the soft body cloth.

```
protected override Mesh CreateMesh(GraphicsDevice graphicsDevice)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for creating the mesh.

Returns

Mesh

A Bliss.CSharp.Geometry.Mesh object containing the vertices, indices, and material for the soft body cloth.

DebugDraw(IDebugDrawer)

Draws debug visualization of the soft body cloth, including its triangles, springs, and center point.

```
public override void DebugDraw(IDebugDrawer drawer)
```

Parameters

drawer IDebugDrawer

The debug drawer used for rendering visual debug information.

UpdateMesh(CommandList)

Updates the mesh data of the soft body cloth by recalculating vertex positions, texture coordinates, and colors.

```
protected override void UpdateMesh(CommandList commandList)
```

Parameters

commandList [CommandList](#)

The command list used to update the vertex buffer with the new mesh data.

Class SoftBodyCube

Namespace: [Sparkle.CSharp.Physics.Dim3.SoftBodies.Types](#)

Assembly: Sparkle.dll

```
public class SoftBodyCube : SimpleSoftBody, IDebugDrawable
```

Inheritance

[object](#) ← SoftBody ← [SimpleSoftBody](#) ← SoftBodyCube

Implements

IDebugDrawable

Inherited Members

[SimpleSoftBody.GraphicsDevice](#) , [SimpleSoftBody.Mesh](#) , [SimpleSoftBody.WorldOnPostStep\(float\)](#) ,
[SimpleSoftBody.GetLerpedVertexPos\(int\)](#) , [SimpleSoftBody.Destroy\(\)](#) ,
SoftBody.AddShape(SoftBodyShape) , SoftBody.AddSpring(Constraint) , SoftBody.Vertices ,
SoftBody.Springs , SoftBody.Shapes , SoftBody.World , SoftBody.IsActive , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

SoftBodyCube(GraphicsDevice, World, Vector3, Quaternion, Vector3, Vector3, float, float, float, float)

Constructs a new soft body cube within the specified physics world.

```
public SoftBodyCube(GraphicsDevice graphicsDevice, World world, Vector3 position, Quaternion rotation, Vector3 size, Vector3 scale, float vertexMass = 5, float centerMass = 0.1, float centerInertia = 0.05, float softness = 1)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for rendering-related operations.

world World

The physics simulation world.

position [Vector3](#)

The initial position of the cube.

rotation [Quaternion](#)

The rotation of the cube.

size [Vector3](#)

The actual physical size of the cube.

scale [Vector3](#)

Scale applied to the cube vertices before positioning.

vertexMass [float](#)

Mass applied to each vertex body.

centerMass [float](#)

Mass of the central rigid body.

centerInertia [float](#)

Inertia tensor multiplier for the central body.

softness [float](#)

Softness of the constraints connecting vertices to the center.

Exceptions

[ArgumentException](#)

Thrown if any size component is less than or equal to zero.

Fields

Edges

Defines the edges connecting the vertices to form a cube.

```
public static readonly (int, int)[] Edges
```

Field Value

(int[], int[])[]

Properties

Center

The central rigid body of the soft body.

```
public override sealed Rigidbody Center { get; protected set; }
```

Property Value

Rigidbody

Methods

CreateMesh(GraphicsDevice)

Creates a mesh representation of the soft body cube for rendering purposes.

```
protected override Mesh CreateMesh(GraphicsDevice graphicsDevice)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for rendering-related operations.

Returns

Mesh

A new Bliss.CSharp.Geometry.Mesh instance representing the soft body cube.

DebugDraw(IDebugDrawer)

Renders the debug visualization for the soft body by drawing its edges and center.

```
public override void DebugDraw(IDebugDrawer drawer)
```

Parameters

drawer IDebugDrawer

The debug drawer responsible for rendering shapes and points.

UpdateMesh(CommandList)

Updates the vertex data of the mesh to match the current vertex positions of the soft body.

