Namespace Hallowed.Core

Classes

Container
: Represents a scene graph node that can group and transform multiple DisplayObject instances. Containers allow hierarchical transformations, positioning, and rendering order management.

DisplayObject
: The abstract base class for all renderable objects. Every object that can be rendered should inherit from this class.

GameEngine
: The main class for the game that handles all game logic and rendering.

Interfaces

IRenderable

Class Container

Namespace: Hallowed.Core

Assembly: Hallowed.dll

Represents a scene graph node that can group and transform multiple DisplayObject instances. Containers allow hierarchical transformations, positioning, and rendering order management.

```
public class Container : DisplayObject, IDisposable
```

Inheritance

object ← DisplayObject ← Container

Implements

IDisposable

Derived

SceneBase

Inherited Members

DisplayObject.IsDisposed , DisplayObject.WorldTransform , DisplayObject.Parent , DisplayObject.Children , DisplayObject.Visible , DisplayObject.Position , DisplayObject.X , DisplayObject.Y , DisplayObject.Scale , DisplayObject.Rotation , DisplayObject.Origin , DisplayObject.Width , DisplayObject.Height , DisplayObject.GetWorldBounds() , DisplayObject.Update(GameTime) , DisplayObject.Draw(SpriteBatch) , DisplayObject.AddChild(DisplayObject) , DisplayObject.AddChild(IEnumerable<DisplayObject>) , DisplayObject.AddChild(params DisplayObject[]) , DisplayObject.RemoveChild(DisplayObject, bool) , DisplayObject.RemoveChild(IEnumerable<DisplayObject>, bool) , DisplayObject.RemoveChild(bool, params DisplayObject[]) , DisplayObject.MarkDirty() , DisplayObject.UpdateWorldTransform() , DisplayObject.DecomposeMatrix2D(Matrix, out Vector2, out float, out Vector2) , DisplayObject.Dispose() , DisplayObject.Dispose(bool) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

A Container can contain both Sprite and other Container objects, forming a recursive display tree. Transformations applied to the Container (such as position, rotation, or scale) affect all of its children.

Methods

GetBounds()

```
public override Rectangle GetBounds()
```

Returns

Rectangle

Class DisplayObject

Namespace: Hallowed.Core

Assembly: Hallowed.dll

The abstract base class for all renderable objects. Every object that can be rendered should inherit from this class.

```
public abstract class DisplayObject : IDisposable
```

Inheritance

object ← DisplayObject

Implements

IDisposable

Derived

Container, Sprite

Inherited Members

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

Constructors

DisplayObject()

```
protected DisplayObject()
```

Properties

Children

The objects children.

```
public List<DisplayObject> Children { get; protected set; }
```

Property Value

List<DisplayObject>

Height

```
public virtual float Height { get; set; }
```

Property Value

float

IsDisposed

```
public bool IsDisposed { get; }
```

Property Value

bool

Origin

The display object origin.

```
public virtual Vector2 Origin { get; set; }
```

Property Value

Vector2

Parent

The Display object parent.

```
public DisplayObject Parent { get; }
```

Property Value

DisplayObject

Position

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

Property Value

Vector2

Rotation

The display object rotation.

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

Property Value

float

Scale

The display object scale.

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

Property Value

Vector2

Visible

return whether or not the object will render to the screen.

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

Property Value

bool

Width

```
public virtual float Width { get; set; }
```

Property Value

float

WorldTransform

the world matrix of the object.

```
public Matrix WorldTransform { get; }
```

Property Value

Matrix

X

The display object x position. This property is an alias for setting the object X position.

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

Property Value

float

Y

The display object y position. This property is an alias for setting the object Y position.

