GXPEngine

Generated by Doxygen 1.9.2

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 GXPEngine Namespace Reference

Classes

- class AnimationSprite

  Animated Sprite. Has all the functionality of a regular sprite, but supports multiple animation frames/subimages.
- class AnimSprite
- class BlendMode

  Defines different BlendModes. Only six present now, but you can add your own.
- class BufferRenderer

  A helper class for SpriteBatches, and possibly other complex objects or collections with larger vertex and uv lists.
- class Camera

  A Camera gameobject, that owns a rectangular render window, and determines the focal point, rotation and scale of what's rendered in that window. (Don't forget to add this as child somewhere in the hierarchy.)
- class Canvas

  The Canvas object can be used for drawing 2D visuals at runtime.
- class CollisionManager
- class EasyDraw

  Creates an easy-to-use layer on top of .NET's System.Drawing methods. The API is inspired by Processing: internal states are maintained for font, fill/stroke color, etc., and everything works with simple methods that have many overloads.
- class Game

  The Game class represents the Game window. Only a single instance of this class is allowed.
- class GameObject

  GameObject is the base class for all display objects.
- class Gizmos

  This class can be used to easily draw line based shapes (like arrows and rectangles), mostly for debug purposes (it is not made for efficiency). For each draw call, shapes are drawn for one frame only, after rendering all sprites. See the DrawLine method for more information.
- class HierarchyManager

  If you are getting strange bugs because you are calling Destroy during the Update loop, you can use this class to do this more cleanly: when using HierarchyManager.Instance.LateDestroy, all these hierarchy changes will be made after the update loop is finished. You can also use HierarchyManager.Instance.LateAdd to add a game object after the update loop is finished. Similarly, you can use HierarchyManager.Instance.LateCall to postpone a certain method call until after the update loop.
- class Input

The Input class contains functions for reading keys and mouse

- class Key

  Contains key definitions for usage with Input.GetKey and Input.GetKeyDown. 0.0.5: Updated keylist, courtesy of Alexandru Tâlvan

- class **Mathf**

  Contains several functions for doing floating point Math

- class MouseHandler
- class Pivot

  This is an 'empty' GameObject. You can use it as a container for other sprites (parent).

- class Settings

  Static class that contains various settings, such as screen resolution and player controls. In your Main method, you can Call the Settings.Load() method to initialize these settings from a text file (typically settings.txt, which should be present in the bin/Debug and/or bin/Release folder).

- class Sound

  The Sound Class represents a Sound resource in memory You can load .mp3, .ogg or .wav

- class SoundChannel

  This class represents a sound channel on the soundcard.

- class Sprite

  The Sprite class holds 2D images that can be used as objects in your game.

- class SpriteBatch

  A SpriteBatch is a GameObject that can be used to render many sprites efficiently, and change the color, alpha and blend mode of all of those sprites simultaneously. Usage: Add any number of sprites as child to a sprite batch, and then call the Freeze method. Note that this will destroy the individual sprites, so there won't be colliders for them anymore, and the position and rotation of individual sprites cannot be changed anymore.

- class Time

  Contains various time related functions.

- class Transformable

  The Transformable class contains all positional data of GameObjects.

- class **Utils**

  The Utils class contains a number of useful functions.

- class Window

  A class that can be used to create "sub windows" (e.g. mini-map, splitscreen, etc). This is not a gameobject. Instead, subscribe the RenderWindow method to the main game's OnAfterRender event.

Enumerations

- enum **MouseEventType** {
  MouseDown , **MouseDownOnTarget** , **MouseUp** , **MouseUpOnTarget** ,
  MouseMove , **MouseMoveOnTarget** , **MouseOverTarget** , **MouseOffTarget** ,
  MouseClick }
- enum **CenterMode** { **Min** , **Center** , **Max** }

4.2 GXPEngine.Core Namespace Reference

Classes

- class BoxCollider
- class Collider
- class Collision

  A class that contains info about collisions, such as returned by the MoveUntilCollision method.

- class FMOD
- class FMODSoundSystem
- class GLContext
- struct Rectangle
- class **Soloud**
- class SoloudSoundSystem
- class SoundSystem
- class Texture2D
- struct Vector2
- class WindowSize

4.3 GXPEngine.Managers Namespace Reference

Classes

- class UpdateManager

4.4 GXPEngine.OpenGL Namespace Reference

Classes

- class GL

4.5 TiledMapParser Namespace Reference

Classes

- class Data
- class Image
- class ImageLayer
- class Layer
- class Map
- class MapParser

  Call the method MapParser.ReadMap, with as argument a Tiled file exported as xml (file extension: .tmx), to get an object of type Map. This object, together with its nested objects, contains most of the information contained in the Tiled file.
- class ObjectGroup
- class Property
- class PropertyContainer
- class PropertyList
- class Text
- class TiledLoader

  A class for automatically creating GXPEngine sprites from Tiled files.
- class TiledObject
- class TiledUtils
- class TileSet

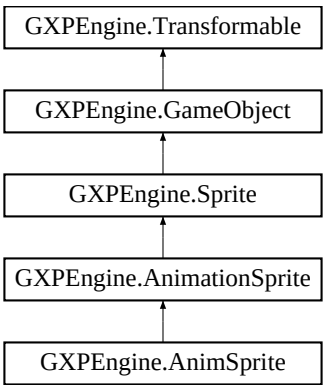
Chapter 5

Class Documentation

5.1 GXPEngine.AnimationSprite Class Reference

Animated Sprite. Has all the functionality of a regular sprite, but supports multiple animation frames/subimages.

Inheritance diagram for GXPEngine.AnimationSprite:



Public Member Functions

- AnimationSprite (string filename, int cols, int rows, int frames=-1, bool keepInCache=false, bool addCollider=true)

  Initializes a new instance of the GXPEngine.AnimSprite class.
- AnimationSprite (System.Drawing.Bitmap bitmap, int cols, int rows, int frames=-1, bool addCollider=true)

  Initializes a new instance of the GXPEngine.AnimSprite class.
- void SetFrame (int frame)

  Sets the current animation frame. Frame should be in range 0...frameCount-1
- void **NextFrame** ()

  Goes to the next frame. At the end of the animation, it jumps back to the start frame. (It loops)
- void SetCycle (int startFrame, int numFrames=1, byte animationDelay=255, bool switchFrame=true)

  Sets the animation cycle: the NextFrame method will go from startFrame to startFrame + numFrames - 1, and then jump back to the startFrame. If animationDelay is smaller than 255, then the Animate method will stay on the same animation frame for this many game frames (so larger value = slower animation). If switchFrame is true then the currentFrame will be set in the given range, if it isn't already.
- void **Animate** (float deltaFrameTime=1)

If the current animation frame has been shown for [_animationDelay] game frames, this jumps to the next animation frame in the cycle. Call this method every update, and use it in combination with SetCycle to create a timed sprite animation. Smaller values for deltaFrameTime slow down the animation.

- void **AnimateFixed** ()

  Plays the animation cycle given by SetCycle at constant speed, independent of current FPS. The number of animation frames per second is [game.targetFps / _animationDelay].

Protected Member Functions

- void **initializeAnimFrames** (int cols, int rows, int frames=-1)
- override void setUVs ()

Protected Attributes

- float **_frameWidth**
- float **_frameHeight**
- int **_rows**
- int **_cols**
- int **_frames**
- int **_currentFrame**
- int **_startFrame** = 0
- byte **_animationDelay** = 1

Properties

- override int **width** `[getset]`

  Gets or sets the sprite's width in pixels.
- override int **height** `[getset]`

  Gets or sets the sprite's height in pixels.
- int **currentFrame** `[getset]`

  Returns the current frame.
- int **frameCount** `[get]`

  Returns the number of frames in this animation.

Additional Inherited Members

5.1.1 Detailed Description

Animated Sprite. Has all the functionality of a regular sprite, but supports multiple animation frames/subimages.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AnimationSprite() `[1/2]`

```
GXPEngine.AnimationSprite.AnimationSprite (
            string filename,
            int cols,
            int rows,
            int frames = -1,
            bool keepInCache = false,
            bool addCollider = true )
```

Initializes a new instance of the GXPEngine.AnimSprite class.

Parameters

filename	The name of the file to be loaded. Files are cached internally. Texture sizes should be a power of two: 1, 2, 4, 8, 16, 32, 64 etc. The width and height don't need to be the same. If you want to load transparent sprites, use .PNG with transparency.
cols	Number of columns in the animation.
rows	Number of rows in the animation.
frames	Optionally, indicate a number of frames. When left blank, defaults to width∗height.
keepInCache	If `true`, the sprite's texture will be kept in memory for the entire lifetime of the game. This takes up more memory, but removes load times.
addCollider	If `true`, this sprite will have a collider that will be added to the collision manager.

5.1.2.2 AnimationSprite() [2/2]

```
GXPEngine.AnimationSprite.AnimationSprite (
            System.Drawing.Bitmap bitmap,
            int cols,
            int rows,
            int frames = -1,
            bool addCollider = true )
```

Initializes a new instance of the GXPEngine.AnimSprite class.

Parameters

bitmap	The Bitmap object to be used to create the sprite. Texture sizes should be a power of two: 1, 2, 4, 8, 16, 32, 64 etc. The width and height don't need to be the same. If you want to load transparent sprites, use .PNG with transparency.
cols	Number of columns in the animation.
rows	Number of rows in the animation.
frames	Optionally, indicate a number of frames. When left blank, defaults to width∗height.
addCollider	If `true`, this sprite will have a collider that will be added to the collision manager.

5.1.3 Member Function Documentation

5.1.3.1 SetCycle()

```
void GXPEngine.AnimationSprite.SetCycle (
            int startFrame,
            int numFrames = 1,
```

```
            byte animationDelay = 255,
            bool switchFrame = true )
```

Sets the animation cycle: the NextFrame method will go from startFrame to startFrame + numFrames - 1, and then jump back to the startFrame. If animationDelay is smaller than 255, then the Animate method will stay on the same animation frame for this many game frames (so larger value = slower animation). If switchFrame is true then the currentFrame will be set in the given range, if it isn't already.

Parameters

startFrame	The first frame of the animation cycle
numFrames	The number of frames in the animation cycle
animationDelay	The number of game frames that the same animation frame should be shown, when calling Animate()
switchFrame	If true, then the currentFrame will be set in the given range, if it isn't already.

5.1.3.2 SetFrame()

```
void GXPEngine.AnimationSprite.SetFrame (
            int frame )
```

Sets the current animation frame. Frame should be in range 0...frameCount-1

Parameters

frame	Frame.

5.1.3.3 setUVs()

```
override void GXPEngine.AnimationSprite.setUVs ( )  [protected], [virtual]
```

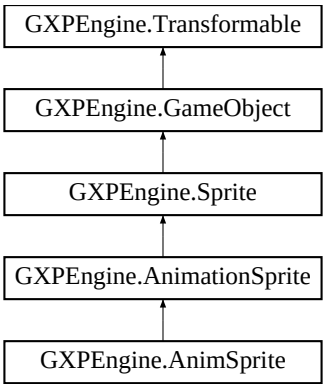
Reimplemented from GXPEngine.Sprite.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AnimationSprite.cs

5.2 GXPEngine.AnimSprite Class Reference

Inheritance diagram for GXPEngine.AnimSprite:



Public Member Functions

- **AnimSprite** (string filename, int cols, int rows, int frames=-1)

Additional Inherited Members

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AnimationSprite.cs

5.3 GXPEngine.BlendMode Class Reference

Defines different BlendModes. Only six present now, but you can add your own.

Public Member Functions

- delegate void **Action** ()
- **BlendMode** (string pLabel, Action pEnable)
- override string **ToString** ()

Public Attributes

- readonly Action **enable**

  This should point to an anonymous function updating the blendfunc
- readonly string **label**

  A label for this blendmode

Static Public Attributes

- static readonly BlendMode NORMAL

  The traditional and default way of blending. (newColor = spriteColor ∗ spriteAlpha + oldColor ∗ (1-spriteAlpha))
- static readonly BlendMode PREMULTIPLIED

  The correct way of doing blending, which however requires preparing your sprites for this (non default). (newColor = spriteColor ∗ 1 + oldColor ∗ (1-spriteAlpha))
- static readonly BlendMode MULTIPLY

  Multiplying colors - use this for darkening. (newColor = spriteColor ∗ oldColor + oldColor ∗ 0)
- static readonly BlendMode LIGHTING

  Brightening existing colors - this mode can be used for lighting effects. (newColor = spriteColor ∗ oldColor + oldColor ∗ 1)
- static readonly BlendMode ADDITIVE

  Adding colors - use this e.g. for "volumetric" lighting effects. (newColor = spriteColor ∗ 1 + oldColor ∗ 1)
- static readonly BlendMode FILLEMPTY

  This mode can be used to fill in empty screen parts (e.g. drawing a background after adding lights to the foreground). (newColor = spriteColor ∗ (1-oldColorAlpha) + oldColor ∗ oldColorAlpha)

5.3.1 Detailed Description

Defines different BlendModes. Only six present now, but you can add your own.

5.3.2 Member Data Documentation

5.3.2.1 ADDITIVE

```
readonly BlendMode GXPEngine.BlendMode.ADDITIVE  [static]
```

Initial value:

```
= new BlendMode(
            "Additive", () => { GL.BlendFunc(GL.ONE, GL.ONE); }
        )
```

Adding colors - use this e.g. for "volumetric" lighting effects. (newColor = spriteColor ∗ 1 + oldColor ∗ 1)

5.3.2.2 FILLEMPTY

```
readonly BlendMode GXPEngine.BlendMode.FILLEMPTY  [static]
```

Initial value:

```
= new BlendMode(
            "Fill", () => { GL.BlendFunc(GL.ONE_MINUS_DST_ALPHA, GL.DST_ALPHA); }
        )
```

This mode can be used to fill in empty screen parts (e.g. drawing a background after adding lights to the foreground). (newColor = spriteColor ∗ (1-oldColorAlpha) + oldColor ∗ oldColorAlpha)

5.3.2.3 LIGHTING

```
readonly BlendMode GXPEngine.BlendMode.LIGHTING  [static]
```

Initial value:

```
= new BlendMode(
            "Lighting", () => { GL.BlendFunc(GL.DST_COLOR, GL.ONE); }
        )
```

Brightening existing colors - this mode can be used for lighting effects. (newColor = spriteColor ∗ oldColor + oldColor ∗ 1)

5.3.2.4 MULTIPLY

```
readonly BlendMode GXPEngine.BlendMode.MULTIPLY  [static]
```

Initial value:

```
= new BlendMode (
            "Multiply", () => { GL.BlendFunc(GL.DST_COLOR, GL.ZERO);    }
        )
```

Multiplying colors - use this for darkening. (newColor = spriteColor ∗ oldColor + oldColor ∗ 0)

5.3.2.5 NORMAL

```
readonly BlendMode GXPEngine.BlendMode.NORMAL  [static]
```

Initial value:

```
= new BlendMode (
            "Normal", () => {   GL.BlendFunc(GL.SRC_ALPHA, GL.ONE_MINUS_SRC_ALPHA); }
        )
```

The traditional and default way of blending. (newColor = spriteColor ∗ spriteAlpha + oldColor ∗ (1-spriteAlpha))

5.3.2.6 PREMULTIPLIED

```
readonly BlendMode GXPEngine.BlendMode.PREMULTIPLIED  [static]
```

Initial value:

```
= new BlendMode(
            "Premultiplied", () => { GL.BlendFunc(GL.ONE, GL.ONE_MINUS_SRC_ALPHA); }
        )
```

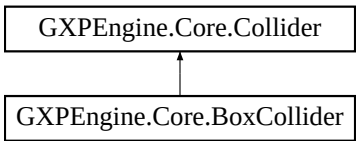
The correct way of doing blending, which however requires preparing your sprites for this (non default). (newColor = spriteColor ∗ 1 + oldColor ∗ (1-spriteAlpha))

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/BlendMode.cs

5.4 GXPEngine.Core.BoxCollider Class Reference

Inheritance diagram for GXPEngine.Core.BoxCollider:



Public Member Functions

- **BoxCollider** (Sprite owner)
- override bool HitTest (Collider other)

  Returns `true` if this collider is currently overlapping with the collider other.
- override bool HitTestPoint (float x, float y)

  Returns `true` if this collider is currently overlapping with the point x,y.
- override float TimeOfImpact (Collider other, float vx, float vy, out Vector2 normal)

  If this collider would collide with collider other after moving by vx,vy, then this method returns the time of impact of the collision, which is a number between 0 (=immediate collision, or already overlapping) and 1 (=collision after moving exactly by vx,vy). Otherwise, a number larger than 1 (e.g. float.MaxValue) is returned. In addition, the collision normal is returned, in case of a collision.
- override Collision GetCollisionInfo (Collider other)

  If this collider and the collider other are overlapping, this method returns useful collision info such as the collision normal, the point of impact, and the penetration depth, contained in a Collision object (the time of impact field will always be zero).