```
protected override void UpdateMesh(CommandList commandList)
```

Parameters

commandList [CommandList](#)

The command list used for issuing graphics commands to the GPU.

Class SoftBodySphere

Namespace: [Sparkle.CSharp.Physics.Dim3.SoftBodies.Types](#)

Assembly: Sparkle.dll

```
public class SoftBodySphere : SimpleSoftBody, IDebugDrawable
```

Inheritance

[object](#) ← SoftBody ← [SimpleSoftBody](#) ← SoftBodySphere

Implements

IDebugDrawable

Inherited Members

[SimpleSoftBody.GraphicsDevice](#) , [SimpleSoftBody.Mesh](#) , [SimpleSoftBody.WorldOnPostStep\(float\)](#) ,
[SimpleSoftBody.GetLerpedVertexPos\(int\)](#) , [SimpleSoftBody.Destroy\(\)](#) ,
SoftBody.AddShape(SoftBodyShape) , SoftBody.AddSpring(Constraint) , SoftBody.Vertices ,
SoftBody.Springs , SoftBody.Shapes , SoftBody.World , SoftBody.IsActive , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

SoftBodySphere(GraphicsDevice, World, Vector3, Quaternion, Vector3, Vector3, int, float, float, float, float)

Initializes a new instance of the [SoftBodySphere](#) class, constructing a soft-body physics object in the shape of a sphere using a triangulated hull based on a subdivided unit sphere.

```
public SoftBodySphere(GraphicsDevice graphicsDevice, World world, Vector3 position,  
Quaternion rotation, Vector3 size, Vector3 scale, int subdivisions = 4, float vertexMass =  
200, float centerMass = 1, float centerInertia = 0.05, float softness = 0.75)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used for rendering.

world [World](#)

The physics simulation world to which the soft body belongs.

position [Vector3](#)

The world position where the soft body should be placed.

rotation [Quaternion](#)

The rotation to apply to the soft body sphere.

size [Vector3](#)

The base size of the sphere before scaling and rotation.

scale [Vector3](#)

The scaling factor to apply to the size of the sphere.

subdivisions [int](#)

The number of recursive subdivisions used to refine the unit sphere's mesh.

vertexMass [float](#)

The mass of each vertex rigid body. Default is 120.0.

centerMass [float](#)

The mass of the central rigid body connecting all vertices. Default is 1.0.

centerInertia [float](#)

The inertia tensor scaling factor for the central body. Default is 0.05.

softness [float](#)

The softness value used for spring constraints between the center and vertex bodies. Default is 0.5.

Properties

Center

The central rigid body of the soft body.

```
public override sealed Rigidbody Center { get; protected set; }
```

Property Value

Rigidbody

Methods

CreateMesh(GraphicsDevice)

Creates a mesh representing the current soft body structure based on its geometry and material properties.

```
protected override Mesh CreateMesh(GraphicsDevice graphicsDevice)
```

Parameters

graphicsDevice [GraphicsDevice](#)

The graphics device used to create the mesh and render its contents.

Returns

Mesh

A mesh constructed from the soft body geometry, configured with vertices, indices, and material.

DebugDraw(IDebugDrawer)

Renders debug visualizations for the soft body.

```
public override void DebugDraw(IDebugDrawer drawer)
```

Parameters

drawer IDebugDrawer

The debug drawer used to render the visualizations.

UpdateMesh(CommandList)

Updates the mesh to reflect the current positions of the vertices in the soft body structure.

```
protected override void UpdateMesh(CommandList commandList)
```

Parameters

`commandList` [CommandList](#)

The command list used to update the vertex buffer for rendering.

Namespace Sparkle.CSharp.Registries

Classes

[Registry](#)

[RegistryManager](#)

Class Registry

Namespace: [Sparkle.CSharp.Registries](#)

Assembly: Sparkle.dll

```
public abstract class Registry : Disposable, IDisposable
```

Inheritance

[object](#) ← Disposable ← Registry

Implements

[IDisposable](#)

Derived

[ContentRegistry](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Init()

Initializes the registry after content has been loaded.

```
protected virtual void Init()
```

Load(ContentManager)

Loads content using the specified [ContentManager](#).

```
protected virtual void Load(ContentManager content)
```

Parameters

content [ContentManager](#)

The content manager used to load assets.

Class RegistryManager

Namespace: [Sparkle.CSharp.Registries](#)

Assembly: Sparkle.dll

```
public static class RegistryManager
```

Inheritance

[object](#) ← RegistryManager

Inherited Members

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

Methods

AddRegistry(Registry)

Adds a registry to the registry manager.

```
public static void AddRegistry(Registry registry)
```

Parameters

registry [Registry](#)

The registry instance to add to the manager.

GetRegistries()

Retrieves an enumerable collection of all active registry instances managed by the registry manager.

```
public static IEnumerable<Registry> GetRegistries()
```

Returns

[IEnumerable](#) <[Registry](#)>

An enumerable collection of [Registry](#) instances currently registered.