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

Property Value

float

Methods

AddChild(DisplayObject)

add a child to the end of the children list.

```
public virtual void AddChild(DisplayObject child)
```

Parameters

`child` DisplayObject

AddChild(params DisplayObject[])

add a collection of children to the end of the children list.

```
public virtual void AddChild(params DisplayObject[] children)
```

Parameters

`children` DisplayObject[]

the collection of display object to add

AddChild(IEnumerable<DisplayObject>)

add a collection of children to the end of the children list.

```
public virtual void AddChild(IEnumerable<DisplayObject> children)
```

Parameters

`children` IEnumerable<DisplayObject>

the collection of display object to add

DecomposeMatrix2D(Matrix, out Vector2, out float, out Vector2)

```
protected void DecomposeMatrix2D(Matrix matrix, out Vector2 position, out float rotation,
out Vector2 scale)
```

Parameters

`matrix` Matrix

`position` Vector2

`rotation` float

`scale` Vector2

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

Dispose(bool)

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

Parameters

`disposing` bool

Draw(SpriteBatch)

draw the object and all of its children.

```
public virtual void Draw(SpriteBatch spriteBatch)
```

Parameters

`spriteBatch` SpriteBatch

Remarks

if an object or its children visible flag is set to false it wont be draw but still be updated.

~DisplayObject()

```
protected ~DisplayObject()
```

GetBounds()

```
public abstract Rectangle GetBounds()
```

Returns

Rectangle

GetWorldBounds()

```
public Rectangle GetWorldBounds()
```

Returns

Rectangle

MarkDirty()

Mark the object and all its children as dirty and force a recompute of the their transforms. changing the transform of the object will trigger a recompute the transform.

```
public void MarkDirty()
```

RemoveChild(DisplayObject, bool)

Remove the specified child from the children list.

```
public virtual void RemoveChild(DisplayObject child, bool dispose = false)
```

Parameters

`child` DisplayObject

the child to remove

`dispose` bool

whether or not to dispose the child

RemoveChild(bool, params DisplayObject[])

Remove the specified children from the children list.

```
public virtual void RemoveChild(bool dispose = false, params DisplayObject[] children)
```

Parameters

`dispose` bool

whether or not to dispose the children

`children` DisplayObject[]

the collection of childre to remove

RemoveChild(IEnumerable<DisplayObject>, bool)

Remove the specified children from the children list.

```
public virtual void RemoveChild(IEnumerable<DisplayObject> children, bool dispose = false)
```

Parameters

`children` IEnumerable<DisplayObject>

the collection of children to remove

`dispose` bool

whether or not to dispose the children

Update(GameTime)

update the object and all of its children.

```
public virtual void Update(GameTime gameTime)
```

Parameters

`gameTime` GameTime

UpdateWorldTransform()

```
protected virtual void UpdateWorldTransform()
```

Class GameEngine

Namespace: Hallowed.Core

Assembly: Hallowed.dll

The main class for the game that handles all game logic and rendering.

```
public class GameEngine : Game, IDisposable
```

Inheritance

object ← Game ← GameEngine

Implements

IDisposable

Inherited Members

Game.Dispose() , Game.Dispose(bool) , Game.Exit() , Game.ResetElapsedTime() , Game.SuppressDraw() , Game.RunOneFrame() , Game.Run() , Game.Run(GameRunBehavior) , Game.Tick() , Game.BeginDraw() , Game.EndDraw() , Game.BeginRun() , Game.EndRun() , Game.UnloadContent() , Game.OnExiting(object, ExitingEventArgs) , Game.OnActivated(object, EventArgs) , Game.OnDeactivated(object, EventArgs) , Game.LaunchParameters , Game.Components , Game.InactiveSleepTime , Game.MaxElapsedTime , Game.IsActive , Game.IsMouseVisible , Game.TargetElapsedTime , Game.IsFixedTimeStep , Game.Services , Game.Content , Game.GraphicsDevice , Game.Window , Game.Activated , Game.Deactivated , Game.Disposed , Game.Exiting , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

GameEngine()

```
public GameEngine()
```

Properties

Graphics

The graphics device manager.

```
public GraphicsDeviceManager Graphics { get; set; }
```

Property Value

GraphicsDeviceManager

Root

```
public Container Root { get; set; }
```

Property Value

Container

SpriteBatch

the sprite batch for rendering most the children.

```
public SpriteBatch SpriteBatch { get; set; }
```

Property Value

SpriteBatch

Methods

Draw(GameTime)

Called when the game should draw a frame.