Additional Inherited Members

5.4.1 Member Function Documentation

5.4.1.1 GetCollisionInfo()

```
override Collision GXPEngine.Core.BoxCollider.GetCollisionInfo (
            Collider other )  [virtual]
```

If this collider and the collider other are overlapping, this method returns useful collision info such as the collision normal, the point of impact, and the penetration depth, contained in a Collision object (the time of impact field will always be zero).

If they are not overlapping, this method returns null.

Reimplemented from GXPEngine.Core.Collider.

5.4.1.2 HitTest()

```
override bool GXPEngine.Core.BoxCollider.HitTest (
            Collider other )  [virtual]
```

Returns `true` if this collider is currently overlapping with the collider other.

Reimplemented from GXPEngine.Core.Collider.

5.4.1.3 HitTestPoint()

```
override bool GXPEngine.Core.BoxCollider.HitTestPoint (
            float x,
            float y )  [virtual]
```

Returns `true` if this collider is currently overlapping with the point x,y.

Reimplemented from GXPEngine.Core.Collider.

5.4.1.4 TimeOfImpact()

```
override float GXPEngine.Core.BoxCollider.TimeOfImpact (
            Collider other,
            float vx,
            float vy,
            out Vector2 normal )  [virtual]
```

If this collider would collide with collider other after moving by vx,vy, then this method returns the time of impact of the collision, which is a number between 0 (=immediate collision, or already overlapping) and 1 (=collision after moving exactly by vx,vy). Otherwise, a number larger than 1 (e.g. float.MaxValue) is returned. In addition, the collision normal is returned, in case of a collision.

Returns

The time of impact.

Parameters

other	Another collider.
vx	x velocity or translation amount.
vy	y velocity or translation amount.
normal	The collision normal.

Reimplemented from GXPEngine.Core.Collider.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/BoxCollider.cs

5.5 GXPEngine.BufferRenderer Class Reference

A helper class for SpriteBatches, and possibly other complex objects or collections with larger vertex and uv lists.

Public Member Functions

- **BufferRenderer** (Texture2D texture)
- void **AddVert** (float x, float y)
- void **AddUv** (float u, float v)
- void **CreateBuffers** ()
- void **DrawBuffers** (GLContext glContext)
- void **Dispose** ()

Protected Attributes

- float[ ] **verts**
- float[ ] **uvs**
- int **numberOfVertices**

5.5.1 Detailed Description

A helper class for SpriteBatches, and possibly other complex objects or collections with larger vertex and uv lists.

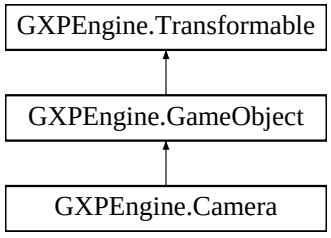
The documentation for this class was generated from the following file:

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/SpriteBatch.cs

5.6 GXPEngine.Camera Class Reference

A Camera gameobject, that owns a rectangular render window, and determines the focal point, rotation and scale of what's rendered in that window. (Don't forget to add this as child somewhere in the hierarchy.)

Inheritance diagram for GXPEngine.Camera:



Public Member Functions

- Camera (int windowX, int windowY, int windowWidth, int windowHeight)

  Creates a camera game object and a sub window to render to. Add this camera as child to the object you want to follow, or update its coordinates directly in an update method. The scale of the camera determines the "zoom factor" (High scale = zoom out)
- bool **ScreenPointInWindow** (int screenX, int screenY)

  Returns whether a screen point (such as received from e.g. Input.mouseX/Y) is in the camera's window
- Vector2 ScreenPointToGlobal (int screenX, int screenY)

  Translates a point from camera space to global space, taking the camera transform and window position into account. The input should be a point in screen space (coordinates between 0 and game.width/height), that is covered by the camera window (use ScreenPointInWindow to check). You can combine this for instance with HitTestPoint and Input.mouseX/Y to check whether the mouse hits a sprite that is shown in the camera's window.

Protected Member Functions

- override void OnDestroy ()

  Subclasses can implement this method to clean up resources once on destruction. Will be called by the engine when the game object is destroyed.

Properties

- Window **RenderTarget** `[get]`

Additional Inherited Members

5.6.1 Detailed Description

A Camera gameobject, that owns a rectangular render window, and determines the focal point, rotation and scale of what's rendered in that window. (Don't forget to add this as child somewhere in the hierarchy.)

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Camera()

```
GXPEngine.Camera.Camera (
            int windowX,
            int windowY,
            int windowWidth,
            int windowHeight )
```

Creates a camera game object and a sub window to render to. Add this camera as child to the object you want to follow, or update its coordinates directly in an update method. The scale of the camera determines the "zoom factor" (High scale = zoom out)

Parameters

windowX	Left x coordinate of the render window.
windowY	Top y coordinate of the render window.
windowWidth	Width of the render window.
windowHeight	Height of the render window.

5.6.3 Member Function Documentation

5.6.3.1 OnDestroy()

```
override void GXPEngine.Camera.OnDestroy ( )  [protected], [virtual]
```

Subclasses can implement this method to clean up resources once on destruction. Will be called by the engine when the game object is destroyed.

Reimplemented from GXPEngine.GameObject.

5.6.3.2 ScreenPointToGlobal()

```
Vector2 GXPEngine.Camera.ScreenPointToGlobal (
            int screenX,
            int screenY )
```

Translates a point from camera space to global space, taking the camera transform and window position into account. The input should be a point in screen space (coordinates between 0 and game.width/height), that is covered by the camera window (use ScreenPointInWindow to check). You can combine this for instance with HitTestPoint and Input.mouseX/Y to check whether the mouse hits a sprite that is shown in the camera's window.

Parameters

screenX	The x coordinate of a point in screen space (like Input.mouseX)
screenY	The y coordinate of a point in screen space (like Input.mouseY)

Returns

Global space coordinates (to be used e.g. with HitTestPoint)

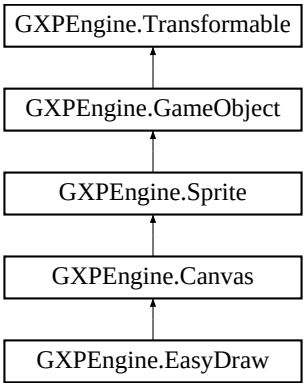
The documentation for this class was generated from the following file:

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/Camera.cs

5.7 GXPEngine.Canvas Class Reference

The Canvas object can be used for drawing 2D visuals at runtime.

Inheritance diagram for GXPEngine.Canvas:



Public Member Functions

- Canvas (int width, int height, bool addCollider=true)

  Initializes a new instance of the Canvas class. It is a regular GameObject that can be added to any displaylist via commands such as AddChild. It contains a `System.Drawing.Graphics` *component.*
- **Canvas** (System.Drawing.Bitmap bitmap, bool addCollider=true)
- **Canvas** (string filename, bool addCollider=true)
- void **DrawSprite** (Sprite sprite)

Protected Member Functions

- override void RenderSelf (GLContext glContext)

Protected Attributes

- Graphics **_graphics**
- bool **_invalidate** = false

Properties

- Graphics **graphics** `[get]`

  Returns the graphics component. This interface provides tools to draw on the sprite. See: `System.Drawing.Graphics`

Additional Inherited Members

5.7.1 Detailed Description

The Canvas object can be used for drawing 2D visuals at runtime.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Canvas()

```
GXPEngine.Canvas.Canvas (
            int width,
            int height,
            bool addCollider = true )
```

Initializes a new instance of the Canvas class. It is a regular GameObject that can be added to any displaylist via commands such as AddChild. It contains a `System.Drawing.Graphics` component.

Parameters

width	Width of the canvas in pixels.
height	Height of the canvas in pixels.

5.7.3 Member Function Documentation

5.7.3.1 RenderSelf()

```
override void GXPEngine.Canvas.RenderSelf (
            GLContext glContext )  [protected], [virtual]
```

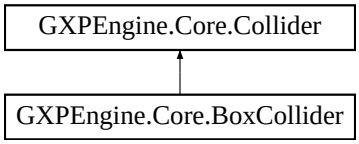
Reimplemented from GXPEngine.GameObject.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/Canvas.cs

5.8 GXPEngine.Core.Collider Class Reference

Inheritance diagram for GXPEngine.Core.Collider:



Public Member Functions

- virtual bool HitTest (Collider other)

  Returns `true` if this collider is currently overlapping with the collider other.
- virtual bool HitTestPoint (float x, float y)

  Returns `true` if this collider is currently overlapping with the point x,y.
- virtual float TimeOfImpact (Collider other, float vx, float vy, out Vector2 normal)

  If this collider would collide with collider other after moving by vx,vy, then this method returns the time of impact of the collision, which is a number between 0 (=immediate collision, or already overlapping) and 1 (=collision after moving exactly by vx,vy). Otherwise, a number larger than 1 (e.g. float.MaxValue) is returned. In addition, the collision normal is returned, in case of a collision.
- virtual Collision GetCollisionInfo (Collider other)

  If this collider and the collider other are overlapping, this method returns useful collision info such as the collision normal, the point of impact, and the penetration depth, contained in a Collision object (the time of impact field will always be zero).

Properties

- bool **isTrigger** `[getset]`

5.8.1 Member Function Documentation

5.8.1.1 GetCollisionInfo()

```
virtual Collision GXPEngine.Core.Collider.GetCollisionInfo (
            Collider other )  [virtual]
```

If this collider and the collider other are overlapping, this method returns useful collision info such as the collision normal, the point of impact, and the penetration depth, contained in a Collision object (the time of impact field will always be zero).

If they are not overlapping, this method returns null.

Reimplemented in GXPEngine.Core.BoxCollider.

5.8.1.2 HitTest()

```
virtual bool GXPEngine.Core.Collider.HitTest (
            Collider other )  [virtual]
```

Returns `true` if this collider is currently overlapping with the collider other.

Reimplemented in GXPEngine.Core.BoxCollider.

5.8.1.3 HitTestPoint()

```
virtual bool GXPEngine.Core.Collider.HitTestPoint (
            float x,
            float y )  [virtual]
```

Returns `true` if this collider is currently overlapping with the point x,y.

Reimplemented in GXPEngine.Core.BoxCollider.

5.8.1.4 TimeOfImpact()

```
virtual float GXPEngine.Core.Collider.TimeOfImpact (
            Collider other,
            float vx,
            float vy,
            out Vector2 normal )  [virtual]
```

If this collider would collide with collider other after moving by vx,vy, then this method returns the time of impact of the collision, which is a number between 0 (=immediate collision, or already overlapping) and 1 (=collision after moving exactly by vx,vy). Otherwise, a number larger than 1 (e.g. float.MaxValue) is returned. In addition, the collision normal is returned, in case of a collision.

Returns

The time of impact.

Parameters

other	Another collider.
vx	x velocity or translation amount.
vy	y velocity or translation amount.
normal	The collision normal.

Reimplemented in GXPEngine.Core.BoxCollider.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/Collider.cs

5.9 GXPEngine.Core.Collision Class Reference

A class that contains info about collisions, such as returned by the MoveUntilCollision method.

Public Member Functions

- **Collision** (GameObject pSelf, GameObject pOther, Vector2 pNormal, Vector2 pPoint, float pTimeOfImpact, float pPenetrationDepth)
- **Collision** (GameObject pSelf, GameObject pOther, Vector2 pNormal, float pTimeOfImpact)
- **Collision** (GameObject pSelf, GameObject pOther, Vector2 pNormal, Vector2 pPoint, float pPenetration↩ Depth)

Public Attributes

- GameObject **self**
- GameObject **other**
- Vector2 **normal**
- Vector2 **point**
- float **timeOfImpact**
- float **penetrationDepth**

5.9.1 Detailed Description

A class that contains info about collisions, such as returned by the MoveUntilCollision method.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/Collision.cs

5.10 GXPEngine.CollisionManager Class Reference

Public Member Functions

- void **Step** ()
- GameObject[ ] **GetCurrentCollisions** (GameObject gameObject, bool includeTriggers=true, bool include↩ Solid=true)
- void **Add** (GameObject gameObject)
- void **Remove** (GameObject gameObject)
- string **GetDiagnostics** ()

Static Public Attributes

- static bool **SafeCollisionLoop** =true

  Set this to false if you want to be able to remove game objects from the game during OnCollision (=the old, unsafe default behavior).
- static bool **TriggersOnlyOnCollision** = false

  Set this to true if you only want to include trigger colliders in OnCollision (=more efficient).

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Managers/CollisionManager.cs

5.11 TiledMapParser.Data Class Reference

Public Member Functions

- override string **ToString** ()

Public Attributes

- string **Encoding**
- string **innerXML**

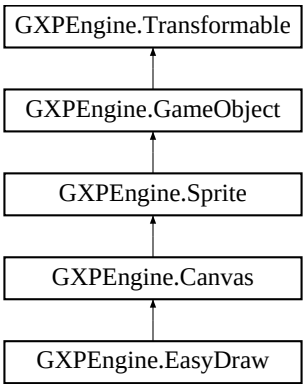
The documentation for this class was generated from the following file:

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.12 GXPEngine.EasyDraw Class Reference

Creates an easy-to-use layer on top of .NET's System.Drawing methods. The API is inspired by Processing↩ : internal states are maintained for font, fill/stroke color, etc., and everything works with simple methods that have many overloads.

Inheritance diagram for GXPEngine.EasyDraw:



Public Member Functions

- EasyDraw (int width, int height, bool addCollider=true)

  Creates a new EasyDraw canvas with the given width and height in pixels.
- EasyDraw (System.Drawing.Bitmap bitmap, bool addCollider=true)

  Creates a new EasyDraw canvas from a given bitmap.
- EasyDraw (string filename, bool addCollider=true)

  Creates a new EasyDraw canvas from a file that contains a sprite (png, jpg).
- void TextFont (Font newFont)

  Change the font that is used when rendering text (using the Text method).
- void TextFont (string fontName, float pointSize, FontStyle style=FontStyle.Regular)

Change the font that is used when rendering text (using the Text method), using one of the available system fonts

- void TextSize (float pointSize)

  Change the size of the current font.
- void TextAlign (CenterMode horizontal, CenterMode vertical)

  Sets the horizontal and vertical alignment of text. For instance, when choosing CenterMode.Min for both and calling Text, the x and y coordinates give the top left corner of the rendered text.
- void ShapeAlign (CenterMode horizontal, CenterMode vertical)

  Sets the horizontal and vertical alignment of shapes. For instance, when choosing CenterMode.Min for both and calling Ellipse, the x and y coordinates give the top left corner of the drawn ellipse.
- void **NoStroke** ()

  Draw shapes without outline
- void Stroke (Color newColor, int alpha=255)

  Set the outline color for drawing shapes
- void Stroke (int grayScale, int alpha=255)

  Set the outline color for drawing shapes to a grayscale value
- void Stroke (int red, int green, int blue, int alpha=255)

  Set the outline color for drawing shapes.
- void StrokeWeight (float width)

  Sets the width of the outline for drawing shapes. (Default value: 1)
- void **NoFill** ()

  Draw shapes without fill color.
- void Fill (Color newColor, int alpha=255)

  Set the fill color for drawing shapes and text.
- void Fill (int grayScale, int alpha=255)

  Set the fill color for drawing shapes and text to a gray scale value.
- void Fill (int red, int green, int blue, int alpha=255)

  Set the fill color for drawing shapes and text.
- void Clear (Color newColor)

  Clear the canvas with a given color
- void Clear (int grayScale)

  Clear the canvas with a grayscale
- void Clear (int red, int green, int blue, int alpha=255)

  Clear the canvas with a given color.
- void **ClearTransparent** ()

  Clear the canvas with a transparent color. Note that this will fully clear the canvas, but will make the sprites behind the canvas visible.
- void Text (string text, float x, float y)

  Draw text on the canvas, using the currently selected font, at position x,y. This uses the current TextAlign values (e.g. if both are CenterMode.Center, (x,y) will be at the center of the rendered text).
- void Text (string text, bool clear=false, int clearAlpha=0, int clearRed=0, int clearGreen=0, int clearBlue=0)

  Draw text on the canvas, using the currently selected font. The text is aligned on the canvas using the current TextAlign values.
- float TextWidth (string text)

  Returns the width in pixels of a string, when rendered with the current font.
- float TextHeight (string text)

  Returns the height in pixels of a string, when rendered with the current font.
- void TextDimensions (string text, out float width, out float height)

  Returns the width and height in pixels of a string, when rendered with the current font.
- void Rect (float x, float y, float width, float height)

  Draw an (axis aligned) rectangle with given width and height, using the current stroke and fill settings. Uses the current ShapeAlign values to position the rectangle relative to the point (x,y)
- void Ellipse (float x, float y, float width, float height)

Draw an (axis aligned) ellipse (or circle) with given width and height, using the current stroke and fill settings. Uses the current ShapeAlign values to position the rectangle relative to the point (x,y)

- void Arc (float x, float y, float width, float height, float startAngleDegrees, float sweepAngleDegrees)

  Draws an arc (=segment of an ellipse), where width and height give the ellipse size, and start angle and sweep angle can be given (in degrees, clockwise). Uses the current stroke and fill settings. Uses the current ShapeAlign values to position the ellipse relative to the point (x,y)
- void Line (float x1, float y1, float x2, float y2)

  Draw a line segment between two points, using the current stroke settings.
- void **Triangle** (float x1, float y1, float x2, float y2, float x3, float y3)

  Draw a triangle between three points, using the current stroke and fill settings.
- void **Quad** (float x1, float y1, float x2, float y2, float x3, float y3, float x4, float y4)

  Draw a quad (="deformed rectangle") between four points, using the current stroke and fill settings.
- void **Polygon** (params float[ ] pt)

  Draw a polygon shape between any number of points, using the current stroke and fill settings. This requires passing in an even number of float coordinates, where the odd parameters are x coordinates and even parameters are y coordinates.
- void **Polygon** (PointF[ ] pts)

  Draw a polygon shape between any number of points, using the current stroke and fill settings.