GetRegistry<T>()

Retrieves a registry of the specified type from the registry manager.

```
public static T? GetRegistry<T>() where T : Registry
```

Returns

T

Returns an instance of the registry if found; otherwise, null.

Type Parameters

T

The type of the registry to retrieve.

HasRegistry<T>()

Determines whether a registry of the specified type exists in the registry manager.

```
public static bool HasRegistry<T>() where T : Registry
```

Returns

bool

True if a registry of the specified type exists; otherwise, false.

Type Parameters

T

The type of the registry to check for.

RemoveRegistry(Registry)

Removes a specified registry from the list of managed registries.

```
public static void RemoveRegistry(Registry registry)
```

Parameters

`registry Registry`

The registry to be removed.

RemoveRegistry<T>()

Removes a registry of the specified type from the registry manager.

```
public static void RemoveRegistry<T>() where T : Registry
```

Type Parameters

`T`

The type of the registry to remove.

TryAddRegistry(Registry)

Attempts to add a new registry to the registry manager.

```
public static bool TryAddRegistry(Registry registry)
```

Parameters

`registry Registry`

The registry instance to add.

Returns

[bool](#)

Returns true if the registry was successfully added; false if the registry type is already present.

TryGetRegistry<T>(out T?)

Attempts to retrieve a registry of the specified type from the registry manager.

```
public static bool TryGetRegistry<T>(out T? registry) where T : Registry
```

Parameters

[registry](#) [T](#)

The output parameter that will contain the registry instance if found; otherwise, null.

Returns

[bool](#)

Returns true if the registry is successfully located; otherwise, false.

Type Parameters

[T](#)

The type of the registry to retrieve.

TryRemoveRegistry(Registry)

Attempts to remove a specified registry from the list of managed registries.

```
public static bool TryRemoveRegistry(Registry registry)
```

Parameters

[registry](#) [Registry](#)

The registry to be removed.

Returns

[bool](#)

True if the registry is successfully removed; otherwise, false if the registry does not exist in the list.

TryRemoveRegistry<T>()

Attempts to remove the specified registry type from the list of managed registries.

```
public static bool TryRemoveRegistry<T>() where T : Registry
```

Returns

[bool](#)

True if the registry was successfully removed; otherwise, false.

Type Parameters

T

The type of the registry to remove.

Namespace Sparkle.CSharp.Scenes

Classes

[Scene](#)

[SceneManager](#)

Enums

[SceneType](#)

Class Scene

Namespace: [Sparkle.CSharp.Scenes](#)

Assembly: Sparkle.dll

```
public abstract class Scene : Disposable, IDisposable
```

Inheritance

[object](#) ← Disposable ← Scene

Implements

[IDisposable](#)

Derived

[TestScene2D](#), [TestScene3D](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

Scene(string, SceneType, Func<GraphicsDevice, IRenderer>?, Func<Simulation>?)

Initializes a new instance of the [Scene](#) class.

```
protected Scene(string name, SceneType sceneType, Func<GraphicsDevice, IRenderer>?  
rendererFactory = null, Func<Simulation>? simulationFactory = null)
```

Parameters

name [string](#)

The name of the scene.

sceneType [SceneType](#)

The type of the scene (2D or 3D).

`rendererFactory` [Func<GraphicsDevice, IRenderer>](#)

Optional factory used to create a custom renderer for the scene.

`simulationFactory` [Func<Simulation>](#)

Optional factory used to create a custom physics simulation.

Fields

FilterEffect

The filter effect used in the scene.

```
public Effect? FilterEffect
```

Field Value

Effect

SkyBox

The skybox used in the scene.

```
public SkyBox? SkyBox
```

Field Value

[SkyBox](#)

Properties

Name

Gets the name of the scene.

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

Property Value

[string](#)

Physics3DDrawDrawer

Provides tools for visual debugging of 3D physics simulations.

```
public Physics3DDrawDrawer Physics3DDrawDrawer { get; }
```

Property Value

[Physics3DDrawDrawer](#)

Renderer

Gets the forward renderer responsible for rendering 3D content in the scene.

```
public IRenderer Renderer { get; }
```

Property Value

[IRenderer](#)

SceneType

Gets the type of the scene (2D or 3D).

```
public SceneType SceneType { get; }
```

Property Value

[SceneType](#)

Simulation

Gets the physics simulation associated with the scene.

```
public Simulation Simulation { get; }
```

Property Value

[Simulation](#)

SpriteRenderer

Manages and draw sprites for the entity sprite component.

```
public SpriteRenderer SpriteRenderer { get; }
```

Property Value

[SpriteRenderer](#)

Methods

AddEntity(Entity)

Adds an entity to the scene.

```
public void AddEntity(Entity entity)
```

Parameters

entity [Entity](#)

The entity to add.