Draws the Microsoft.Xna.Framework.DrawableGameComponent instances attached to this game. Override this to render your game.

```
protected override void Draw(GameTime gameTime)
```

Parameters

`gameTime` GameTime

A Microsoft.Xna.Framework.GameTime instance containing the elapsed time since the last call to Microsoft.Xna.Framework.Game.Draw(Microsoft.Xna.Framework.GameTime) and the total time elapsed since the game started.

Initialize()

Override this to initialize the game and load any needed non-graphical resources.

Initializes attached Microsoft.Xna.Framework.GameComponent instances and calls Microsoft.Xna.Framework.Game.LoadContent().

```
protected override void Initialize()
```

LoadContent()

Override this to load graphical resources required by the game.

```
protected override void LoadContent()
```

Update(GameTime)

Called when the game should update.

Updates the Microsoft.Xna.Framework.GameComponent instances attached to this game. Override this to update your game.

```
protected override void Update(GameTime gameTime)
```

Parameters

`gameTime` GameTime

The elapsed time since the last call to Microsoft.Xna.Framework.Game.Update(Microsoft.Xna.Framework.GameTime).

Interface IRenderable

Namespace: Hallowed.Core

Assembly: Hallowed.dll

```
public interface IRenderable
```

Properties

Parent

```
Container Parent { get; set; }
```

Property Value

Container

Visible

```
bool Visible { get; set; }
```

Property Value

bool

Methods

Draw(SpriteBatch)

```
void Draw(SpriteBatch spriteBatch)
```

Parameters

`spriteBatch` SpriteBatch

Update(GameTime)

```
void Update(GameTime gameTime)
```

Parameters

`gameTime` GameTime

Namespace Hallowed.Graphics

Classes

AnimatedSprite

Sprite

Represents a 2D renderable object that displays a texture in the scene graph. A Sprite is the primary visual element used for rendering images, supporting transformations (position, rotation, scale), color tinting, and orientation flips.

SpriteAnchor

Provides common anchor point presets for sprites. Anchors are normalized (0-1 range) positions within a sprite.

TextureAtlas

The utility class that manage a texture atlas.

TextureWrapper

Wrapper class for allow easy texture manipulation. This class is optional and not required for creating sprites. it however, has a lot of useful methods to help with texture manipulation.

```
var textureWrapper = new TextureWrapper(Content.Load("texture"));
var sprite = new Sprite(textureWrapper.Texture);
var texture2 = Content.Load("texture2");
textureWrapper.blit(texture2, new Rectangle(0, 0, 100, 100), new Rectangle(0, 0,
100, 100);
```

Structs

SpriteAnimationData

Enums

SpriteOrientation

Class AnimatedSprite

Namespace: Hallowed.Graphics

Assembly: Hallowed.dll

```
public class AnimatedSprite : Sprite, IDisposable
```

Inheritance

object ← DisplayObject ← Sprite ← AnimatedSprite

Implements

IDisposable

Inherited Members

Sprite.Texture , Sprite.Color , Sprite.Frame , Sprite.Orientation , Sprite.Anchor , Sprite.Origin , Sprite.GetBounds() , Sprite.Update(GameTime) , Sprite.Draw(SpriteBatch) , Sprite.GetSpriteEffects() , Sprite.Dispose(bool) , Sprite.UpdateOriginFromAnchor() , Sprite.UpdateAnchorFromOrigin() , DisplayObject.IsDisposed , DisplayObject.WorldTransform , DisplayObject.Parent , DisplayObject.Children , DisplayObject.Visible , DisplayObject.Position , DisplayObject.X , DisplayObject.Y , DisplayObject.Scale , DisplayObject.Rotation , DisplayObject.Width , DisplayObject.Height , DisplayObject.GetWorldBounds() , DisplayObject.AddChild(DisplayObject) , DisplayObject.AddChild(IEnumerable<DisplayObject>) , DisplayObject.AddChild(params DisplayObject[]) , DisplayObject.RemoveChild(DisplayObject, bool) , DisplayObject.RemoveChild(IEnumerable<DisplayObject>, bool) , DisplayObject.RemoveChild(bool, params DisplayObject[]) , DisplayObject.MarkDirty() , DisplayObject.UpdateWorldTransform() , DisplayObject.DecomposeMatrix2D(Matrix, out Vector2, out float, out Vector2) , DisplayObject.Dispose() , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