Public Attributes

- CenterMode **HorizontalTextAlign** =CenterMode.Min
- CenterMode **VerticalTextAlign** =CenterMode.Max
- CenterMode **HorizontalShapeAlign** =CenterMode.Center
- CenterMode **VerticalShapeAlign** =CenterMode.Center

Protected Member Functions

- void **ShapeAlign** (ref float x, ref float y, float width, float height)

Protected Attributes

- bool **_stroke** =true
- bool **_fill** =true

Properties

- Font **font** `[getprotected set]`
- Pen **pen** `[getprotected set]`
- SolidBrush **brush** `[getprotected set]`

5.12.1 Detailed Description

Creates an easy-to-use layer on top of .NET's System.Drawing methods. The API is inspired by Processing↩ : internal states are maintained for font, fill/stroke color, etc., and everything works with simple methods that have many overloads.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 EasyDraw() [1/3]

```
GXPEngine.EasyDraw.EasyDraw (
            int width,
            int height,
            bool addCollider = true )
```

Creates a new EasyDraw canvas with the given width and height in pixels.

Parameters

width	width in pixels
height	height in pixels
addCollider	whether the canvas should have a collider

5.12.2.2 EasyDraw() [2/3]

```
GXPEngine.EasyDraw.EasyDraw (
            System.Drawing.Bitmap bitmap,
            bool addCollider = true )
```

Creates a new EasyDraw canvas from a given bitmap.

Parameters

bitmap	The bitmap (image) that should be on the canvas
addCollider	whether the canvas should have a collider

5.12.2.3 EasyDraw() [3/3]

```
GXPEngine.EasyDraw.EasyDraw (
            string filename,
            bool addCollider = true )
```

Creates a new EasyDraw canvas from a file that contains a sprite (png, jpg).

Parameters

filename	the name of the file that contains a sprite (png, jpg)
addCollider	whether the canvas should have a collider

5.12.3 Member Function Documentation

5.12.3.1 Arc()

```
void GXPEngine.EasyDraw.Arc (
            float x,
            float y,
            float width,
            float height,
            float startAngleDegrees,
            float sweepAngleDegrees )
```

Draws an arc (=segment of an ellipse), where width and height give the ellipse size, and start angle and sweep angle can be given (in degrees, clockwise). Uses the current stroke and fill settings. Uses the current ShapeAlign values to position the ellipse relative to the point (x,y)

Parameters

x	x position in canvas coordinates
y	y position in canvas coordinates
width	width in pixels
height	height in pixels
startAngleDegrees	angle in degrees (clockwise) to start drawing
sweepAngleDegrees	sweep angle in degrees, clockwise. Use e.g. 180 for a half-circle

5.12.3.2 Clear() [1/3]

```
void GXPEngine.EasyDraw.Clear (
            Color newColor )
```

Clear the canvas with a given color

Parameters

newColor	the clear color

5.12.3.3 Clear() [2/3]

```
void GXPEngine.EasyDraw.Clear (
            int grayScale )
```

Clear the canvas with a grayscale

Parameters

grayScale	the grayscale value (between 0=black and 255=white)

5.12.3.4 Clear() [3/3]

```
void GXPEngine.EasyDraw.Clear (
            int red,
            int green,
            int blue,
            int alpha = 255 )
```

Clear the canvas with a given color.

Parameters

red	The red value of the clear color (from 0 to 255)
green	The green value of the clear color (from 0 to 255)
blue	The blue value of the clear color (from 0 to 255)
alpha	The opacity of the clear color (from 0=transparent to 255=opaque)

5.12.3.5 Ellipse()

```
void GXPEngine.EasyDraw.Ellipse (
            float x,
            float y,
            float width,
            float height )
```

Draw an (axis aligned) ellipse (or circle) with given width and height, using the current stroke and fill settings. Uses the current ShapeAlign values to position the rectangle relative to the point (x,y)

Parameters

x	x position in canvas coordinates
y	y position in canvas coordinates
width	width in pixels
height	height in pixels

5.12.3.6 Fill() [1/3]

```
void GXPEngine.EasyDraw.Fill (
```

```
            Color newColor,
            int alpha = 255 )
```

Set the fill color for drawing shapes and text.

Parameters

newColor	the fill color
alpha	the fill opacity (from 0=transparent to 255=opaque)

5.12.3.7 Fill() [2/3]

```
void GXPEngine.EasyDraw.Fill (
            int grayScale,
            int alpha = 255 )
```

Set the fill color for drawing shapes and text to a gray scale value.

Parameters

grayScale	gray scale value (from 0=black to 255=white)
alpha	the fill opacity (from 0=transparent to 255=opaque)

5.12.3.8 Fill() [3/3]

```
void GXPEngine.EasyDraw.Fill (
            int red,
            int green,
            int blue,
            int alpha = 255 )
```

Set the fill color for drawing shapes and text.

Parameters

red	The red value of the color (from 0 to 255)
green	The green value of the color (from 0 to 255)
blue	The blue value of the color (from 0 to 255)
alpha	The fill opacity (from 0=transparent to 255=opaque)

5.12.3.9 Line()

```
void GXPEngine.EasyDraw.Line (
```

```
            float x1,
            float y1,
            float x2,
            float y2 )
```

Draw a line segment between two points, using the current stroke settings.

Parameters

x1	x coordinate of the start point
y1	y coordinate of the end point
x2	x coordinate of the start point
y2	y coordinate of the end point

5.12.3.10 Rect()

```
void GXPEngine.EasyDraw.Rect (
            float x,
            float y,
            float width,
            float height )
```

Draw an (axis aligned) rectangle with given width and height, using the current stroke and fill settings. Uses the current ShapeAlign values to position the rectangle relative to the point (x,y)

Parameters

x	x position in canvas coordinates
y	y position in canvas coordinates
width	width in pixels
height	height in pixels

5.12.3.11 ShapeAlign()

```
void GXPEngine.EasyDraw.ShapeAlign (
            CenterMode horizontal,
            CenterMode vertical )
```

Sets the horizontal and vertical alignment of shapes. For instance, when choosing CenterMode.Min for both and calling Ellipse, the x and y coordinates give the top left corner of the drawn ellipse.

Parameters

horizontal	Horizontal alignment
vertical	Vertical alignment

5.12.3.12 Stroke() [1/3]

```
void GXPEngine.EasyDraw.Stroke (
            Color newColor,
            int alpha = 255 )
```

Set the outline color for drawing shapes

Parameters

newColor	the color of the outline
alpha	the opacity of the outline (from 0=transparent to 255=opaque)

5.12.3.13 Stroke() [2/3]

```
void GXPEngine.EasyDraw.Stroke (
            int grayScale,
            int alpha = 255 )
```

Set the outline color for drawing shapes to a grayscale value

Parameters

grayScale	A grayscale value (from 0=black to 255=white)
alpha	the opacity of the outline (from 0=transparent to 255=opaque)

5.12.3.14 Stroke() [3/3]

```
void GXPEngine.EasyDraw.Stroke (
            int red,
            int green,
            int blue,
            int alpha = 255 )
```

Set the outline color for drawing shapes.

Parameters

red	The red value of the color (from 0 to 255)
green	The green value of the color (from 0 to 255)
blue	The blue value of the color (from 0 to 255)
alpha	The opacity of the outline (from 0=transparent to 255=opaque)

5.12.3.15 StrokeWeight()

```
void GXPEngine.EasyDraw.StrokeWeight (
            float width )
```

Sets the width of the outline for drawing shapes. (Default value: 1)

Parameters

width	The width (in pixels)

5.12.3.16 Text() [1/2]

```
void GXPEngine.EasyDraw.Text (
            string text,
            bool clear = false,
            int clearAlpha = 0,
            int clearRed = 0,
            int clearGreen = 0,
            int clearBlue = 0 )
```

Draw text on the canvas, using the currently selected font. The text is aligned on the canvas using the current TextAlign values.

Parameters

text	The text to be rendered
clear	Whether the canvas should be cleared before drawing the text
clearAlpha	The opacity of the clear color (from 0=transparent to 255=opaque)
clearRed	The red value of the clear color (0-255)
clearGreen	The green value of the clear color (0-255)
clearBlue	The blue value of the clear color (0-255)

5.12.3.17 Text() [2/2]

```
void GXPEngine.EasyDraw.Text (
            string text,
            float x,
            float y )
```

Draw text on the canvas, using the currently selected font, at position x,y. This uses the current TextAlign values (e.g. if both are CenterMode.Center, (x,y) will be at the center of the rendered text).

Parameters

text	The text to be rendered
x	The x coordinate to draw the text, using canvas (pixel) coordinates
y	The y coordinate to draw the text, using canvas (pixel) coordinates

5.12.3.18 TextAlign()

```
void GXPEngine.EasyDraw.TextAlign (
            CenterMode horizontal,
            CenterMode vertical )
```

Sets the horizontal and vertical alignment of text. For instance, when choosing CenterMode.Min for both and calling Text, the x and y coordinates give the top left corner of the rendered text.

Parameters

horizontal	Horizontal alignment
vertical	Vertical alignment

5.12.3.19 TextDimensions()

```
void GXPEngine.EasyDraw.TextDimensions (
            string text,
            out float width,
            out float height )
```

Returns the width and height in pixels of a string, when rendered with the current font.

Parameters

text	input string
width	width in pixels
height	height in pixels

5.12.3.20 TextFont() [1/2]

```
void GXPEngine.EasyDraw.TextFont (
            Font newFont )
```

Change the font that is used when rendering text (using the Text method).

Parameters

newFont	The new font (see also Utils.LoadFont)

5.12.3.21 TextFont() [2/2]

```
void GXPEngine.EasyDraw.TextFont (
            string fontName,
            float pointSize,
            FontStyle style = FontStyle.Regular )
```

Change the font that is used when rendering text (using the Text method), using one of the available system fonts

Parameters

fontName	The name of the system font (e.g. "Arial", "Verdana", "Vivaldi")
pointSize	font size in points
style	font style (e.g. FontStyle.Italic|FontStyle.Bold)

5.12.3.22 TextHeight()

```
float GXPEngine.EasyDraw.TextHeight (
            string text )
```

Returns the height in pixels of a string, when rendered with the current font.

Parameters

text	input string

Returns

height in pixels

5.12.3.23 TextSize()

```
void GXPEngine.EasyDraw.TextSize (
            float pointSize )
```

Change the size of the current font.

Parameters

pointSize	The font size in points

5.12.3.24 TextWidth()

```
float GXPEngine.EasyDraw.TextWidth (
            string text )
```

Returns the width in pixels of a string, when rendered with the current font.

Parameters

text	input string

Returns

width in pixels

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/EasyDraw.cs

5.13 GXPEngine.Core.FMOD Class Reference

Public Member Functions

- static void **System_Create** (out IntPtr system)
- static void **System_Init** (IntPtr system, int maxChannels, uint flags, int extraDriverData)
- static void **System_CreateSound** (IntPtr system, string filename, uint mode, int uk, out IntPtr sound)
- static void **System_CreateStream** (IntPtr system, string filename, uint mode, int uk, out IntPtr sound)
- static int **System_PlaySound** (IntPtr system, uint channelpref, IntPtr sound, bool paused, out uint channel)
- static void **System_Update** (IntPtr system)
- static void **Channel_GetFrequency** (uint channel, out float frequency)
- static void **Channel_SetFrequency** (uint channel, float frequency)
- static void **Channel_Stop** (uint channel)
- static void **Channel_GetMute** (uint channel, out bool mute)
- static void **Channel_SetMute** (uint channel, bool mute)
- static void **Channel_GetPan** (uint channel, out float pan)
- static void **Channel_SetPan** (uint channel, float pan)
- static void **Channel_GetPaused** (uint channel, out bool paused)
- static void **Channel_SetPaused** (uint channel, bool paused)
- static void **Channel_IsPlaying** (uint channel, out bool playing)
- static void **Channel_GetSpectrum** (uint channel, float[ ] spectrumarray, int numvalues, uint channeloffset, int windowtype)
- static void **Channel_GetVolume** (uint channel, out float volume)
- static void **Channel_SetVolume** (uint channel, float volume)

Static Public Attributes

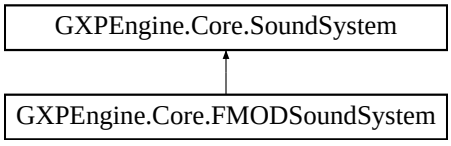
- const int **FMOD_DEFAULT** = 0x00000000
- const int **FMOD_LOOP_OFF** = 0x00000001
- const int **FMOD_LOOP_NORMAL** = 0x00000002
- const int **FMOD_LOOP_BIDI** = 0x00000004
- const int **FMOD_2D** = 0x00000008
- const int **FMOD_3D** = 0x00000010
- const int **FMOD_HARDWARE** = 0x00000020
- const int **FMOD_SOFTWARE** = 0x00000040
- const int **FMOD_CREATESTREAM** = 0x00000080
- const int **FMOD_CREATESAMPLE** = 0x00000100
- const int **FMOD_CREATECOMPRESSEDSAMPLE** = 0x00000200
- const int **FMOD_OPENUSER** = 0x00000400
- const int **FMOD_OPENMEMORY** = 0x00000800
- const int **FMOD_OPENMEMORY_POINT** = 0x10000000
- const int **FMOD_OPENRAW** = 0x00001000
- const int **FMOD_OPENONLY** = 0x00002000
- const int **FMOD_ACCURATETIME** = 0x00004000
- const int **FMOD_MPEGSEARCH** = 0x00008000
- const int **FMOD_NONBLOCKING** = 0x00010000
- const int **FMOD_UNIQUE** = 0x00020000
- const int **FMOD_3D_HEADRELATIVE** = 0x00040000
- const int **FMOD_3D_WORLDRELATIVE** = 0x00080000
- const int **FMOD_3D_INVERSEROLLOFF** = 0x00100000
- const int **FMOD_3D_LINEARROLLOFF** = 0x00200000
- const int **FMOD_3D_LINEARSQUAREROLLOFF** = 0x00400000
- const int **FMOD_3D_CUSTOMROLLOFF** = 0x04000000
- const int **FMOD_3D_IGNOREGEOMETRY** = 0x40000000
- const int **FMOD_UNICODE** = 0x01000000
- const int **FMOD_IGNORETAGS** = 0x02000000
- const int **FMOD_LOWMEM** = 0x08000000
- const int **FMOD_LOADSECONDARYRAM** = 0x20000000
- const uint **FMOD_VIRTUAL_PLAYFROMSTART** = 0x80000000
- const int **FMOD_CHANNEL_FREE** = -1
- const int **FMOD_CHANNEL_REUSE** = -2

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/FMOD/FMOD.cs

5.14 GXPEngine.Core.FMODSoundSystem Class Reference

Inheritance diagram for GXPEngine.Core.FMODSoundSystem:



Public Member Functions

- override void Init ()
- override void Deinit ()
- override IntPtr CreateStream (string filename, bool looping)
- override IntPtr LoadSound (string filename, bool looping)
- override void Step ()
- override uint PlaySound (IntPtr id, uint channelId, bool paused)
- override uint PlaySound (IntPtr id, uint channelId, bool paused, float volume, float pan)
- override float GetChannelFrequency (uint channelId)
- override void SetChannelFrequency (uint channelId, float frequency)
- override float GetChannelPan (uint channelId)
- override void SetChannelPan (uint channelId, float pan)
- override bool GetChannelPaused (uint channelId)
- override void SetChannelPaused (uint channelId, bool pause)
- override bool ChannelIsPlaying (uint channelId)
- override void StopChannel (uint channelId)
- override float GetChannelVolume (uint channelId)
- override void SetChannelVolume (uint channelId, float volume)

5.14.1 Member Function Documentation

5.14.1.1 ChannelIsPlaying()

```
override bool GXPEngine.Core.FMODSoundSystem.ChannelIsPlaying (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.2 CreateStream()

```
override IntPtr GXPEngine.Core.FMODSoundSystem.CreateStream (
            string filename,
            bool looping )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.3 Deinit()

```
override void GXPEngine.Core.FMODSoundSystem.Deinit ( )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.4 GetChannelFrequency()

```
override float GXPEngine.Core.FMODSoundSystem.GetChannelFrequency (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.5 GetChannelPan()

```
override float GXPEngine.Core.FMODSoundSystem.GetChannelPan (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.6 GetChannelPaused()