Exceptions

[Exception](#) ↗

Thrown if the entity is already present in the scene or belongs to another scene.

AfterUpdate(double)

Called after the main update loop. Allows for post-update logic.

```
protected virtual void AfterUpdate(double delta)
```

Parameters

delta [double](#)

The time elapsed since the last update.

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Draw(GraphicsContext, Framebuffer)

Draws all entities in the scene.

```
protected virtual void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context.

framebuffer [Framebuffer](#)

The framebuffer to render into.

FixedUpdate(double)

Performs a fixed update for the current scene, updating the simulation and invoking the fixed update for all entities.

```
protected virtual void FixedUpdate(double fixedStep)
```

Parameters

fixedStep [double](#)

The fixed time interval to update the scene and entities.

GetEntities()

Retrieves all entities in the scene.

```
public IEnumerable<Entity> GetEntities()
```

Returns

[IEnumerable](#) <[Entity](#)>

GetEntitiesWithTag(string)

Retrieves all entities with a specific tag.

```
public IEnumerable<Entity> GetEntitiesWithTag(string tag)
```

Parameters

tag [string](#)

The tag to filter entities.

Returns

[IEnumerable](#) <[Entity](#)>

GetEntity(uint)

Gets an entity by its ID.

```
public Entity? GetEntity(uint id)
```

Parameters

id [uint](#)

The entity ID.

Returns

[Entity](#)

HasEntity(uint)

Checks if an entity exists in the scene.

```
public bool HasEntity(uint id)
```

Parameters

id [uint](#)

The entity ID.

Returns

[bool](#)

Init()

Initializes the scene. Can be overridden in derived classes.

```
protected virtual void Init()
```

RemoveEntity(Entity)

Removes an entity from the scene.

```
public void RemoveEntity(Entity entity)
```

Parameters

entity [Entity](#)

The entity to remove.

Exceptions

[Exception](#) ↗

Thrown if the entity could not be removed.

RemoveEntity(uint)

Removes an entity from the scene by its ID.

```
public void RemoveEntity(uint id)
```

Parameters

id [uint](#) ↗

The ID of the entity to remove.

Exceptions

Exception

Thrown if the entity could not be removed.

Resize(Rectangle)

Handles window resizing events and updates entities accordingly.

```
protected virtual void Resize(Rectangle rectangle)
```

Parameters

rectangle Rectangle

The new window dimensions.

TryAddEntity(Entity)

Attempts to add an entity to the scene.

```
public bool TryAddEntity(Entity entity)
```

Parameters

entity Entity

The entity to add.

Returns

bool

True if the entity was successfully added; otherwise, false.

TryGetEntity(uint, out Entity?)

Tries to retrieve an entity by its ID.

```
public bool TryGetEntity(uint id, out Entity? entity)
```

Parameters

id [uint](#)

The entity ID.

entity [Entity](#)

The retrieved entity, if found.

Returns

[bool](#)

TryRemoveEntity(Entity)

Attempts to remove an entity from the scene.

```
public bool TryRemoveEntity(Entity entity)
```

Parameters

entity [Entity](#)

The entity to remove.

Returns

[bool](#)

True if the entity was successfully removed; otherwise, false.

TryRemoveEntity(uint)

Attempts to remove an entity from the scene by its ID.

```
public bool TryRemoveEntity(uint id)
```

Parameters

id [uint](#)

The ID of the entity to remove.

Returns

[bool](#)

True if the entity was successfully removed; otherwise, false.

Update(double)

Updates all entities in the scene.

```
protected virtual void Update(double delta)
```

Parameters

delta [double](#)

The time elapsed since the last update.

Class SceneManager

Namespace: [Sparkle.CSharp.Scenes](#)

Assembly: Sparkle.dll

```
public static class SceneManager
```

Inheritance

[object](#) ← SceneManager

Inherited Members

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

Fields

ActiveCam2D

The active 2D camera in the scene.

```
public static Camera2D? ActiveCam2D
```

Field Value

[Camera2D](#)

ActiveCam3D

The active 3D camera in the scene.

```
public static Camera3D? ActiveCam3D
```

Field Value

[Camera3D](#)

PostEffect

The post-processing effect applied to the rendered scene.