AnimatedSprite()

```
public AnimatedSprite()
```

Properties

Animations

```
public IReadOnlyDictionary<string, SpriteAnimationData> Animations { get; }
```

Property Value

IReadOnlyDictionary<string, SpriteAnimationData>

Methods

AddAnimation(string, SpriteAnimationData)

```
public void AddAnimation(string name, SpriteAnimationData animation)
```

Parameters

`name` string

`animation` SpriteAnimationData

Class Sprite

Namespace: Hallowed.Graphics

Assembly: Hallowed.dll

Represents a 2D renderable object that displays a texture in the scene graph. A Sprite is the primary visual element used for rendering images, supporting transformations (position, rotation, scale), color tinting, and orientation flips.

```
public class Sprite : DisplayObject, IDisposable
```

Inheritance

object ← DisplayObject ← Sprite

Implements

IDisposable

Derived

AnimatedSprite

Inherited Members

DisplayObject.IsDisposed , DisplayObject.WorldTransform , DisplayObject.Parent , DisplayObject.Children , DisplayObject.Visible , DisplayObject.Position , DisplayObject.X , DisplayObject.Y , DisplayObject.Scale , DisplayObject.Rotation , DisplayObject.Width , DisplayObject.Height , DisplayObject.GetWorldBounds() , DisplayObject.AddChild(DisplayObject) , DisplayObject.AddChild(IEnumerable<DisplayObject>) , DisplayObject.AddChild(params DisplayObject[]) , DisplayObject.RemoveChild(DisplayObject, bool) , DisplayObject.RemoveChild(IEnumerable<DisplayObject>, bool) , DisplayObject.RemoveChild(bool, params DisplayObject[]) , DisplayObject.MarkDirty() , DisplayObject.UpdateWorldTransform() , DisplayObject.DecomposeMatrix2D(Matrix, out Vector2, out float, out Vector2) , DisplayObject.Dispose() , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

The Sprite class inherits from DisplayObject, meaning it participates in the world transform hierarchy. The texture is automatically disposed when the sprite is disposed, so it should not be shared across

multiple sprites unless handled externally.

Constructors

Sprite()

```
public Sprite()
```

Sprite(TextureAtlas, string)

```
public Sprite(TextureAtlas textureAtlas, string regionName)
```

Parameters

`textureAtlas` TextureAtlas

`regionName` string

Sprite(Texture2D)

The sprite constructor.

```
public Sprite(Texture2D texture)
```

Parameters

`texture` Texture2D

the sprite texture

Properties

Anchor

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

Property Value

Vector2

Color

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

Property Value

Color

Frame

```
public Rectangle Frame { get; set; }
```

Property Value

Rectangle

Orientation

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

Property Value

SpriteOrientation

Origin

The display object origin.

```
public override Vector2 Origin { get; set; }
```

Property Value

Vector2

Texture

The sprite texture.

```
public virtual Texture2D Texture { get; set; }
```

Property Value

Texture2D

Methods

Dispose(bool)

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

Parameters

`disposing` bool

Draw(SpriteBatch)

draw the object and all of its children.

```
public override void Draw(SpriteBatch spriteBatch)
```

Parameters

`spriteBatch` SpriteBatch

Remarks

if an object or its children visible flag is set to false it wont be draw but still be updated.