```
override bool GXPEngine.Core.FMODSoundSystem.GetChannelPaused (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.7 GetChannelVolume()

```
override float GXPEngine.Core.FMODSoundSystem.GetChannelVolume (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.8 Init()

```
override void GXPEngine.Core.FMODSoundSystem.Init ( )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.9 LoadSound()

```
override IntPtr GXPEngine.Core.FMODSoundSystem.LoadSound (
            string filename,
            bool looping )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.10 PlaySound() [1/2]

```
override uint GXPEngine.Core.FMODSoundSystem.PlaySound (
            IntPtr id,
            uint channelId,
            bool paused )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.11 PlaySound() [2/2]

```
override uint GXPEngine.Core.FMODSoundSystem.PlaySound (
            IntPtr id,
            uint channelId,
            bool paused,
            float volume,
            float pan )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.12 SetChannelFrequency()

```
override void GXPEngine.Core.FMODSoundSystem.SetChannelFrequency (
            uint channelId,
            float frequency )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.13 SetChannelPan()

```
override void GXPEngine.Core.FMODSoundSystem.SetChannelPan (
            uint channelId,
            float pan )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.14 SetChannelPaused()

```
override void GXPEngine.Core.FMODSoundSystem.SetChannelPaused (
            uint channelId,
            bool pause )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.15 SetChannelVolume()

```
override void GXPEngine.Core.FMODSoundSystem.SetChannelVolume (
            uint channelId,
            float volume )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.16 Step()

```
override void GXPEngine.Core.FMODSoundSystem.Step ( )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.14.1.17 StopChannel()

```
override void GXPEngine.Core.FMODSoundSystem.StopChannel (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

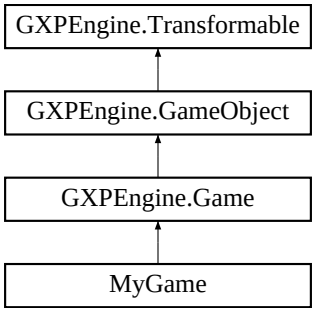
The documentation for this class was generated from the following file:

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/FMOD/FMODSoundSystem.cs

5.15 GXPEngine.Game Class Reference

The Game class represents the Game window. Only a single instance of this class is allowed.

Inheritance diagram for GXPEngine.Game:



Public Member Functions

- delegate void **StepDelegate** ()

  Step delegate defines the signature of a method used for step callbacks, see OnBeforeStep, OnAfterStep.
- delegate void **RenderDelegate** (GLContext glContext)
- Game (int pWidth, int pHeight, bool pFullScreen, bool pVSync=true, int pRealWidth=-1, int pRealHeight=-1, bool pPixelArt=false)

  Initializes a new instance of the GXPEngine.Game class. This class represents a game window, containing an openGL view.
- void SetViewport (int x, int y, int width, int height, bool setRenderRange=true)

  Sets the rendering output view port. All rendering will be done within the given rectangle. The default setting is {0, 0, game.width, game.height}.
- void ShowMouse (bool enable)

  Shows or hides the mouse cursor.
- void **Start** ()

  Start the game loop. Call this once at the start of your game.
- override void Render (GLContext glContext)

  This function is called by the renderer. You can override it to change this object's rendering behaviour. When not inside the GXPEngine package, specify the parameter as GXPEngine.Core.GLContext. This function was made public to accomodate split screen rendering. Use SetViewPort for that.
- override void Destroy ()

  Destroys the game, and closes the game window.
- void **SetVSync** (bool enableVSync)
- string GetDiagnostics ()

  Returns a string with some internal engine information. Use this for debugging, especially when the game slows down.

Public Attributes

- bool **RenderMain** = true

  Set this to false if you only want to render using your own cameras. (Don't say we didn't warn you if you end up with a black screen...)
- readonly bool **PixelArt**

Protected Member Functions

- override void RenderSelf (GLContext glContext)

Properties

- Rectangle RenderRange `[getset]`

  Sprites will be rendered if and only if they overlap with this rectangle. Default value: (0,0,game.width,game.height). You only need to change this when rendering to subwindows (e.g. split screen).
- int **width** `[get]`

  Returns the width of the window.
- int **height** `[get]`

  Returns the height of the window.
- int **currentFps** `[get]`
- int **targetFps** `[getset]`

Events

- StepDelegate **OnBeforeStep**

  Occurs before the engine starts the new update loop. This allows you to define general manager classes that can update itself on/after each frame.
- StepDelegate **OnAfterStep**

  Occurs after the engine has finished its last update loop. This allows you to define general manager classes that can update itself on/after each frame.
- RenderDelegate **OnAfterRender**

Additional Inherited Members

5.15.1 Detailed Description

The Game class represents the Game window. Only a single instance of this class is allowed.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 Game()

```
GXPEngine.Game.Game (
            int pWidth,
            int pHeight,
            bool pFullScreen,
            bool pVSync = true,
            int pRealWidth = -1,
            int pRealHeight = -1,
            bool pPixelArt = false )
```

Initializes a new instance of the GXPEngine.Game class. This class represents a game window, containing an openGL view.

Parameters

width	Width of the window in pixels. (As used by all the logic, e.g. the coordinate system)
height	Height of the window in pixels. (As used by all the logic, e.g. the coordinate system)
fullScreen	If set to `true` the application will run in fullscreen mode.
vSync	If `true`, the frame rate will sync to the screen refresh rate, so setting targetFps has no effect. (It's best to give this the same value as the 'fullScreen' parameter.)
realWidth	The actual window width. By default (if passing a negative value), this is equal to the 'width' parameter, but using this parameter you can easily change the window width without having to change any other code.
realHeight	The actual window height. By default (if passing a negative value), this is equal to the 'height' parameter, but using this parameter you can easily change the window height without having to change any other code.
pixelArt	If `true`, textures will not be interpolated ('blurred'). This way, you can get a typical pixel art look.

5.15.3 Member Function Documentation

5.15.3.1 Destroy()

```
override void GXPEngine.Game.Destroy ( )  [virtual]
```

Destroys the game, and closes the game window.

Reimplemented from GXPEngine.GameObject.

5.15.3.2 GetDiagnostics()

```
string GXPEngine.Game.GetDiagnostics ( )
```

Returns a string with some internal engine information. Use this for debugging, especially when the game slows down.

Returns

Internal engine information.

5.15.3.3 Render()

```
override void GXPEngine.Game.Render (
            GLContext glContext )  [virtual]
```

This function is called by the renderer. You can override it to change this object's rendering behaviour. When not inside the GXPEngine package, specify the parameter as GXPEngine.Core.GLContext. This function was made public to accomodate split screen rendering. Use SetViewPort for that.

Parameters

glContext	Gl context, will be supplied by internal caller.

Reimplemented from GXPEngine.GameObject.

5.15.3.4 RenderSelf()

```
override void GXPEngine.Game.RenderSelf (
            GLContext glContext )  [protected], [virtual]
```

Reimplemented from GXPEngine.GameObject.

5.15.3.5 SetViewport()

```
void GXPEngine.Game.SetViewport (
            int x,
            int y,
            int width,
            int height,
            bool setRenderRange = true )
```

Sets the rendering output view port. All rendering will be done within the given rectangle. The default setting is {0, 0, game.width, game.height}.

Parameters

x	The x coordinate.
y	The y coordinate.
width	The new width of the viewport.
height	The new height of the viewport.

5.15.3.6 ShowMouse()

```
void GXPEngine.Game.ShowMouse (
            bool enable )
```

Shows or hides the mouse cursor.

Parameters

enable	Set this to 'true' to enable the cursor. Else, set this to 'false'.

5.15.4 Property Documentation

5.15.4.1 RenderRange

```
Rectangle GXPEngine.Game.RenderRange  [get], [set]
```

Sprites will be rendered if and only if they overlap with this rectangle. Default value: (0,0,game.width,game.height). You only need to change this when rendering to subwindows (e.g. split screen).

The render range.

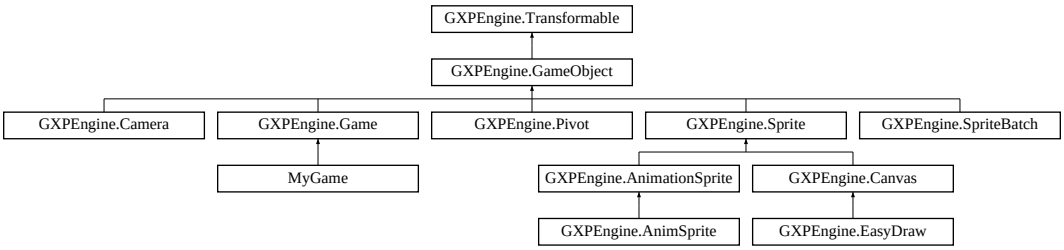
The documentation for this class was generated from the following file:

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Game.cs

5.16 GXPEngine.GameObject Class Reference

GameObject is the base class for all display objects.

Inheritance diagram for GXPEngine.GameObject:



Public Member Functions

- GameObject (bool addCollider=false)

  Initializes a new instance of the GXPEngine.GameObject class. Since GameObjects contain a display hierarchy, a GameObject can be used as a container for other objects. Other objects can be added using child commands as AddChild.
- GameObject[ ] **GetCollisions** (bool includeTriggers=true, bool includeSolid=true)

  Get all a list of all objects that currently overlap this one. Calling this method will test collisions between this object and all other colliders in the scene. It can be called mid-step and is included for convenience, not performance. Set includeTriggers to true to include trigger colliders in the list, and includeSolid to include solid (=non-trigger) colliders.
- virtual void Render (GLContext glContext)

  This function is called by the renderer. You can override it to change this object's rendering behaviour. When not inside the GXPEngine package, specify the parameter as GXPEngine.Core.GLContext. This function was made public to accomodate split screen rendering. Use SetViewPort for that.
- virtual void Destroy ()

  Destroy this instance, and removes it from the game. To complete garbage collection, you must nullify all your own references to this object.
- void **LateDestroy** ()

  Destroy this instance, and removes it from the game, after finishing the current Update + OnCollision loops. To complete garbage collection, you must nullify all your own references to this object.
- void AddChild (GameObject child)

  Adds the specified GameObject as a child to this one.
- void LateAddChild (GameObject child)

  Adds the specified GameObject as a child to this one, after finishing the current Update + OnCollision loops.
- void **Remove** ()

  Removes this GameObject from the hierarchy (=sets the parent to null).
- void **LateRemove** ()

  Removes this GameObject from the hierarchy, after finishing the current Update + OnCollision loops.
- void RemoveChild (GameObject child)

  Removes the specified child GameObject from this object.
- void AddChildAt (GameObject child, int index)

  Adds the specified GameObject as a child to this object at an specified index. This will alter the position of other objects as well. You can use this to determine the draw order of child objects.

- void **LateAddChildAt** (GameObject child, int index)

  Adds the specified GameObject as a child to this one, at the specified index, after finishing the current Update + OnCollision loops.
- bool HasChild (GameObject gameObject)

  Returns 'true' if the specified object is a descendant of this object.
- bool **InHierarchy** ()

  Returns whether this game object is currently active, or equivalently, a descendant of Game.
- List< GameObject > **GetChildren** (bool safe=true)

  Returns a list of all children that belong to this object. The function returns System.Collections.Generic.List<Game↩ Object>. (If safe=false, then the method is slightly faster, but modifying the list will break the engine!)
- int GetChildCount ()

  Returns the number of children of this game object.
- void SetChildIndex (GameObject child, int index)

  Inserts the specified object in this object's child list at given location. This will alter the position of other objects as well. You can use this to determine the drawing order of child objects.
- virtual bool HitTest (GameObject other)

  Tests if this object overlaps with the one specified.
- virtual float **TimeOfImpact** (GameObject other, float vx, float vy, out Vector2 normal)

  If changing the x and y coordinates of this GameObject by vx and vy respectively would cause a collision with the GameObject other, this method returns a "time of impact" between 0 and 1, which is a scalar multiplier for vx and vy, giving the amount of safe movement until collision. It is zero if the two game objects are already overlapping, and moving by vx and vy would cause a worse overlap. In all other cases, the returned value is bigger than 1. If a time of impact below 1 is returned, the normal will be the collision normal (otherwise it is undefined).
- virtual Collision MoveUntilCollision (float vx, float vy, IEnumerable< GameObject > objectsToCheck)

  Tries to move this object by vx,vy (in parent space, similar to the translate method), until it collides with one of the given objects. Objects without a solid (=non-trigger) collider are ignored. In case of a collision, it returns a Collision object with information such as the normal and time of impact (the point and penetration depth fields of the collision object will always be zero). Otherwise it returns null.
- virtual Collision MoveUntilCollision (float vx, float vy)

  Tries to move this object by vx,vy (in parent space, similar to the translate method), until it collides with another object that has a solid (=non-trigger) collider. In case of a collision, it returns a Collision object with information such as the normal and time of impact (the point and penetration depth fields of the collision object will always be zero). Otherwise it returns null.
- virtual bool HitTestPoint (float x, float y)

  Returns `true` if a 2D point (given in global / screen space) overlaps with this object. You could use this for instance to check if the mouse (Input.mouseX, Input.mouseY) is over the object.
- override Vector2 TransformPoint (float x, float y)

  Transforms a point from local to global space. If you insert a point relative to the object, it will return that same point relative to the game.
- override Vector2 TransformDirection (float x, float y)

  Transforms a direction vector from local to global space. If you insert a vector relative to the object, it will return that same vector relative to the game. Note: only scale and rotation information are taken into account, not translation (coordinates).
- override Vector2 InverseTransformPoint (float x, float y)

  Transforms the point from global into local space. If you insert a point relative to the game, it will return that same point relative to this GameObject.
- override Vector2 InverseTransformDirection (float x, float y)

  Transforms the vector from global into local space. If you insert a vector relative to the game, it will return that same vector relative to this GameObject. Note: only scale and rotation information are taken into account, not translation (coordinates).
- GameObject FindObjectOfType (Type type)

  Returns the first object of the given type, found within the descendants of this game object (including this game object itself). If there's no descendant of the given type, returns null. For example, if you have made a Player class, call this method like this: game.FindObjectOfType(typeof(Player));
- T FindObjectOfType< T > ()

Returns the first object of the given type, found within the descendants of this game object (including this game object itself). If there's no descendant of the given type, returns null. The given type must inherit from GameObject.

- GameObject[ ] FindObjectsOfType (Type type)

  Returns the all objects of the given type, found within the descendants of this game object (including this game object itself). For example, if you have made a Player class, call this like this: game.FindObjectsOfType(typeof(Player));
- T[ ] FindObjectsOfType< T > ()

  Returns the all objects of the given type, found within the descendants of this game object (including this game object itself). The type must inherit from GameObject.
- override string **ToString** ()

Public Attributes

- string **name**
- bool **visible** = true

Protected Member Functions

- virtual Collider createCollider ()

  Create and return a collider to use for this game object. Null is allowed.
- virtual void **RenderSelf** (GLContext glContext)
- virtual void OnDestroy ()

  Subclasses can implement this method to clean up resources once on destruction. Will be called by the engine when the game object is destroyed.

Properties

- int Index `[get]`

  Gets the index of this object in the parent's hierarchy list. Returns -1 if no parent is defined.
- Game **game** `[get]`

  Gets the game that this object belongs to. This is a unique instance throughout the runtime of the game. Use this to access the top of the displaylist hierarchy, and to retreive the width and height of the screen.
- GameObject **parent** `[getset]`

  Gets or sets the parent GameObject. When the parent moves, this object moves along.

Additional Inherited Members

5.16.1 Detailed Description

GameObject is the base class for all display objects.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 GameObject()

```
GXPEngine.GameObject.GameObject (
            bool addCollider = false )
```

Initializes a new instance of the GXPEngine.GameObject class. Since GameObjects contain a display hierarchy, a GameObject can be used as a container for other objects. Other objects can be added using child commands as AddChild.

Parameters

addCollider	If `true`, then the virtual function createCollider will be called, which can be overridden to create a collider that will be added to the collision manager.

5.16.3 Member Function Documentation

5.16.3.1 AddChild()

```
void GXPEngine.GameObject.AddChild (
            GameObject child )
```

Adds the specified GameObject as a child to this one.

Parameters

child	Child object to add.

5.16.3.2 AddChildAt()

```
void GXPEngine.GameObject.AddChildAt (
            GameObject child,
            int index )
```

Adds the specified GameObject as a child to this object at an specified index. This will alter the position of other objects as well. You can use this to determine the draw order of child objects.

Parameters

child	Child object to add.
index	Index in the child list where the object should be added.

5.16.3.3 createCollider()

```
virtual Collider GXPEngine.GameObject.createCollider ( )  [protected], [virtual]
```

Create and return a collider to use for this game object. Null is allowed.

Reimplemented in GXPEngine.Sprite.

5.16.3.4 Destroy()

```
virtual void GXPEngine.GameObject.Destroy ( )  [virtual]
```

Destroy this instance, and removes it from the game. To complete garbage collection, you must nullify all your own references to this object.

Reimplemented in GXPEngine.Game.

5.16.3.5 FindObjectOfType()