```
public static Effect? PostEffect
```

Field Value

Effect

Properties

ActiveScene

Gets the currently active scene.

```
public static Scene? ActiveScene { get; }
```

Property Value

[Scene](#)

FilterResult

The final texture result obtained after applying filter effects to a scene.

```
public static Texture2D FilterResult { get; }
```

Property Value

Texture2D

FilterTarget

The render target used for filter effects.

```
public static RenderTexture2D FilterTarget { get; }
```

Property Value

RenderTexture2D

GraphicsDevice

The graphics device used for rendering.

```
public static GraphicsDevice GraphicsDevice { get; }
```

Property Value

[GraphicsDevice](#)

PostProcessingResult

The final texture result obtained after applying post-processing effects to a scene.

```
public static Texture2D PostProcessingResult { get; }
```

Property Value

Texture2D

PostProcessingTarget

The render target used for post-processing effects.

```
public static RenderTexture2D PostProcessingTarget { get; }
```

Property Value

RenderTexture2D

Simulation

Gets the simulation associated with the active scene.

```
public static Simulation? Simulation { get; }
```

Property Value

[Simulation](#)

Methods

SetScene(Scene?)

Sets a new active scene and initializes it.

```
public static void SetScene(Scene? scene)
```

Parameters

`scene` [Scene](#)

The scene to set as active.

Enum SceneType

Namespace: [Sparkle.CSharp.Scenes](#)

Assembly: Sparkle.dll

```
public enum SceneType
```

Fields

Scene2D = 0

A 2D scene, typically used for 2D rendering.

Scene3D = 1

A 3D scene, typically used for 3D rendering.

Namespace Sparkle.CSharp.Terrain

Classes

[MarchingCubes](#)

[MarchingCubesChunk](#)

[MarchingCubesTables](#)

Class MarchingCubes

Namespace: [Sparkle.CSharp.Terrain](#)

Assembly: Sparkle.dll

```
public class MarchingCubes
```

Inheritance

[object](#) ← MarchingCubes

Inherited Members

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

Constructors

MarchingCubes(int, int, int, float, float, bool)

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

```
public MarchingCubes(int seed, int width, int height, float scale, float heightThreshold,  
bool use3DNoise = false)
```

Parameters

seed [int](#)

Seed value for initializing the Perlin noise generator.

width [int](#)

Width of the 3D grid.

height [int](#)

Height of the 3D grid.

scale [float](#)

Scaling factor applied to the grid.

heightThreshold [float](#)

Threshold value determining the surface generation.

use3DNoise [bool](#)

Optional parameter indicating whether to use 3D noise for generating terrain.

Methods

GenMesh(GraphicsDevice)

Generates a mesh based on the vertices, normals, texture coordinates, and indices stored within the MarchingCubes instance.

```
public Mesh GenMesh(GraphicsDevice graphicsDevice)
```

Parameters

graphicsDevice [GraphicsDevice](#)

Returns

Mesh

MarchCubes(Vector3, int, int)

Marches cubes within a specified region to generate terrain.

```
public void MarchCubes(Vector3 position, int width, int height)
```

Parameters

position [Vector3](#)

The position of the starting corner of the region.

width [int](#)

The width of the region in units.

height [int](#)

The height of the region in units.

SetHeights(Vector3, int, int)

Sets heights for the terrain within a specified region.

```
public void SetHeights(Vector3 position, int width, int height)
```

Parameters

position [Vector3](#)

The position of the starting corner of the region.

width [int](#)

The width of the region in units.

height [int](#)

The height of the region in units.

Class MarchingCubesChunk

Namespace: [Sparkle.CSharp.Terrain](#)

Assembly: Sparkle.dll

```
public class MarchingCubesChunk : Disposable, IDisposable
```

Inheritance

[object](#) ← Disposable ← MarchingCubesChunk

Implements

[IDisposable](#)

Inherited Members

Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

MarchingCubesChunk(MarchingCubes, Vector3, int, int)

Initializes a new instance of the [MarchingCubesChunk](#) class.

```
public MarchingCubesChunk(MarchingCubes marchingCubes, Vector3 position, int width,  
int height)
```

Parameters

marchingCubes [MarchingCubes](#)

The MarchingCubes instance associated with this chunk.

position [Vector3](#)

The position of the chunk within the grid.

width [int](#)

The width of the chunk in units.

height [int](#)

The height of the chunk in units.

Properties

Height

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

Property Value

[int](#)

MarchingCubes

```
public MarchingCubes MarchingCubes { get; }
```

Property Value

[MarchingCubes](#)

Mesh

```
public Mesh Mesh { get; }
```

Property Value

Mesh

Position

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

Property Value

[Vector3](#)

Width

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

Property Value

[int](#)

Methods

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Generate()

Generates a Marching Cubes chunk.