GetBounds()

```
public override Rectangle GetBounds()
```

Returns

Rectangle

GetSpriteEffects()

```
protected virtual SpriteEffects GetSpriteEffects()
```

Returns

SpriteEffects

Update(GameTime)

update the object and all of its children.

```
public override void Update(GameTime gameTime)
```

Parameters

`gameTime` GameTime

UpdateAnchorFromOrigin()

```
protected void UpdateAnchorFromOrigin()
```

UpdateOriginFromAnchor()

```
protected void UpdateOriginFromAnchor()
```

Class SpriteAnchor

Namespace: Hallowed.Graphics

Assembly: Hallowed.dll

Provides common anchor point presets for sprites. Anchors are normalized (0-1 range) positions within a sprite.

```
public static class SpriteAnchor
```

Inheritance

object ← SpriteAnchor

Inherited Members

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

Fields

BottomCenter

Bottom-center (0.5, 1) - Useful for characters standing on ground

```
public static readonly Vector2 BottomCenter
```

Field Value

Vector2

BottomLeft

Bottom-left corner (0, 1)

```
public static readonly Vector2 BottomLeft
```

Field Value

Vector2

BottomRight

Bottom-right corner (1, 1)

```
public static readonly Vector2 BottomRight
```

Field Value

Vector2

Center

Center (0.5, 0.5) - Useful for rotation and centering

```
public static readonly Vector2 Center
```

Field Value

Vector2

CenterLeft

Center-left (0, 0.5)

```
public static readonly Vector2 CenterLeft
```

Field Value

Vector2

CenterRight

Center-right (1, 0.5)

```
public static readonly Vector2 CenterRight
```

Field Value

Vector2

TopCenter

Top-center (0.5, 0)

```
public static readonly Vector2 TopCenter
```

Field Value

Vector2

TopLeft

Top-left corner (0, 0) - Default anchor point

```
public static readonly Vector2 TopLeft
```

Field Value

Vector2

TopRight

Top-right corner (1, 0)

```
public static readonly Vector2 TopRight
```

Field Value

Vector2

Struct SpriteAnimationData

Namespace: Hallowed.Graphics

Assembly: Hallowed.dll

```
public struct SpriteAnimationData
```

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

Fields

frameRate

```
public float frameRate
```

Field Value

float

frames

```
public List<Rectangle> frames
```

Field Value

List<Rectangle>

loop

```
public bool loop
```

Field Value

bool

onAnimationEnd

```
public Action onAnimationEnd
```

Field Value

Action

Enum SpriteOrientation

Namespace: Hallowed.Graphics

Assembly: Hallowed.dll

```
public enum SpriteOrientation
```

Fields

`Both = 3`

`FlipHorizontal = 1`

`FlipVertical = 2`

`Normal = 0`

Class TextureAtlas

Namespace: Hallowed.Graphics

Assembly: Hallowed.dll

The utility class that manage a texture atlas.

```
public class TextureAtlas
```

Inheritance

object ← TextureAtlas

Inherited Members

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

Constructors

TextureAtlas(Texture2D)

The texture atlas constructor.

```
public TextureAtlas(Texture2D texture)
```

Parameters

`texture` Texture2D

the atlas texture

Properties

Regions

The atlas regions.

```
public IReadOnlyDictionary<string, Rectangle> Regions { get; }
```

Property Value

IReadOnlyDictionary<string, Rectangle>

Texture

The Atlas texture

```
public Texture2D Texture { get; }
```

Property Value

Texture2D

Methods

AddRegion(string, Rectangle)

Add a region to the atlas.

```
public void AddRegion(string name, Rectangle region)
```

Parameters

`name` string

the region name

`region` Rectangle

the region rect

Exceptions

ArgumentException

GetRegion(string, out Rectangle)

Get the Atlas region by name.

```
public bool GetRegion(string name, out Rectangle region)
```

Parameters

`name` string

the region name

`region` Rectangle

the region rectangle to copy the value to

Returns

bool

return the textures region coordinates

Parse(string)

Parse a json file to create the atlas such as Texture Packer.

```
public void Parse(string json)
```

Parameters

`json` string

Class TextureWrapper

Namespace: Hallowed.Graphics

Assembly: Hallowed.dll

Wrapper class for allow easy texture manipulation. This class is optional and not required for creating sprites. it however, has a lot of useful methods to help with texture manipulation.

```
var textureWrapper = new TextureWrapper(Content.Load("texture"));
var sprite = new Sprite(textureWrapper.Texture);
var texture2 = Content.Load("texture2");
textureWrapper.blit(texture2, new Rectangle(0, 0, 100, 100), new Rectangle(0, 0, 100, 100);
```

```
public class TextureWrapper : IDisposable
```