```
GameObject GXPEngine.GameObject.FindObjectOfType (
            Type type )
```

Returns the first object of the given type, found within the descendants of this game object (including this game object itself). If there's no descendant of the given type, returns null. For example, if you have made a Player class, call this method like this: game.FindObjectOfType(typeof(Player));

Parameters

type	The object type you're looking for (must inherit from GameObject)

Returns

A descendant of the given type, if it exists.

5.16.3.6 FindObjectOfType< T >()

```
T GXPEngine.GameObject.FindObjectOfType< T > ( )
```

Returns the first object of the given type, found within the descendants of this game object (including this game object itself). If there's no descendant of the given type, returns null. The given type must inherit from GameObject.

Returns

A descendant of the given type, if it exists.

Type Constraints

T* : *GameObject

5.16.3.7 FindObjectsOfType()

```
GameObject[] GXPEngine.GameObject.FindObjectsOfType (
            Type type )
```

Returns the all objects of the given type, found within the descendants of this game object (including this game object itself). For example, if you have made a Player class, call this like this: game.FindObjectsOfType(typeof(↩ Player));

Parameters

type	The object type you're looking for (must inherit from GameObject)

Returns

All descendants of the given type.

5.16.3.8 FindObjectsOfType< T >()

```
T[] GXPEngine.GameObject.FindObjectsOfType< T > ( )
```

Returns the all objects of the given type, found within the descendants of this game object (including this game object itself). The type must inherit from GameObject.

Returns

All descendants of the given type.

Type Constraints

T* : *GameObject

5.16.3.9 GetChildCount()

```
int GXPEngine.GameObject.GetChildCount ( )
```

Returns the number of children of this game object.

Returns

The number of children of this game object.

5.16.3.10 HasChild()

```
bool GXPEngine.GameObject.HasChild (
            GameObject gameObject )
```

Returns 'true' if the specified object is a descendant of this object.

Parameters

gameObject	The GameObject that should be tested.

5.16.3.11 HitTest()

```
virtual bool GXPEngine.GameObject.HitTest (
            GameObject other )  [virtual]
```

Tests if this object overlaps with the one specified.

Returns

`true`, if 'this' overlaps with 'other'.

Parameters

other	The other game object.

5.16.3.12 HitTestPoint()

```
virtual bool GXPEngine.GameObject.HitTestPoint (
            float x,
            float y )  [virtual]
```

Returns `true` if a 2D point (given in global / screen space) overlaps with this object. You could use this for instance to check if the mouse (Input.mouseX, Input.mouseY) is over the object.

Parameters

x	The x coordinate to test.
y	The y coordinate to test.

5.16.3.13 InverseTransformDirection()

```
override Vector2 GXPEngine.GameObject.InverseTransformDirection (
            float x,
            float y )  [virtual]
```

Transforms the vector from global into local space. If you insert a vector relative to the game, it will return that same vector relative to this GameObject. Note: only scale and rotation information are taken into account, not translation (coordinates).

Parameters

x	The x coordinate to transform.
y	The y coordinate to transform.

Reimplemented from GXPEngine.Transformable.

5.16.3.14 InverseTransformPoint()

```
override Vector2 GXPEngine.GameObject.InverseTransformPoint (
            float x,
            float y )  [virtual]
```

Transforms the point from global into local space. If you insert a point relative to the game, it will return that same point relative to this GameObject.

Parameters

x	The x coordinate to transform.
y	The y coordinate to transform.

Reimplemented from GXPEngine.Transformable.

5.16.3.15 LateAddChild()

```
void GXPEngine.GameObject.LateAddChild (
            GameObject child )
```

Adds the specified GameObject as a child to this one, *after* finishing the current Update + OnCollision loops.

Parameters

child	Child object to add.

5.16.3.16 MoveUntilCollision() [1/2]

```
virtual Collision GXPEngine.GameObject.MoveUntilCollision (
            float vx,
            float vy )  [virtual]
```

Tries to move this object by vx,vy (in parent space, similar to the translate method), until it collides with another object that has a solid (=non-trigger) collider. In case of a collision, it returns a Collision object with information such

as the normal and time of impact (the point and penetration depth fields of the collision object will always be zero). Otherwise it returns null.

Note: this is a very expensive method since it uses GetCollisions, and tunneling is possible since it uses discrete collision detection - use with care.

5.16.3.17 MoveUntilCollision() [2/2]

```
virtual Collision GXPEngine.GameObject.MoveUntilCollision (
            float vx,
            float vy,
            IEnumerable< GameObject > objectsToCheck )  [virtual]
```

Tries to move this object by vx,vy (in parent space, similar to the translate method), until it collides with one of the given objects. Objects without a solid (=non-trigger) collider are ignored. In case of a collision, it returns a Collision object with information such as the normal and time of impact (the point and penetration depth fields of the collision object will always be zero). Otherwise it returns null.

As objectsToCheck, pass an array or List of game objects to check against (this moving game object will move through all objects that are not in the given array or list).

5.16.3.18 OnDestroy()

```
virtual void GXPEngine.GameObject.OnDestroy ( )  [protected], [virtual]
```

Subclasses can implement this method to clean up resources once on destruction. Will be called by the engine when the game object is destroyed.

Reimplemented in GXPEngine.Camera, GXPEngine.SpriteBatch, and GXPEngine.Sprite.

5.16.3.19 RemoveChild()

```
void GXPEngine.GameObject.RemoveChild (
            GameObject child )
```

Removes the specified child GameObject from this object.

Parameters

child	Child object to remove.

5.16.3.20 Render()

```
virtual void GXPEngine.GameObject.Render (
            GLContext glContext )  [virtual]
```

This function is called by the renderer. You can override it to change this object's rendering behaviour. When not inside the GXPEngine package, specify the parameter as GXPEngine.Core.GLContext. This function was made public to accomodate split screen rendering. Use SetViewPort for that.

Parameters

glContext	Gl context, will be supplied by internal caller.

Reimplemented in GXPEngine.Game.

5.16.3.21 SetChildIndex()

```
void GXPEngine.GameObject.SetChildIndex (
            GameObject child,
            int index )
```

Inserts the specified object in this object's child list at given location. This will alter the position of other objects as well. You can use this to determine the drawing order of child objects.

Parameters

child	Child.
index	Index.

5.16.3.22 TransformDirection()

```
override Vector2 GXPEngine.GameObject.TransformDirection (
            float x,
            float y )  [virtual]
```

Transforms a direction vector from local to global space. If you insert a vector relative to the object, it will return that same vector relative to the game. Note: only scale and rotation information are taken into account, not translation (coordinates).

Parameters

x	The x coordinate to transform.
y	The y coordinate to transform.

Reimplemented from GXPEngine.Transformable.

5.16.3.23 TransformPoint()

```
override Vector2 GXPEngine.GameObject.TransformPoint (
```

```
float x,
float y )  [virtual]
```

Transforms a point from local to global space. If you insert a point relative to the object, it will return that same point relative to the game.

Parameters

x	The x coordinate to transform.
y	The y coordinate to transform.

Reimplemented from GXPEngine.Transformable.

5.16.4 Property Documentation

5.16.4.1 Index

```
int GXPEngine.GameObject.Index  [get]
```

Gets the index of this object in the parent's hierarchy list. Returns -1 if no parent is defined.

The index.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/GameObject.cs

5.17 GXPEngine.Gizmos Class Reference

This class can be used to easily draw line based shapes (like arrows and rectangles), mostly for debug purposes (it is not made for efficiency). For each draw call, shapes are drawn for one frame only, after rendering all sprites. See the DrawLine method for more information.

Static Public Member Functions

- static void **SetStyle** (uint color, byte width)

  Set a default color and line width for the subsequent draw calls. The color should be given as a uint consisting of four byte values, in the order ARGB.
- static void SetWidth (byte width)

  Set a default line width for the subsequent draw calls.
- static void **SetColor** (float R, float G, float B, float alpha=1)

  Set the default color for subsequent draw calls. The R,G,B color and alpha values should be given as floats between 0 and 1.
- static void **DrawLine** (float x1, float y1, float x2=0, float y2=0, GameObject space=null, uint color=0, byte width=0)

You can call this method from anywhere. A (debug) line will be drawn from (x1,y1) to (x2,y2), in the space of the given game object. If no game object is given, it will be drawn in screen space. The line will be drawn after drawing all other game objects. You can give color and line width. If no values are given (=0), the default values are used. These can be set using SetStyle, SetColor and SetWidth.

- static void **DrawPlus** (float x, float y, float radius, GameObject space=null, uint color=0, byte width=0)

  Draws a plus shape centered at the point x,y, with given radius, using DrawLine.
- static void **DrawCross** (float x, float y, float radius, GameObject space=null, uint color=0, byte width=0)

  Draws a cross shape centered at the point x,y, with given radius, using DrawLine.
- static void **DrawRay** (float x, float y, float dx, float dy, GameObject space=null, uint color=0, byte width=0)

  Draws a line segment from (x,y) to (x+dx, y+dy), using DrawLine.
- static void **DrawRayAngle** (float x, float y, float angleDegrees, float length, GameObject space=null, uint color=0, byte width=0)

  Draws a line segment starting at (x,y), with the given length and angle in degrees, using DrawLine.
- static void **DrawArrow** (float x, float y, float dx, float dy, float relativeArrowSize=0.25f, GameObject space=null, uint color=0, byte width=0)

  Draws an arrow from (x,y) to (x+dx, y+dy), using DrawLine. The relativeArrowSize is the size of the arrow head compared to the arrow length.
- static void **DrawArrowAngle** (float x, float y, float angleDegrees, float length, float relativeArrowSize=0.25f, GameObject space=null, uint color=0, byte width=0)

  Draws an arrow starting at (x,y), with the given length and angle in degrees, using DrawLine. The relativeArrowSize is the size of the arrow head compared to the arrow length.
- static void **DrawRectangle** (float xCenter, float yCenter, float width, float height, GameObject space=null, uint color=0, byte lineWidth=0)

  Draws an axis-aligned rectangle centered at a given point, with given width and height, using DrawLine.
- static void **RenderLine** (float x1, float y1, float x2, float y2, uint pColor=0xffffffff, uint pLineWidth=1, bool pGlobalCoords=false)

  This method should typically be called from the RenderSelf method of a GameObject, or from the game's OnAfter↩ Render event. The line from (x1,y1) to (x2,y2) is then drawn immediately, behind objects that are drawn later. It is drawn in the space of the game object itself if called from RenderSelf with pGlobalCoords=false, and in screen space otherwise. You can give color and line width. If no values are given (=0), the default values are used. These can be set using SetStyle, SetColor and SetWidth.

5.17.1 Detailed Description

This class can be used to easily draw line based shapes (like arrows and rectangles), mostly for debug purposes (it is not made for efficiency). For each draw call, shapes are drawn for one frame only, after rendering all sprites. See the DrawLine method for more information.

5.17.2 Member Function Documentation

5.17.2.1 SetWidth()

```
static void GXPEngine.Gizmos.SetWidth (
            byte width )  [static]
```

Set a default line width for the subsequent draw calls.

Parameters

width	

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/Gizmos.cs

5.18 GXPEngine.OpenGL.GL Class Reference

Public Member Functions

- static void **Enable** (int cap)
- static void **Disable** (int cap)
- static void **BlendFunc** (int sourceFactor, int destFactor)
- static void **BlendEquation** (int mode)
- static void **ClearColor** (float r, float g, float b, float a)
- static void **MatrixMode** (int mode)
- static void **LoadIdentity** ()
- static void **Ortho** (double left, double right, double top, double bottom, double near, double far)
- static void **Clear** (int mask)
- static void **Color4ub** (byte r, byte g, byte b, byte a)
- static void **PushMatrix** ()
- static void **MultMatrixf** (float[ ] matrix)
- static void **PopMatrix** ()
- static void **Begin** (int mode)
- static void **TexCoord2f** (float u, float v)
- static void **Vertex2f** (float x, float y)
- static void **Vertex3f** (float x, float y, float z)
- static void **End** ()
- static void **BindTexture** (int target, int texture)
- static void **GenTextures** (int count, int[ ] textures)
- static void **TexParameteri** (int target, int name, int value)
- static void **TexImage2D** (int target, int level, int internalFormat, int width, int height, int border, int format, int type, IntPtr pixels)
- static void **DeleteTextures** (int count, int[ ] textures)
- static void **Flush** ()
- static void **Finish** ()
- static void **Hint** (int target, int mode)
- static void **Viewport** (int x, int y, int width, int height)
- static void **Scissor** (int x, int y, int width, int height)
- static void **VertexPointer** (int size, int type, int stride, float[ ] pointer)
- static void **TexCoordPointer** (int size, int type, int stride, float[ ] pointer)
- static void **DrawElements** (int mode, int count, int type, int[ ] indices)
- static void **EnableClientState** (int array)
- static void **ArrayElement** (int element)
- static void **DrawArrays** (int mode, int offset, int count)
- static void **DisableClientState** (int state)
- static int **GetError** ()
- static void **GetIntegerv** (int name, int[ ] param)

- static void **LineWidth** (float width)
- delegate void **GLFWWindowSizeCallback** (int width, int height)
- delegate void **GLFWKeyCallback** (int key, int action)
- delegate void **GLFWMouseButtonCallback** (int button, int action)
- static void **glfwSetTime** (double time)
- static double **glfwGetTime** ()
- static void **glfwPollEvents** ()
- static int **glfwGetWindowParam** (int param)
- static void **glfwInit** ()
- static void **glfwOpenWindow** (int width, int height, int r, int g, int b, int a, int depth, int stencil, int mode)
- static void **glfwSetWindowTitle** (string title)
- static void **glfwSwapInterval** (bool mode)
- static void **glfwSetWindowSizeCallback** (GLFWWindowSizeCallback callback)
- static void **glfwCloseWindow** ()
- static void **glfwTerminate** ()
- static void **glfwSwapBuffers** ()
- static bool **glfwGetKey** (int key)
- static void **glfwSetKeyCallback** (GLFWKeyCallback callback)
- static void **glfwOpenWindowHint** (int name, int value)
- static bool **glfwGetMousePos** (out int x, out int y)
- static void **glfwSetMouseButtonCallback** (GLFWMouseButtonCallback callback)
- static void **glfwEnable** (int property)
- static void **glfwDisable** (int property)

Static Public Attributes

- const int **TEXTURE_2D** = 0x0DE1
- const int **BLEND** = 0x0BE2
- const int **MODELVIEW** = 0x1700
- const int **PROJECTION** = 0x1701
- const int **COLOR_BUFFER_BIT** = 0x4000
- const int **QUADS** = 0x0007
- const int **TRIANGLES** = 0x0004
- const int **LINES** = 0x0001
- const int **TEXTURE_MIN_FILTER** = 0x2801
- const int **TEXTURE_MAG_FILTER** = 0x2800
- const int **LINEAR** = 0x2601
- const int **TEXTURE_WRAP_S** = 0x2802
- const int **TEXTURE_WRAP_T** = 0x2803
- const int **CLAMP** = 0x2900
- const int **GL_CLAMP_TO_EDGE_EXT** = 0x812F
- const int **RGBA** = 0x1908
- const int **BGRA** = 0x80E1
- const int **UNSIGNED_BYTE** = 0x1401
- const int **PERSPECTIVE_CORRECTION** = 0x0C50
- const int **FASTEST** = 0x1101
- const int **NICEST** = 0x1102
- const int **NEAREST** = 0x2600
- const int **POLYGON_SMOOTH** = 0x0B41
- const int **MULTISAMPLE** = 0x809D
- const int **FLOAT** = 0x1406
- const int **UNSIGNED_INT** = 0x1405
- const int **VERTEX_ARRAY** = 0x8074
- const int **INT** = 0x1404

- const int **DOUBLE** = 0x140A
- const int **INDEX_ARRAY** = 0x8077
- const int **TEXTURE_COORD_ARRAY** = 0x8078
- const int **SCISSOR_TEST** = 0x0C11
- const int **MAX_TEXTURE_SIZE** = 0x0D33
- const int **ZERO** = 0x0000
- const int **ONE** = 0x0001
- const int **SRC_COLOR** = 0x0300
- const int **ONE_MINUS_SRC_COLOR** = 0x0301
- const int **DST_COLOR** = 0x0306
- const int **ONE_MINUS_DST_COLOR** = 0x0307
- const int **SRC_ALPHA** = 0x0302
- const int **ONE_MINUS_SRC_ALPHA** = 0x0303
- const int **DST_ALPHA** = 0x0304
- const int **ONE_MINUS_DST_ALPHA** = 0x0305
- const int **CONSTANT_COLOR** = 0x8001
- const int **ONE_MINUS_CONSTANT_COLOR** = 0x8002
- const int **CONSTANT_ALPHA** = 0x8003
- const int **ONE_MINUS_CONSTANT_ALPHA** = 0x8004
- const int **SRC_ALPHA_SATURATE** = 0x0308
- const int **MIN** = 0x8007
- const int **MAX** = 0x8008
- const int **FUNC_ADD** = 0x8006
- const int **FUNC_SUBTRACT** = 0x800A
- const int **FUNC_REVERSE_SUBTRACT** = 0x800B
- const int **GL_REPEAT** = 0x2901
- const int **GLFW_OPENED** = 0x00020001
- const int **GLFW_WINDOWED** = 0x00010001
- const int **GLFW_FULLSCREEN** = 0x00010002
- const int **GLFW_ACTIVE** = 0x00020001
- const int **GLFW_FSAA_SAMPLES** = 0x0002100E
- const int **GLFW_MOUSE_CURSOR** = 0x00030001

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/OpenGL/GL.cs

5.19 GXPEngine.Core.GLContext Class Reference

Public Member Functions

- **GLContext** (Game owner)
- void **CreateWindow** (int width, int height, bool fullScreen, bool vSync, int realWidth, int realHeight)
- void **ShowCursor** (bool enable)
- void **SetVSync** (bool enableVSync)
- void **SetScissor** (int x, int y, int width, int height)
- void **Close** ()
- void **Run** ()
- void **SetColor** (byte r, byte g, byte b, byte a)
- void **PushMatrix** (float[ ] matrix)
- void **PopMatrix** ()
- void **DrawQuad** (float[ ] vertices, float[ ] uv)

Static Public Member Functions

- static bool **GetKey** (int key)
- static bool **GetKeyDown** (int key)
- static bool **GetKeyUp** (int key)
- static bool **AnyKey** ()
- static bool **AnyKeyDown** ()
- static bool **GetMouseButton** (int button)
- static bool **GetMouseButtonDown** (int button)
- static bool **GetMouseButtonUp** (int button)
- static void **ResetHitCounters** ()
- static void **UpdateMouseInput** ()

Static Public Attributes

- static int **mouseX** = 0
- static int **mouseY** = 0

Properties

- int **width** `[get]`
- int **height** `[get]`
- static SoundSystem **soundSystem** `[get]`
- int **currentFps** `[get]`
- int **targetFps** `[getset]`

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/GLContext.cs

5.20 GXPEngine.HierarchyManager Class Reference

If you are getting strange bugs because you are calling Destroy during the Update loop, you can use this class to do this more cleanly: when using HierarchyManager.Instance.LateDestroy, all these hierarchy changes will be made after the update loop is finished. You can also use HierarchyManager.Instance.LateAdd to add a game object after the update loop is finished. Similarly, you can use HierarchyManager.Instance.LateCall to postpone a certain method call until after the update loop.

Public Member Functions

- void **LateAdd** (GameObject parent, GameObject child, int index=-1)
- void **LateDestroy** (GameObject obj)
- void **LateRemove** (GameObject obj)
- bool **IsOnDestroyList** (GameObject obj)
- void **LateCall** (Action meth)
- void **UpdateHierarchy** ()

Properties

- static HierarchyManager **Instance** `[get]`

5.20.1 Detailed Description

If you are getting strange bugs because you are calling Destroy during the Update loop, you can use this class to do this more cleanly: when using HierarchyManager.Instance.LateDestroy, all these hierarchy changes will be made after the update loop is finished. You can also use HierarchyManager.Instance.LateAdd to add a game object after the update loop is finished. Similarly, you can use HierarchyManager.Instance.LateCall to postpone a certain method call until after the update loop.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Managers/HierarchyManager.cs

5.21 TiledMapParser.Image Class Reference

Public Member Functions

- override string **ToString** ()

Public Attributes

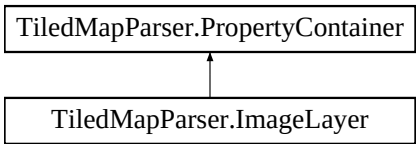
- int **Width**
- int **Height**
- string **FileName**

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.22 TiledMapParser.ImageLayer Class Reference

Inheritance diagram for TiledMapParser.ImageLayer:



Public Member Functions

- override string **ToString** ()

Public Attributes

- string **Name**
- Image **Image**
- float **offsetX** = 0
- float **offsetY** = 0
- float **Opacity** =1

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.23 GXPEngine.Input Class Reference

The Input class contains functions for reading keys and mouse

Static Public Member Functions

- static bool GetKey (int key)

  Returns 'true' if given key is down, else returns 'false'
- static bool GetKeyDown (int key)

  Returns 'true' if specified key was pressed down during the current frame
- static bool GetKeyUp (int key)

  Returns 'true' if specified key was released during the current frame
- static bool **AnyKey** ()

  Returns true if any key is currently pressed.
- static bool **AnyKeyDown** ()

  Returns true if any key was pressed down during the current frame.
- static bool GetMouseButton (int button)

  Returns 'true' if mousebutton is down, else returns 'false'
- static bool GetMouseButtonDown (int button)

  Returns 'true' if specified mousebutton was pressed down during the current frame
- static bool GetMouseButtonUp (int button)

  Returns 'true' if specified mousebutton was released during the current frame

Properties

- static int **mouseX** `[get]`

  Gets the current mouse x position in pixels.
- static int **mouseY** `[get]`

  Gets the current mouse y position in pixels.

5.23.1 Detailed Description

The Input class contains functions for reading keys and mouse

5.23.2 Member Function Documentation

5.23.2.1 GetKey()