```
public void Generate()
```

Class MarchingCubesTables

Namespace: [Sparkle.CSharp.Terrain](#)

Assembly: Sparkle.dll

```
public static class MarchingCubesTables
```

Inheritance

[object](#) ← MarchingCubesTables

Inherited Members

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

Fields

Corners

Represents the eight corners of a cube in 3D space.

```
public static readonly Vector3[] Corners
```

Field Value

[Vector3](#)[]

Edges

Represents the edges of a cube in 3D space.

```
public static readonly Vector3[,] Edges
```

Field Value

[Vector3](#)[,]

Triangles

Represents a collection of triangles.

```
public static readonly int[,] Triangles
```

Field Value

[int](#)[]

Namespace Sparkle.Test.CSharp

Classes

[ContentRegistry](#)

[TestGame](#)

[TestGui](#)

[TestOverlay](#)

Class ContentRegistry

Namespace: [Sparkle.Test.CSharp](#)

Assembly: Sparkle.Test.dll

```
public class ContentRegistry : Registry, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Registry](#) ← ContentRegistry

Implements

[IDisposable](#)

Inherited Members

[Registry.Init\(\)](#) , [Disposable.Dispose\(\)](#) , [Disposable.ThrowIfDisposed\(\)](#) , [Disposable.HasDisposed](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Fields

AnimatedImage

```
public static AnimatedImage AnimatedImage
```

Field Value

AnimatedImage

Gif

```
public static Texture2D Gif
```

Field Value

Texture2D

Properties

Button

```
public static Texture2D Button { get; }
```

Property Value

Texture2D

CyberCarModel

```
public static Model CyberCarModel { get; }
```

Property Value

Model

CyberCarTexture

```
public static Texture2D CyberCarTexture { get; }
```

Property Value

Texture2D

FilledSlideBar

```
public static Texture2D FilledSlideBar { get; }
```

Property Value

Texture2D

Fontoe

```
public static Font Fontoe { get; }
```

Property Value

Font

PlayerModel

```
public static Model PlayerModel { get; }
```

Property Value

Model

PlayerMultiInstanceRenderer

```
public static MultiInstanceRenderer PlayerMultiInstanceRenderer { get; }
```

Property Value

[MultiInstanceRenderer](#)

SkyBox

```
public static SkyBox SkyBox { get; }
```

Property Value

[SkyBox](#)

SlideBar

```
public static Texture2D SlideBar { get; }
```

Property Value

Texture2D

Slider

```
public static Texture2D Slider { get; }
```

Property Value

Texture2D

Sprite

```
public static Texture2D Sprite { get; }
```

Property Value

Texture2D

TextBox

```
public static Texture2D TextBox { get; }
```

Property Value

Texture2D

ToggleBackground

```
public static Texture2D ToggleBackground { get; }
```

Property Value

Texture2D

ToggleCheckmark

```
public static Texture2D ToggleCheckmark { get; }
```

Property Value

Texture2D

TreeModel

```
public static Model TreeModel { get; }
```

Property Value

Model

UiArrowTexture

```
public static Texture2D UiArrowTexture { get; }
```

Property Value

Texture2D

UiBannerEdgeLessTexture

```
public static Texture2D UiBannerEdgeLessTexture { get; }
```

Property Value

Texture2D

UiBannerTexture

```
public static Texture2D UiBannerTexture { get; }
```

Property Value

Texture2D

UiSliderTexture

```
public static Texture2D UiSliderTexture { get; }
```

Property Value

Texture2D

Methods

Dispose(bool)

Disposes of the object and releases associated resources.

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

Parameters

disposing [bool](#)

True if called from user code; false if called from a finalizer.

Load(ContentManager)

Loads content using the specified [ContentManager](#).

```
protected override void Load(ContentManager content)
```

Parameters

content [ContentManager](#)

The content manager used to load assets.