Inheritance

object ← TextureWrapper

Implements

IDisposable

Inherited Members

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

Constructors

TextureWrapper(GraphicsDevice, Texture2D)

The texture wrapper constructor.

```
public TextureWrapper(GraphicsDevice device, Texture2D texture)
```

Parameters

`device` GraphicsDevice

The graphic device

`texture` Texture2D

the texture to wrap

TextureWrapper(GraphicsDevice, int, int)

The texture wrapper constructor.

```
public TextureWrapper(GraphicsDevice device, int width, int height)
```

Parameters

`device` GraphicsDevice

The graphic device

`width` int

the texture wrapper width (automatically adjust when assigning a texture)

`height` int

the texture wrapper height (automatically adjust when assigning a texture)

Properties

Height

the texture height.

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

Property Value

int

Texture

The texture

```
public Texture2D Texture { get; set; }
```

Property Value

Texture2D

Width

the texture width.

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

Property Value

int

Methods

Blit(Texture2D, Rectangle, Rectangle)

copy a region of one texture to another.

```
public void Blit(Texture2D source, Rectangle sourceRect, Rectangle destRect)
```

Parameters

`source` Texture2D

`sourceRect` Rectangle

`destRect` Rectangle

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

GetFrameRanges(Vector2, int, int)

return an array of frames Rectangle Data for animation based on a range of frames.

```
public List<Rectangle> GetFrameRanges(Vector2 frameSize, int startIndex, int endIndex)
```

Parameters

`frameSize` Vector2

the frame size

`startIndex` int

`endIndex` int

Returns

List<Rectangle>

GetFrameRanges(int, int, int, int)

returns an array of frames Rectangle Data for animation based on a range of frames.

```
public List<Rectangle> GetFrameRanges(int frameWidth, int frameHeight, int startIndex,
int endIndex)
```

Parameters

`frameWidth` int

`frameHeight` int

`startIndex` int

`endIndex` int

Returns

List<Rectangle>

Namespace Hallowed.Rendering

Classes

Camera2D

Class Camera2D

Namespace: Hallowed.Rendering

Assembly: Hallowed.dll

```
public class Camera2D
```

Inheritance

object ← Camera2D

Inherited Members

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

Constructors

Camera2D(Viewport)

```
public Camera2D(Viewport viewport)
```

Parameters

`viewport` Viewport

Namespace Hallowed.Rendering.Utilities

Classes

PingPongBuffer

Provides a double-buffered render target system ("ping-pong" buffer) for iterative texture operations and post-processing in MonoGame.

Class PingPongBuffer

Namespace: Hallowed.Rendering.Utilities

Assembly: Hallowed.dll

Provides a double-buffered render target system ("ping-pong" buffer) for iterative texture operations and post-processing in MonoGame.

```
public class PingPongBuffer : IDisposable
```

Inheritance

object ← PingPongBuffer

Implements

IDisposable

Inherited Members

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

Examples

Example usage:

```
var buffer = new PingPongBuffer(graphicsDevice, 1024, 1024);

// Draw to the current target
graphicsDevice.SetRenderTarget(buffer.Target);
spriteBatch.Begin();
spriteBatch.Draw(texture1, Vector2.Zero, Color.White);
spriteBatch.Draw(texture2, new Vector2(512, 0), Color.White);
spriteBatch.End();
graphicsDevice.SetRenderTarget(null);

// Swap read/write targets for the next iteration
buffer.Swap();

// Use the final composed texture
Texture2D result = buffer.Source;
```

Remarks

The ping-pong technique alternates between two render targets — one for reading (Source) and one for writing (Target) — avoiding GPU read/write conflicts. This allows chained rendering passes, feedback effects, and texture blitting without reallocating textures or copying data between frames.