```
static bool GXPEngine.Input.GetKey (
            int key )  [static]
```

Returns 'true' if given key is down, else returns 'false'

Parameters

key	Key number, use Key.KEYNAME or integer value.

5.23.2.2 GetKeyDown()

```
static bool GXPEngine.Input.GetKeyDown (
            int key )  [static]
```

Returns 'true' if specified key was pressed down during the current frame

Parameters

key	Key number, use Key.KEYNAME or integer value.

5.23.2.3 GetKeyUp()

```
static bool GXPEngine.Input.GetKeyUp (
            int key )  [static]
```

Returns 'true' if specified key was released during the current frame

Parameters

key	Key number, use Key.KEYNAME or integer value.

5.23.2.4 GetMouseButton()

```
static bool GXPEngine.Input.GetMouseButton (
            int button )  [static]
```

Returns 'true' if mousebutton is down, else returns 'false'

Parameters

button	Number of button: 0 = left button 1 = right button 2 = middle button

5.23.2.5 GetMouseButtonDown()

```
static bool GXPEngine.Input.GetMouseButtonDown (
            int button )  [static]
```

Returns 'true' if specified mousebutton was pressed down during the current frame

Parameters

button	Number of button: 0 = left button 1 = right button 2 = middle button

5.23.2.6 GetMouseButtonUp()

```
static bool GXPEngine.Input.GetMouseButtonUp (
            int button )  [static]
```

Returns 'true' if specified mousebutton was released during the current frame

Parameters

button	Number of button: 0 = left button 1 = right button 2 = middle button

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Utils/Input.cs

5.24 GXPEngine.Key Class Reference

Contains key definitions for usage with Input.GetKey and Input.GetKeyDown. 0.0.5: Updated keylist, courtesy of Alexandru Tâlvan

Static Public Attributes

- const int **F1** = 258

- const int **F2** = 259
- const int **F3** = 260
- const int **F4** = 261
- const int **F5** = 262
- const int **F6** = 263
- const int **F7** = 264
- const int **F8** = 265
- const int **F9** = 266
- const int **F10** = 267
- const int **F11** = 268
- const int **F12** = 269
- const int **LEFT** = 285
- const int **UP** = 283
- const int **RIGHT** = 286
- const int **DOWN** = 284
- const int **PAGE_UP** = 104
- const int **PAGE_DOWN** = 105
- const int **HOME** = 106
- const int **END** = 107
- const int **INSERT** = 108
- const int **A** = 65
- const int **B** = 66
- const int **C** = 67
- const int **D** = 68
- const int **E** = 69
- const int **F** = 70
- const int **G** = 71
- const int **H** = 72
- const int **I** = 73
- const int **J** = 74
- const int **K** = 75
- const int **L** = 76
- const int **M** = 77
- const int **N** = 78
- const int **O** = 79
- const int **P** = 80
- const int **Q** = 81
- const int **R** = 82
- const int **S** = 83
- const int **T** = 84
- const int **U** = 85
- const int **V** = 86
- const int **W** = 87
- const int **X** = 88
- const int **Y** = 89
- const int **Z** = 90
- const int **ZERO** = 48
- const int **ONE** = 49
- const int **TWO** = 50
- const int **THREE** = 51
- const int **FOUR** = 52
- const int **FIVE** = 53
- const int **SIX** = 54
- const int **SEVEN** = 55
- const int **EIGHT** = 56

- const int **NINE** = 57
- const int **ESCAPE** = 257
- const int **SPACE** = 32
- const int **ENTER** = 294
- const int **DELETE** = 127
- const int **BACKSPACE** = 295
- const int **LEFT_CTRL** = 289
- const int **RIGHT_CTRL** = 290
- const int **NUMPAD_1** = 303
- const int **NUMPAD_2** = 304
- const int **NUMPAD_3** = 305
- const int **NUMPAD_4** = 306
- const int **NUMPAD_5** = 307
- const int **NUMPAD_6** = 308
- const int **NUMPAD_7** = 309
- const int **NUMPAD_8** = 310
- const int **NUMPAD_9** = 311
- const int **MINUS** = 314
- const int **PLUS** = 315
- const int **LEFT_ALT** = 291
- const int **RIGHT_ALT** = 292
- const int **TAB** = 293
- const int **LEFT_SHIFT** = 287
- const int **RIGHT_SHIFT** = 288
- const int **TILDE** = 96
- const int **SQ_BRACKET_1** = 91
- const int **SQ_BRACKET_2** = 93
- const int **SEMICOLON** = 59
- const int **APOSTROPHE** = 39
- const int **COMMA** = 44
- const int **DOT** = 46
- const int **QUESTION_MARK** = 47

5.24.1 Detailed Description

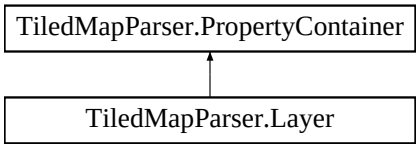
Contains key definitions for usage with Input.GetKey and Input.GetKeyDown. 0.0.5: Updated keylist, courtesy of Alexandru Tâlvan

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Utils/Key.cs

5.25 TiledMapParser.Layer Class Reference

Inheritance diagram for TiledMapParser.Layer:



Public Member Functions

- override string **ToString** ()
- short[,] GetTileArray ()

  Returns the tile data from this layer as a 2-dimensional array of shorts. It's a column-major array, so use [column,row] as indices.
- uint[,] GetTileArrayRaw ()

  Returns the tile data from this layer as a 2-dimensional array of uints. It's a column-major array, so use [column,row] as indices.

Public Attributes

- string **Name**
- int **Width**
- int **Height**
- Data **Data**

5.25.1 Member Function Documentation

5.25.1.1 GetTileArray()

```
short[,] TiledMapParser.Layer.GetTileArray ( )
```

Returns the tile data from this layer as a 2-dimensional array of shorts. It's a column-major array, so use [column,row] as indices.

This method does a lot of string parsing and memory allocation, so use it only once, during level loading.

Returns

The tile array.

5.25.1.2 GetTileArrayRaw()

```
uint[,] TiledMapParser.Layer.GetTileArrayRaw ( )
```

Returns the tile data from this layer as a 2-dimensional array of uints. It's a column-major array, so use [column,row] as indices.

Use the methods GetRotation, GetMirrorX and GetTileFrame from TiledUtils to convert the uints to rotated and mirrored animation sprites.

This method does a lot of string parsing and memory allocation, so use it only once, during level loading.

Returns

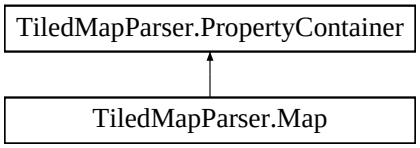
The tile array.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.26 TiledMapParser.Map Class Reference

Inheritance diagram for TiledMapParser.Map:



Public Member Functions

- override string **ToString** ()
- TileSet **GetTileSet** (int tileID)

  A helper function that returns the tile set that belongs to the tile ID read from the layer data:

Public Attributes

- int **Width**
- int **Height**
- string **Version**
- string **Orientation**
- string **RenderOrder**
- int **TileWidth**
- int **TileHeight**
- int **NextObjectId**
- TileSet[ ] **TileSets**
- Layer[ ] **Layers**
- ObjectGroup[ ] **ObjectGroups**
- ImageLayer[ ] **ImageLayers**
- string **InnerXML**

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.27 TiledMapParser.MapParser Class Reference

Call the method MapParser.ReadMap, with as argument a Tiled file exported as xml (file extension: .tmx), to get an object of type Map. This object, together with its nested objects, contains most of the information contained in the Tiled file.

Static Public Member Functions

- static Map **ReadMap** (string filename)
- static void **WriteMap** (string filename, Map map)

5.27.1 Detailed Description

Call the method MapParser.ReadMap, with as argument a Tiled file exported as xml (file extension: .tmx), to get an object of type Map. This object, together with its nested objects, contains most of the information contained in the Tiled file.

The nesting of objects mimics the structure of the Tiled file exactly. (For instance, a Map can contain multiple (tile) Layers, ObjectgroupLayers, ImageLayers, which all can have a PropertyList, etc.)

You should extend this class yourself if you want to read more information from the Tiled file (such as geometry objects). See http://docs.mapeditor.org/en/stable/reference/tmx-map-format/ for details.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.28 GXPEngine.MouseHandler Class Reference

Public Member Functions

- delegate void **OnMouseEvent** (GameObject target, MouseEventType type)
- MouseHandler (GameObject target)

  Create a new MouseHandler for the given target.

Properties

- Vector2 **offsetToTarget** `[get]`

Events

- OnMouseEvent **OnMouseDown**
- OnMouseEvent **OnMouseDownOnTarget**
- OnMouseEvent **OnMouseUp**
- OnMouseEvent **OnMouseUpOnTarget**
- OnMouseEvent **OnMouseMove**
- OnMouseEvent **OnMouseMoveOnTarget**
- OnMouseEvent **OnMouseOverTarget**
- OnMouseEvent **OnMouseOffTarget**
- OnMouseEvent **OnMouseClick**

5.28.1 Detailed Description

Simple MouseHandler interface between the Input class and your own code, that turns the Input.mouseX Input.mouseY into a more event based approach.

Do not forget to unsubscribe manually from each event before you set the handler to null.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 MouseHandler()