Class TestGame

Namespace: [Sparkle.Test.CSharp](#)

Assembly: Sparkle.Test.dll

```
public class TestGame : Game, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Game](#) ← TestGame

Implements

[IDisposable](#)

Inherited Members

[Game.Version](#) , [Game.Instance](#) , [Game.Settings](#) , [Game.MainWindow](#) , [Game.GraphicsDevice](#) ,
[Game.CommandList](#) , [Game.FullScreenRenderPass](#) , [Game.GlobalSpriteBatch](#) ,
[Game.GlobalPrimitiveBatch](#) , [Game.GlobalImmediateRenderer](#) , [Game.GraphicsContext](#) , [Game.Content](#) ,
[Game.ShouldClose](#) , [Game.Run\(Scene\)](#) , [Game.Load\(ContentManager\)](#) , [Game.Update\(double\)](#) ,
[Game.AfterUpdate\(double\)](#) , [Game.FixedUpdate\(double\)](#) , [Game.Draw\(GraphicsContext, Framebuffer\)](#) ,
[Game.OnResize\(Rectangle\)](#) , [Game.OnClose\(\)](#) , [Game.GetTargetFps\(\)](#) , [Game.SetTargetFps\(int\)](#) ,
[Game.GetSampleCount\(\)](#) , [Game.SetSampleCount\(TextureSampleCount\)](#) , [Game.Dispose\(bool\)](#) ,
Disposable.Dispose() , Disposable.ThrowIfDisposed() , Disposable.HasDisposed , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

TestGame(GameSettings)

```
public TestGame(GameSettings settings)
```

Parameters

settings [GameSettings](#)

Methods

Init()

Initializes global game resources.

```
protected override void Init()
```

OnRun()

Virtual method for additional setup when the game starts.

```
protected override void OnRun()
```

Class TestGui

Namespace: [Sparkle.Test.CSharp](#)

Assembly: Sparkle.Test.dll

```
public class TestGui : Gui, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Gui](#) ← TestGui

Implements

[IDisposable](#)

Inherited Members

[Gui.Name](#) , [Gui.Size](#) , [Gui.ScaleFactor](#) , [Gui.AfterUpdate\(double\)](#) , [Gui.FixedUpdate\(double\)](#) ,
[Gui.Resize\(Rectangle\)](#) , [Gui.GetElements\(\)](#) , [Gui.HasElement\(string\)](#) , [Gui.GetElement\(string\)](#) ,
[Gui.TryGetElement\(string, out GuiElement\)](#) , [Gui.AddElement\(string, GuiElement\)](#) ,
[Gui.TryAddElement\(string, GuiElement\)](#) , [Gui.RemoveElement\(GuiElement\)](#) ,
[Gui.TryRemoveElement\(GuiElement\)](#) , [Gui.RemoveElement\(string\)](#) , [Gui.TryRemoveElement\(string\)](#) ,
[Gui.Dispose\(bool\)](#) , [Disposable.Dispose\(\)](#) , [Disposable.ThrowIfDisposed\(\)](#) , [Disposable.HasDisposed](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

TestGui(string)

```
public TestGui(string name)
```

Parameters

name [string](#)

Methods

Draw(GraphicsContext, Framebuffer)

Draws all enabled elements to the specified framebuffer using the provided graphics context.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The rendering context.

framebuffer [Framebuffer](#)

The framebuffer to draw into.

Init()

Called when the GUI is initialized.

```
protected override void Init()
```

Update(double)

Updates all enabled elements in the GUI.

```
protected override void Update(double delta)
```

Parameters

delta [double](#)

Elapsed time since the last frame in seconds.

Class TestOverlay

Namespace: [Sparkle.Test.CSharp](#)

Assembly: Sparkle.Test.dll

```
public class TestOverlay : Overlay
```

Inheritance

[object](#) ← [Overlay](#) ← TestOverlay

Inherited Members

[Overlay.Name](#) , [Overlay.Enabled](#) , [Overlay.Update\(double\)](#) , [Overlay.AfterUpdate\(double\)](#) ,
[Overlay.FixedUpdate\(double\)](#) , [Overlay.Resize\(Rectangle\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

TestOverlay(string, bool)

```
public TestOverlay(string name, bool enabled = false)
```

Parameters

name [string](#)

enabled [bool](#)

Methods

Draw(GraphicsContext, Framebuffer)

Draws the overlay.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context used for rendering.

framebuffer [Framebuffer](#)

The framebuffer to which the overlay is rendered.