Constructors

PingPongBuffer(GraphicsDevice, int, int, SurfaceFormat)

The ping pong buffer constructor.

```
public PingPongBuffer(GraphicsDevice device, int width, int height, SurfaceFormat format
= SurfaceFormat.Color)
```

Parameters

`device` GraphicsDevice

`width` int

`height` int

`format` SurfaceFormat

Properties

HasSource

Check if the buffer has been initialized.

```
public bool HasSource { get; }
```

Property Value

bool

Height

the buffer height.

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

Property Value

int

Source

Gets the current source texture (read from this).

```
public RenderTarget2D Source { get; }
```

Property Value

RenderTarget2D

Target

Gets the current target texture (write to this).

```
public RenderTarget2D Target { get; }
```

Property Value

RenderTarget2D

Width

the buffer width.

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

Property Value

int

Methods

Assign(Texture2D)

First assign a texture to the source and target buffers.

```
public void Assign(Texture2D texture)
```

Parameters

`texture` Texture2D

Clear()

Clear the current source to transparent.

```
public void Clear()
```

ClearAll()

Clear both buffers.

```
public void ClearAll()
```

Dispose()

Dispose the buffer.

```
public void Dispose()
```

Resize(int, int, SurfaceFormat)

Resize both buffers.

```
public void Resize(int width, int height, SurfaceFormat format = SurfaceFormat.Color)
```

Parameters

`width` int

`height` int

`format` SurfaceFormat

Swap()

Swap source and target buffers.

```
public void Swap()
```

See Also

Ping-pong scheme (Wikipedia)

Namespace Hallowed.Scenes

Classes

SceneBase

SceneManager

Class SceneBase

Namespace: Hallowed.Scenes

Assembly: Hallowed.dll

```
public abstract class SceneBase : Container, IDisposable
```

Inheritance

object ← DisplayObject ← Container ← SceneBase

Implements

IDisposable

Inherited Members

Container.GetBounds() , DisplayObject.IsDisposed , DisplayObject.WorldTransform , DisplayObject.Parent , DisplayObject.Children , DisplayObject.Visible , DisplayObject.Position , DisplayObject.X , DisplayObject.Y , DisplayObject.Scale , DisplayObject.Rotation , DisplayObject.Origin , DisplayObject.Width , DisplayObject.Height , DisplayObject.GetWorldBounds() , DisplayObject.Update(GameTime) , DisplayObject.Draw(SpriteBatch) , DisplayObject.AddChild(DisplayObject) , DisplayObject.AddChild(IEnumerable<DisplayObject>) , DisplayObject.AddChild(params DisplayObject[]) , DisplayObject.RemoveChild(DisplayObject, bool) , DisplayObject.RemoveChild(IEnumerable<DisplayObject>, bool) , DisplayObject.RemoveChild(bool, params DisplayObject[]) , DisplayObject.MarkDirty() , DisplayObject.UpdateWorldTransform() , DisplayObject.DecomposeMatrix2D(Matrix, out Vector2, out float, out Vector2) , DisplayObject.Dispose() , DisplayObject.Dispose(bool) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Game

```
protected Game Game { get; set; }
```

Property Value

Game

Graphics

```
protected GraphicsDeviceManager Graphics { get; set; }
```

Property Value

GraphicsDeviceManager

SpriteBatch

```
protected SpriteBatch SpriteBatch { get; set; }
```

Property Value

SpriteBatch

Methods

Initialize()

```
public void Initialize()
```

LoadContent()

```
public void LoadContent()
```

Class SceneManager

Namespace: Hallowed.Scenes

Assembly: Hallowed.dll

```
public class SceneManager : IGameComponent
```

Inheritance

object ← SceneManager

Implements

IGameComponent

Inherited Members

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

Constructors

SceneManager(GameEngine)

```
public SceneManager(GameEngine game)
```

Parameters

`game` GameEngine

Properties

Scene

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

Property Value

SceneBase

Methods

Goto(SceneBase)

```
public void Goto(SceneBase scene)
```

Parameters

`scene` SceneBase

Initialize()

Initializes the component. Used to load non-graphical resources.

```
public void Initialize()
```