```
GXPEngine.MouseHandler.MouseHandler (
            GameObject target )
```

Create a new MouseHandler for the given target.

Parameters

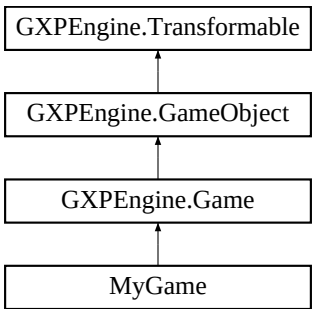
target	Target.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/MouseHandler.cs

5.29 MyGame Class Reference

Inheritance diagram for MyGame:



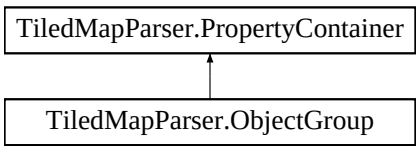
Additional Inherited Members

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/My↩ Game.cs

5.30 TiledMapParser.ObjectGroup Class Reference

Inheritance diagram for TiledMapParser.ObjectGroup:



Public Member Functions

- override string **ToString** ()

Public Attributes

- string **Name**
- TiledObject[ ] **Objects**

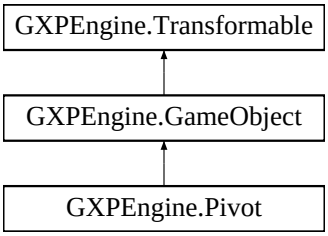
The documentation for this class was generated from the following file:

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.31 GXPEngine.Pivot Class Reference

This is an 'empty' GameObject. You can use it as a container for other sprites (parent).

Inheritance diagram for GXPEngine.Pivot:



Additional Inherited Members

5.31.1 Detailed Description

This is an 'empty' GameObject. You can use it as a container for other sprites (parent).

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Pivot.cs

5.32 TiledMapParser.Property Class Reference

Public Member Functions

- override string **ToString** ()

Public Attributes

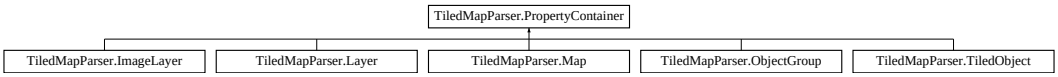
- string **Name**
- string **Type** ="string"
- string **Value**

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.33 TiledMapParser.PropertyContainer Class Reference

Inheritance diagram for TiledMapParser.PropertyContainer:



Public Member Functions

- bool **HasProperty** (string key, string type)

  Returns true if this object has a property with name [key] of type [type]. As [type], you can pass in "int", "float", "bool", "string" and "color".
- string **GetStringProperty** (string key, string defaultValue="")

  Returns the value of this object's string property with name [key], if it has such a property. Otherwise, it returns the default value that you can pass as second parameter.
- float **GetFloatProperty** (string key, float defaultValue=1)

  Returns the value of this object's float property with name [key], if it has such a property. Otherwise, it returns the default value that you can pass as second parameter.
- int **GetIntProperty** (string key, int defaultValue=1)

  Returns the value of this object's int property with name [key], if it has such a property. Otherwise, it returns the default value that you can pass as second parameter.
- bool **GetBoolProperty** (string key, bool defaultValue=false)

  Returns the value of this object's bool property with name [key], if it has such a property. Otherwise, it returns the default value that you can pass as second parameter.
- uint **GetColorProperty** (string key, uint defaultvalue=0xffffffff)

  Returns the value of this object's color property with name [key], if it has such a property. Otherwise, it returns the default value that you can pass as second parameter. The returned color can be set directly as color value of a GXPEngine Sprite.

Public Attributes

- PropertyList **propertyList**

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.34 TiledMapParser.PropertyList Class Reference

Public Member Functions

- override string **ToString** ()

Public Attributes

- Property[ ] **properties**

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.35 GXPEngine.Core.Rectangle Struct Reference

Public Member Functions

- **Rectangle** (float x, float y, float width, float height)
- override string **ToString** ()

Public Attributes

- float **x**
- float **y**
- float **width**
- float **height**

Properties

- float **left** `[get]`
- float **right** `[get]`
- float **top** `[get]`
- float **bottom** `[get]`

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/Rectangle.cs

5.36 GXPEngine.Settings Class Reference

Static class that contains various settings, such as screen resolution and player controls. In your Main method, you can Call the Settings.Load() method to initialize these settings from a text file (typically settings.txt, which should be present in the bin/Debug and/or bin/Release folder).

Static Public Member Functions

- static void **Load** ()

  Load new values from the file settings.txt

Static Public Attributes

- static string **SettingsFileName** = "settings.txt"
- static bool **ShowSettingsParsing** = false
- static bool **ThrowExceptionOnMissingSetting** = true
- static int **Width** = 800
- static int **Height** = 600
- static int **ScreenResolutionX** = 800
- static int **ScreenResolutionY** = 600
- static bool **FullScreen** = false
- static int **P1Up** =87
- static int **P1Left** =65
- static int **P1Down** =83
- static int **P1Right** =68
- static int **P1Fire1** =70
- static int **P1Fire2** =71
- static int **P1Fire3** =72
- static int **P1Fire4** =86
- static int **P1Fire5** =66
- static int **P1Fire6** =78
- static int **P2Up** =73
- static int **P2Left** =74
- static int **P2Down** =75
- static int **P2Right** =76
- static int **P2Fire1** =306
- static int **P2Fire2** =307
- static int **P2Fire3** =308
- static int **P2Fire4** =303
- static int **P2Fire5** =304
- static int **P2Fire6** =305
- static int **Start1P** =49
- static int **Start2P** =50
- static int **Menu** =294

5.36.1 Detailed Description

Static class that contains various settings, such as screen resolution and player controls. In your Main method, you can Call the Settings.Load() method to initialize these settings from a text file (typically settings.txt, which should be present in the bin/Debug and/or bin/Release folder).

The purpose is that you can easily change certain settings this way, without recompiling the code, and that during development different people can use different settings while working with the same build.

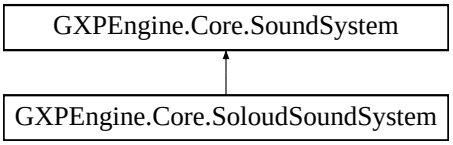
Feel free to add all kinds of other useful properties to this class. They will be initialized from the text file as long as they are of one of the following types: public static int public static float public static bool public static string public static string[]

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/Settings.cs

5.37 GXPEngine.Core.SoloudSoundSystem Class Reference

Inheritance diagram for GXPEngine.Core.SoloudSoundSystem:



Public Member Functions

- override void Init ()
- override void Deinit ()
- override IntPtr CreateStream (string filename, bool looping)
- override IntPtr LoadSound (string filename, bool looping)
- override void Step ()
- override uint PlaySound (IntPtr id, uint channelId, bool paused)
- override uint PlaySound (IntPtr id, uint channelId, bool paused, float volume, float pan)
- override float GetChannelFrequency (uint channelId)
- override void SetChannelFrequency (uint channelId, float frequency)
- override float GetChannelPan (uint channelId)
- override void SetChannelPan (uint channelId, float pan)
- override bool GetChannelPaused (uint channelId)
- override void SetChannelPaused (uint channelId, bool pause)
- override bool ChannelIsPlaying (uint channelId)
- override void StopChannel (uint channelId)
- override float GetChannelVolume (uint channelId)
- override void SetChannelVolume (uint channelId, float volume)

5.37.1 Member Function Documentation

5.37.1.1 ChannelIsPlaying()

```
override bool GXPEngine.Core.SoloudSoundSystem.ChannelIsPlaying (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.2 CreateStream()

```
override IntPtr GXPEngine.Core.SoloudSoundSystem.CreateStream (
            string filename,
            bool looping )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.3 Deinit()

```
override void GXPEngine.Core.SoloudSoundSystem.Deinit ( )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.4 GetChannelFrequency()

```
override float GXPEngine.Core.SoloudSoundSystem.GetChannelFrequency (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.5 GetChannelPan()

```
override float GXPEngine.Core.SoloudSoundSystem.GetChannelPan (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.6 GetChannelPaused()

```
override bool GXPEngine.Core.SoloudSoundSystem.GetChannelPaused (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.7 GetChannelVolume()

```
override float GXPEngine.Core.SoloudSoundSystem.GetChannelVolume (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.8 Init()

```
override void GXPEngine.Core.SoloudSoundSystem.Init ( )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.9 LoadSound()

```
override IntPtr GXPEngine.Core.SoloudSoundSystem.LoadSound (
            string filename,
            bool looping )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.10 PlaySound() [1/2]

```
override uint GXPEngine.Core.SoloudSoundSystem.PlaySound (
            IntPtr id,
            uint channelId,
            bool paused )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.11 PlaySound() [2/2]

```
override uint GXPEngine.Core.SoloudSoundSystem.PlaySound (
            IntPtr id,
            uint channelId,
            bool paused,
            float volume,
            float pan )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.12 SetChannelFrequency()

```
override void GXPEngine.Core.SoloudSoundSystem.SetChannelFrequency (
            uint channelId,
            float frequency )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.13 SetChannelPan()

```
override void GXPEngine.Core.SoloudSoundSystem.SetChannelPan (
            uint channelId,
            float pan )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.14 SetChannelPaused()

```
override void GXPEngine.Core.SoloudSoundSystem.SetChannelPaused (
            uint channelId,
            bool pause )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.15 SetChannelVolume()

```
override void GXPEngine.Core.SoloudSoundSystem.SetChannelVolume (
            uint channelId,
            float volume )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.16 Step()

```
override void GXPEngine.Core.SoloudSoundSystem.Step ( )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

5.37.1.17 StopChannel()

```
override void GXPEngine.Core.SoloudSoundSystem.StopChannel (
            uint channelId )  [virtual]
```

Implements GXPEngine.Core.SoundSystem.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/SoLoud/SoloudSoundSystem.cs

5.38 GXPEngine.Sound Class Reference

The Sound Class represents a Sound resource in memory You can load .mp3, .ogg or .wav

Public Member Functions

- Sound (String filename, bool looping=false, bool streaming=false)

  Creates a new GXPEngine.Sound. This class represents a sound file. Sound files are loaded into memory unless you set them to 'streamed'. An optional parameter allows you to create a looping sound.
- SoundChannel Play (bool paused=false, uint channelId=0, float volume=1, float pan=0)

  Play the specified paused and return the newly created SoundChannel

5.38.1 Detailed Description

The Sound Class represents a Sound resource in memory You can load .mp3, .ogg or .wav

5.38.2 Constructor & Destructor Documentation

5.38.2.1 Sound()

```
GXPEngine.Sound.Sound (
            String filename,
            bool looping = false,
            bool streaming = false )
```

Creates a new GXPEngine.Sound. This class represents a sound file. Sound files are loaded into memory unless you set them to 'streamed'. An optional parameter allows you to create a looping sound.

Parameters

filename	Filename, should include path and extension.
looping	If set to `true` the sound file repeats itself. (It loops)
streaming	If set to `true`, the file will be streamed rather than loaded into memory.
cached	If set to `true`, the sound will be stored in cache, preserving memory when creating the same sound multiple times.

5.38.3 Member Function Documentation

5.38.3.1 Play()

```
SoundChannel GXPEngine.Sound.Play (
            bool paused = false,
            uint channelId = 0,
            float volume = 1,
            float pan = 0 )
```

Play the specified paused and return the newly created SoundChannel

Parameters

paused	When set to `true`, the sound is set up, but remains paused. You can use this to set frequency, panning and volume before playing the sound.
channel↩ Id	When in range 0...31, the selected channel will be used. If it already contains a playing sound, that sound will be stopped. When set to -1 (the default), the next free channel will be used. However, when all channels are in use, Sound.Play will silently fail.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Sound.cs

5.39 GXPEngine.SoundChannel Class Reference

This class represents a sound channel on the soundcard.

Public Member Functions

- **SoundChannel** (uint id)
- void **Stop** ()

  Stop the channel.

Properties

- uint **ID** `[get]`
- float Frequency `[getset]`

  Gets or sets the channel frequency.
- bool Mute `[getset]`

  Gets or sets a value indicating whether this GXPEngine.SoundChannel is mute.
- float **Pan** `[getset]`

  Gets or sets the pan. Value should be in range -1..0..1, for left..center..right
- bool IsPaused `[getset]`

  Gets or sets a value indicating whether this GXPEngine.Channel is paused.
- bool IsPlaying `[get]`

  Gets a value indicating whether this GXPEngine.Channel is playing. (readonly)
- float Volume `[getset]`

  Gets or sets the volume. Should be in range 0...1

5.39.1 Detailed Description

This class represents a sound channel on the soundcard.

5.39.2 Property Documentation

5.39.2.1 Frequency

```
float GXPEngine.SoundChannel.Frequency  [get], [set]
```

Gets or sets the channel frequency.

The frequency. Defaults to the sound frequency. (Usually 44100Hz)

5.39.2.2 IsPaused

```
bool GXPEngine.SoundChannel.IsPaused  [get], [set]
```

Gets or sets a value indicating whether this GXPEngine.Channel is paused.

`true` if paused; otherwise, `false`.

5.39.2.3 IsPlaying

```
bool GXPEngine.SoundChannel.IsPlaying  [get]
```

Gets a value indicating whether this GXPEngine.Channel is playing. (readonly)

`true` if playing; otherwise, `false`.

5.39.2.4 Mute

```
bool GXPEngine.SoundChannel.Mute  [get], [set]
```

Gets or sets a value indicating whether this GXPEngine.SoundChannel is mute.

`true` if you want to mute the sound

5.39.2.5 Volume

```
float GXPEngine.SoundChannel.Volume  [get], [set]
```

Gets or sets the volume. Should be in range 0...1

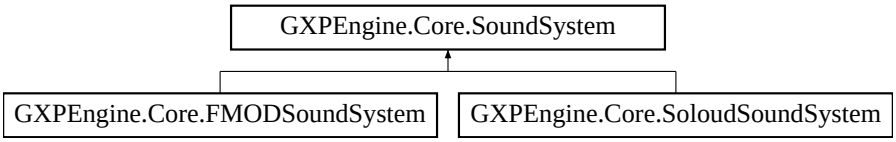
The volume.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/SoundChannel.cs

5.40 GXPEngine.Core.SoundSystem Class Reference

Inheritance diagram for GXPEngine.Core.SoundSystem:



Public Member Functions

- abstract void **Init** ()
- abstract void **Deinit** ()
- abstract IntPtr **CreateStream** (string filename, bool looping)
- abstract IntPtr **LoadSound** (string filename, bool looping)
- abstract void **Step** ()
- abstract uint **PlaySound** (IntPtr id, uint channelId, bool paused)
- abstract uint **PlaySound** (IntPtr id, uint channelId, bool paused, float volume, float pan)
- abstract float **GetChannelFrequency** (uint channelId)
- abstract void **SetChannelFrequency** (uint channelId, float frequency)
- abstract float **GetChannelPan** (uint channelId)
- abstract void **SetChannelPan** (uint channelId, float pan)
- abstract float **GetChannelVolume** (uint channelId)
- abstract void **SetChannelVolume** (uint channelId, float volume)
- abstract bool **GetChannelPaused** (uint channelId)
- abstract void **SetChannelPaused** (uint channelId, bool pause)
- abstract bool **ChannelIsPlaying** (uint channelId)
- abstract void **StopChannel** (uint channelId)

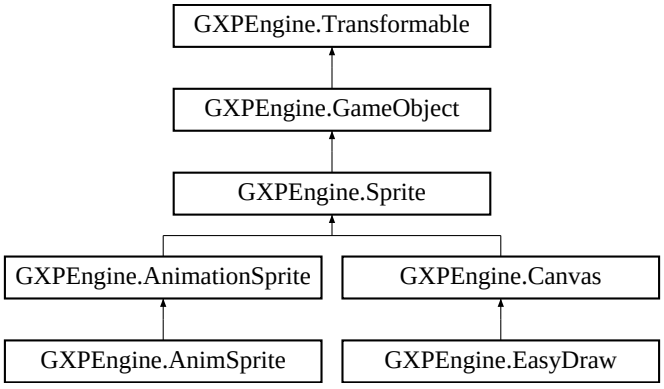
The documentation for this class was generated from the following file:

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/SoundSystem.cs

5.41 GXPEngine.Sprite Class Reference

The Sprite class holds 2D images that can be used as objects in your game.

Inheritance diagram for GXPEngine.Sprite:



Public Member Functions

- Sprite (System.Drawing.Bitmap bitmap, bool addCollider=true)

  Initializes a new instance of the GXPEngine.Sprite class. Specify a System.Drawing.Bitmap to use. Bitmaps will not be cached.
- **Sprite** (Texture2D texture, bool addCollider=true)
- Sprite (string filename, bool keepInCache=false, bool addCollider=true)

  Initializes a new instance of the GXPEngine.Sprite class. Specify an image file to load. Please use a full filename. Initial path is the application folder. Images will be cached internally. That means once it is loaded, the same data will be used when you load the file again.
- float[ ] **GetUVs** (bool safe=true)
- Vector2[ ] GetExtents ()

  Gets the four corners of this object as a set of 4 Vector2s.
- void SetOrigin (float x, float y)

  Sets the origin, the pivot point of this Sprite in pixels.
- void Mirror (bool mirrorX, bool mirrorY)

  This function can be used to enable mirroring for the sprite in either x or y direction.
- void SetColor (float r, float g, float b)

  Sets the color of the sprite.

Public Attributes

- BlendMode **blendMode** = null

Protected Member Functions

- override void OnDestroy ()

  Subclasses can implement this method to clean up resources once on destruction. Will be called by the engine when the game object is destroyed.
- void **initializeFromTexture** (Texture2D texture)
- virtual void **setUVs** ()
- override Collider createCollider ()

  Create and return a collider to use for this game object. Null is allowed.
- override void RenderSelf (GLContext glContext)

Protected Attributes

- Texture2D **_texture**
- Rectangle **_bounds**
- float[ ] **_uvs**
- bool **_mirrorX** = false
- bool **_mirrorY** = false

Properties

- Texture2D **texture** `[get]`

  Returns the texture that is used to create this sprite. If no texture is used, null will be returned. Use this to retreive the original width/height or filename of the texture.
- virtual int **width** `[getset]`

  Gets or sets the sprite's width in pixels.
- virtual int **height** `[getset]`

  Gets or sets the sprite's height in pixels.
- uint **color** `[getset]`

  Gets or sets the color filter for this sprite. This can be any value between 0x000000 and 0xFFFFFF.
- float **alpha** `[getset]`

  Gets or sets the alpha value of the sprite. Setting this value allows you to make the sprite (semi-)transparent. The alpha value should be in the range 0...1, where 0 is fully transparent and 1 is fully opaque.

5.41.1 Detailed Description

The Sprite class holds 2D images that can be used as objects in your game.

5.41.2 Constructor & Destructor Documentation

5.41.2.1 Sprite() `[1/2]`

```
GXPEngine.Sprite.Sprite (
            System.Drawing.Bitmap bitmap,
            bool addCollider = true )
```

Initializes a new instance of the GXPEngine.Sprite class. Specify a System.Drawing.Bitmap to use. Bitmaps will not be cached.

Parameters

bitmap	Bitmap.
addCollider	If `true`, this sprite will have a collider that will be added to the collision manager.

5.41.2.2 Sprite() `[2/2]`

```
GXPEngine.Sprite.Sprite (
            string filename,
            bool keepInCache = false,
            bool addCollider = true )
```

Initializes a new instance of the GXPEngine.Sprite class. Specify an image file to load. Please use a full filename. Initial path is the application folder. Images will be cached internally. That means once it is loaded, the same data will be used when you load the file again.

Parameters

filename	The name of the file that should be loaded.
keepInCache	If `true`, the sprite's texture will be kept in memory for the entire lifetime of the game. This takes up more memory, but removes load times.
addCollider	If `true`, this sprite will have a collider that will be added to the collision manager.

5.41.3 Member Function Documentation

5.41.3.1 createCollider()