Namespace Sparkle.Test.CSharp.Dim2D

Classes

[TestScene2D](#)

Class TestScene2D

Namespace: [Sparkle.Test.CSharp.Dim2D](#)

Assembly: Sparkle.Test.dll

```
public class TestScene2D : Scene, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Scene](#) ← TestScene2D

Implements

[IDisposable](#)

Inherited Members

[Scene.Name](#) , [Scene.SceneType](#) , [Scene.Simulation](#) , [Scene.Renderer](#) , [Scene.Physics3DDraw](#) ,
[Scene.SpriteRenderer](#) , [Scene.FilterEffect](#) , [Scene.SkyBox](#) , [Scene.Update\(double\)](#) ,
[Scene.AfterUpdate\(double\)](#) , [Scene.Resize\(Rectangle\)](#) , [Scene.GetEntities\(\)](#) ,
[Scene.GetEntitiesWithTag\(string\)](#) , [Scene.HasEntity\(uint\)](#) , [Scene.GetEntity\(uint\)](#) ,
[Scene.TryGetEntity\(uint, out Entity\)](#) , [Scene.AddEntity\(Entity\)](#) , [Scene.TryAddEntity\(Entity\)](#) ,
[Scene.RemoveEntity\(Entity\)](#) , [Scene.TryRemoveEntity\(Entity\)](#) , [Scene.RemoveEntity\(uint\)](#) ,
[Scene.TryRemoveEntity\(uint\)](#) , [Scene.Dispose\(bool\)](#) , [Disposable.Dispose\(\)](#) ,
[Disposable.ThrowIfDisposed\(\)](#) , [Disposable.HasDisposed](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

TestScene2D(string)

```
public TestScene2D(string name)
```

Parameters

name [string](#)

Methods

Draw(GraphicsContext, Framebuffer)

Draws all entities in the scene.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context.

framebuffer [Framebuffer](#)

The framebuffer to render into.

FixedUpdate(double)

Performs a fixed update for the current scene, updating the simulation and invoking the fixed update for all entities.

```
protected override void FixedUpdate(double delta)
```

Parameters

delta [double](#)

Init()

Initializes the scene. Can be overridden in derived classes.

```
protected override void Init()
```

Namespace Sparkle.Test.CSharp.Dim3D

Classes

[TestScene3D](#)

Class TestScene3D

Namespace: [Sparkle.Test.CSharp.Dim3D](#)

Assembly: Sparkle.Test.dll

```
public class TestScene3D : Scene, IDisposable
```

Inheritance

[object](#) ← Disposable ← [Scene](#) ← TestScene3D

Implements

[IDisposable](#)

Inherited Members

[Scene.Name](#) , [Scene.SceneType](#) , [Scene.Simulation](#) , [Scene.Renderer](#) , [Scene.Physics3DDrawable](#) ,
[Scene.SpriteRenderer](#) , [Scene.FilterEffect](#) , [Scene.SkyBox](#) , [Scene.AfterUpdate\(double\)](#) ,
[Scene.Resize\(Rectangle\)](#) , [Scene.GetEntities\(\)](#) , [Scene.GetEntitiesWithTag\(string\)](#) , [Scene.HasEntity\(uint\)](#) ,
[Scene.GetEntity\(uint\)](#) , [Scene.TryGetEntity\(uint, out Entity\)](#) , [Scene.AddEntity\(Entity\)](#) ,
[Scene.TryAddEntity\(Entity\)](#) , [Scene.RemoveEntity\(Entity\)](#) , [Scene.TryRemoveEntity\(Entity\)](#) ,
[Scene.RemoveEntity\(uint\)](#) , [Scene.TryRemoveEntity\(uint\)](#) , [Scene.Dispose\(bool\)](#) , [Disposable.Dispose\(\)](#) ,
[Disposable.ThrowIfDisposed\(\)](#) , [Disposable.HasDisposed](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

TestScene3D(string)

```
public TestScene3D(string name)
```

Parameters

name [string](#)

Methods

Draw(GraphicsContext, Framebuffer)

Draws all entities in the scene.

```
protected override void Draw(GraphicsContext context, Framebuffer framebuffer)
```

Parameters

context [GraphicsContext](#)

The graphics context.

framebuffer [Framebuffer](#)

The framebuffer to render into.

FixedUpdate(double)

Performs a fixed update for the current scene, updating the simulation and invoking the fixed update for all entities.

```
protected override void FixedUpdate(double timeStep)
```

Parameters

timeStep [double](#)

Init()

Initializes the scene. Can be overridden in derived classes.

```
protected override void Init()
```

Update(double)

Updates all entities in the scene.

```
protected override void Update(double delta)
```

Parameters

delta [double](#)

The time elapsed since the last update.