```
override Collider GXPEngine.Sprite.createCollider ( )  [protected], [virtual]
```

Create and return a collider to use for this game object. Null is allowed.

Reimplemented from GXPEngine.GameObject.

5.41.3.2 GetExtents()

```
Vector2[] GXPEngine.Sprite.GetExtents ( )
```

Gets the four corners of this object as a set of 4 Vector2s.

Returns

The extents.

5.41.3.3 Mirror()

```
void GXPEngine.Sprite.Mirror (
            bool mirrorX,
            bool mirrorY )
```

This function can be used to enable mirroring for the sprite in either x or y direction.

Parameters

mirrorX	If set to `true` to enable mirroring in x direction.
mirrorY	If set to `true` to enable mirroring in y direction.

5.41.3.4 OnDestroy()

```
override void GXPEngine.Sprite.OnDestroy ( )  [protected], [virtual]
```

Subclasses can implement this method to clean up resources once on destruction. Will be called by the engine when the game object is destroyed.

Reimplemented from GXPEngine.GameObject.

5.41.3.5 RenderSelf()

```
override void GXPEngine.Sprite.RenderSelf (
            GLContext glContext )  [protected], [virtual]
```

Reimplemented from GXPEngine.GameObject.

5.41.3.6 SetColor()

```
void GXPEngine.Sprite.SetColor (
            float r,
            float g,
            float b )
```

Sets the color of the sprite.

Parameters

r	The red component, range 0..1
g	The green component, range 0..1
b	The blue component, range 0..1

5.41.3.7 SetOrigin()

```
void GXPEngine.Sprite.SetOrigin (
            float x,
            float y )
```

Sets the origin, the pivot point of this Sprite in pixels.

Parameters

x	The x coordinate.
y	The y coordinate.

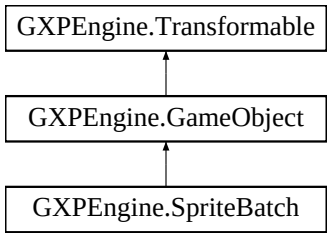
The documentation for this class was generated from the following file:

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Sprite.cs

5.42 GXPEngine.SpriteBatch Class Reference

A SpriteBatch is a GameObject that can be used to render many sprites efficiently, and change the color, alpha and blend mode of all of those sprites simultaneously. Usage: Add any number of sprites as child to a sprite batch, and then call the Freeze method. Note that this will destroy the individual sprites, so there won't be colliders for them anymore, and the position and rotation of individual sprites cannot be changed anymore.

Inheritance diagram for GXPEngine.SpriteBatch:



Public Member Functions

- **SpriteBatch** ()

  Create a new SpriteBatch game object. After adding sprites as child to this game object, call the Freeze method to started batched drawing.
- void **Freeze** ()

  Call this method after adding sprites as child to this game object, and positioning and rotating them correctly. This will destroy the individual sprites and their colliders. Note that the individual color, alpha and blend mode of those sprites is forgotten: use the color, alpha and blend mode of the sprite batch instead.
- Vector2[ ] GetExtents ()

  Gets the four corners of this object as a set of 4 Vector2s.
- void SetColor (float r, float g, float b)

  Sets the color of the sprite batch.

Public Attributes

- BlendMode **blendMode** = null

Protected Member Functions

- override void OnDestroy ()

  Subclasses can implement this method to clean up resources once on destruction. Will be called by the engine when the game object is destroyed.
- override void RenderSelf (GLContext glContext)

Protected Attributes

- uint **_color** = 0xFFFFFF
- float **_alpha** = 1.0f

Properties

- uint **color** `[getset]`

  Gets or sets the color filter for this sprite. This can be any value between 0x000000 and 0xFFFFFF.
- float **alpha** `[getset]`

  Gets or sets the alpha value of the sprite batch. Setting this value allows you to make the sprite batch (semi-)transparent. The alpha value should be in the range 0...1, where 0 is fully transparent and 1 is fully opaque.

5.42.1 Detailed Description

A SpriteBatch is a GameObject that can be used to render many sprites efficiently, and change the color, alpha and blend mode of all of those sprites simultaneously. Usage: Add any number of sprites as child to a sprite batch, and then call the Freeze method. Note that this will destroy the individual sprites, so there won't be colliders for them anymore, and the position and rotation of individual sprites cannot be changed anymore.

5.42.2 Member Function Documentation

5.42.2.1 GetExtents()

```
Vector2[] GXPEngine.SpriteBatch.GetExtents ( )
```

Gets the four corners of this object as a set of 4 Vector2s.

Returns

The extents.

5.42.2.2 OnDestroy()

```
override void GXPEngine.SpriteBatch.OnDestroy ( )  [protected], [virtual]
```

Subclasses can implement this method to clean up resources once on destruction. Will be called by the engine when the game object is destroyed.

Reimplemented from GXPEngine.GameObject.

5.42.2.3 RenderSelf()

```
override void GXPEngine.SpriteBatch.RenderSelf (
            GLContext glContext )  [protected], [virtual]
```

Reimplemented from GXPEngine.GameObject.

5.42.2.4 SetColor()

```
void GXPEngine.SpriteBatch.SetColor (
            float r,
            float g,
            float b )
```

Sets the color of the sprite batch.

Parameters

r	The red component, range 0..1
g	The green component, range 0..1
b	The blue component, range 0..1

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/SpriteBatch.cs

5.43 TiledMapParser.Text Class Reference

Public Member Functions

- override string **ToString** ()

Public Attributes

- string **font**
- int **wrap** =0
- int **bold** =0
- int **italic** =0
- int **fontSize** = 16
- string **text**
- string **horizontalAlign** ="left"
- string **verticalAlign** ="top"
- string **color** ="#FF000000"

Properties

- uint **Color** `[get]`

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.44 GXPEngine.Core.Texture2D Class Reference

Public Member Functions

- **Texture2D** (int width, int height)
- **Texture2D** (string filename)
- **Texture2D** (Bitmap bitmap)
- void **Dispose** ()
- void **Bind** ()
- void **Unbind** ()
- void **UpdateGLTexture** ()
- Texture2D **Clone** (bool deepCopy=false)

Static Public Member Functions

- static Texture2D **GetInstance** (string filename, bool keepInCache=false)
- static void **RemoveInstance** (string filename)
- static string **GetDiagnostics** ()

Properties

- Bitmap **bitmap** `[get]`
- string **filename** `[get]`
- int **width** `[get]`
- int **height** `[get]`
- bool?? **wrap** `[set]`

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/Texture2D.cs

5.45 TiledMapParser.TiledLoader Class Reference

A class for automatically creating GXPEngine sprites from Tiled files.

Public Member Functions

- delegate void **ObjectCreateCallback** (Sprite sprite, TiledObject obj)
- TiledLoader (string filename, GameObject rootObject=null, bool addColliders=true, float defaultOriginX=0.5f, float defaultOriginY=0.5f, bool highQualityText=true, bool autoInstance=false, ObjectCreateCallback callback=null)

  Creates a new TiledLoader and loads the Tiled file given by filename. (The path should be relative to the current folder, typically bin/Debug or bin/Release.) Sets various public state variables for the Tiled loader. Call LoadTileLayers, LoadImageLayers and LoadObjectLayers to create GXPEngine sprites from the layers in the given Tiled file.
- void AddManualType (params string[] typeNames)

  Register names to the list of manually created objects. When a Tiled Object with such a type is found, this loader will not create a sprite, but only fire an event, such that you can create a specific type of game object yourself.
- void LoadObjectGroups (params int[] layerIndices)

  Creates animation sprites and text sprites from the given object groups from the loaded Tiled file. If no object group indices are given, all object groups are loaded. Uses the current settings from TiledLoader (rootObject, addColliders, defaultOrigins, highQualityText, autoInstance). Subscribe to the OnObjectCreated event to get a callback for each created sprite.
- void LoadTileLayers (params int[] layerIndices)

  Creates animation sprites for the given tile layers from the loaded Tiled file. If no tile layer indices are given, all tile layers are loaded. Uses the current settings from TiledLoader (rootObject, addColliders, defaultOrigins).
- void LoadImageLayers (params int[] layerIndices)

  Creates sprites for the given image layers from the loaded Tiled file. If no image layer indices are given, all image layers are loaded. Uses the current settings from TiledLoader (rootObject, addColliders, defaultOrigins).

Static Public Member Functions

- static EasyDraw CreateTextField (TiledObject obj, bool addCollider=true, bool highQualityText=true)

  Creates an EasyDraw for displaying text, based on the configuration parameters of a Tiled object. (Including font, text alignment, color)
- static void DrawText (EasyDraw textCanvas, string text)

  Draws a given textstring to an EasyDraw, following Tiled's text alignment conventions. (It uses the text alignment settings of the textCanvas)
- static void ChangeOrigin (Sprite spr, float newOriginRelativeX=0.5f, float newOriginRelativeY=0.5f, float oldOriginX=0, float oldOriginY=0)

  Changes the origin of a (possibly scaled and rotated) sprite, without changing the position.
- static void SetPositionRotationScaleOrigin (Sprite newSprite, TiledObject obj, float normalizedOriginX=0.5f, float normalizedOriginY=0.5f)

  Sets the position, rotation and scale of a sprite based on the data in a TiledObject, and then sets the origin of the sprite.

Public Attributes

- readonly Map **map**
- GameObject **rootObject**

  All generated objects will be added as child of this object. If null, game will be used as parent object.
- bool **addColliders**

  Whether the newly generated sprites will have colliders.
- bool **autoInstance**

  Set this to true if this TiledLoader should automatically create instances of a custom classes, when LoadObjectLayers is called. In this case, the [Type] value of the Tiled object is used as class name (case sensitive!). For image objects this requires that your class inherits from AnimationSprite, and has a constructor with parameters (string imageFile, int columns, int rows, TiledObject object). For shape objects this requires that your class inherits from Sprite, and has a constructor with parameters (TiledObject object). For text objects this option is ignored. Your class should be in the global namespace. If false, basic AnimationSprites are created.
- float **defaultOriginX**

  The x origin for all newly generated sprites.
- float **defaultOriginY**

  The y origin for all newly generated sprites.
- bool **highQualityText**

  Whether text elements will use high quality antialiasing (at the cost of about 4x more memory usage).

Properties

- int **NumTileLayers** `[get]`

  Returns the number of tile layers in the loaded map.
- int **NumImageLayers** `[get]`

  Returns the number of image layers in the loaded map.
- int **NumObjectGroups** `[get]`

  Returns the number of object groups in the loaded map.

Events

- ObjectCreateCallback **OnObjectCreated**

  This event is fired for each Tiled Object while reading object layers from the Tiled file. It is fired whenever an AnimationSprite or text element (EasyDraw) is generated from a Tiled object. It is fired with sprite=null whenever a Tiled object without text or image data is read, or when a Tiled object with one of the types in the manualObjects list is read (use this in combination with SetPositionRotationScaleOrigin to create your own objects such as a player that inherits from Sprite).

5.45.1 Detailed Description

A class for automatically creating GXPEngine sprites from Tiled files.

5.45.2 Constructor & Destructor Documentation

5.45.2.1 TiledLoader()

```
TiledMapParser.TiledLoader.TiledLoader (
            string filename,
            GameObject rootObject = null,
            bool addColliders = true,
            float defaultOriginX = 0.5f,
            float defaultOriginY = 0.5f,
            bool highQualityText = true,
            bool autoInstance = false,
            ObjectCreateCallback callback = null )
```

Creates a new TiledLoader and loads the Tiled file given by filename. (The path should be relative to the current folder, typically bin/Debug or bin/Release.) Sets various public state variables for the Tiled loader. Call LoadTile↩ Layers, LoadImageLayers and LoadObjectLayers to create GXPEngine sprites from the layers in the given Tiled file.

Parameters

filename	the name of the Tiled file (including .tmx extension).
rootObject	start value for rootObject.
addColliders	start value for addColliders.
defaultOriginX	start value for defaultOriginX.
defaultOriginY	start value for defaultOriginY.
highQualityText	start value for highQualityText.
autoInstance	start value for autoInstance.
callback	A method to be called by the OnObjectCreated event.

5.45.3 Member Function Documentation

5.45.3.1 AddManualType()

```
void TiledMapParser.TiledLoader.AddManualType (
            params string[] typeNames )
```

Register names to the list of manually created objects. When a Tiled Object with such a type is found, this loader will not create a sprite, but only fire an event, such that you can create a specific type of game object yourself.

Parameters

typeName	

5.45.3.2 ChangeOrigin()

```
static void TiledMapParser.TiledLoader.ChangeOrigin (
            Sprite spr,
```

```
            float newOriginRelativeX = 0.5f,
            float newOriginRelativeY = 0.5f,
            float oldOriginX = 0,
            float oldOriginY = 0 )  [static]
```

Changes the origin of a (possibly scaled and rotated) sprite, without changing the position.

Parameters

spr	The sprite
newOriginRelativeX	The new x origin, normalized (typically a value between 0 and 1)
newOriginRelativeY	The new y origin, normalized (typically a value between 0 and 1)
oldOriginX	The old x origin, normalized (typically a value between 0 and 1; GXP default is 0)
oldOriginY	The old y origin, normalized (typically a value between 0 and 1; GXP default is 0)

5.45.3.3 CreateTextField()

```
static EasyDraw TiledMapParser.TiledLoader.CreateTextField (
            TiledObject obj,
            bool addCollider = true,
            bool highQualityText = true )  [static]
```

Creates an EasyDraw for displaying text, based on the configuration parameters of a Tiled object. (Including font, text alignment, color)

Parameters

obj	The Tiled (text) object

Returns

5.45.3.4 DrawText()

```
static void TiledMapParser.TiledLoader.DrawText (
            EasyDraw textCanvas,
            string text )  [static]
```

Draws a given textstring to an EasyDraw, following Tiled's text alignment conventions. (It uses the text alignment settings of the textCanvas)

Parameters

textCanvas	the EasyDraw for displaying text, for instance as created by CreateTextField
text	The text to display. Note that text is not automatically wrapped, so include manual newline symbols ( )

5.45.3.5 LoadImageLayers()

```
void TiledMapParser.TiledLoader.LoadImageLayers (
            params int[] layerIndices )
```

Creates sprites for the given image layers from the loaded Tiled file. If no image layer indices are given, all image layers are loaded. Uses the current settings from TiledLoader (rootObject, addColliders, defaultOrigins).

Parameters

layerIndices	The indices of the image layers that should be loaded.

5.45.3.6 LoadObjectGroups()

```
void TiledMapParser.TiledLoader.LoadObjectGroups (
            params int[] layerIndices )
```

Creates animation sprites and text sprites from the given object groups from the loaded Tiled file. If no object group indices are given, all object groups are loaded. Uses the current settings from TiledLoader (rootObject, add↩ Colliders, defaultOrigins, highQualityText, autoInstance). Subscribe to the OnObjectCreated event to get a callback for each created sprite.

Parameters

layerIndices	The indices of the object layers that should be loaded.

5.45.3.7 LoadTileLayers()

```
void TiledMapParser.TiledLoader.LoadTileLayers (
            params int[] layerIndices )
```

Creates animation sprites for the given tile layers from the loaded Tiled file. If no tile layer indices are given, all tile layers are loaded. Uses the current settings from TiledLoader (rootObject, addColliders, defaultOrigins).

Parameters

layerIndices	The indices of the tile layers that should be loaded.

5.45.3.8 SetPositionRotationScaleOrigin()

```
static void TiledMapParser.TiledLoader.SetPositionRotationScaleOrigin (
            Sprite newSprite,
            TiledObject obj,
            float normalizedOriginX = 0.5f,
            float normalizedOriginY = 0.5f )  [static]
```

Sets the position, rotation and scale of a sprite based on the data in a TiledObject, and then sets the origin of the sprite.

Parameters

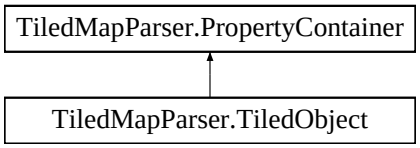
newSprite	The sprite to be changed
obj	The Tiled Object with the transform data
normalizedOriginX	the new x-origin, normalized (=typically between 0 and 1)
normalizedOriginY	the new y-origin, normalized (=typically between 0 and 1)

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledLoader.cs

5.46 TiledMapParser.TiledObject Class Reference

Inheritance diagram for TiledMapParser.TiledObject:



Public Member Functions

- void **Initialize** ()

  Call this method to initialize the MirrorX, MirrorY and ImageID fields. (The GID value read from the file encodes all of this information.)
- override string **ToString** ()

Public Attributes

- int **ID**
- uint **GID** =0xffffffff
- int **ImageID** =-1
- bool **MirrorX** = false
- bool **MirrorY** = false
- float **Rotation** = 0

- string **Name**
- string **Type**
- float **Width**
- float **Height**
- float **X**
- float **Y**
- Text **textField**

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.47 TiledMapParser.TiledUtils Class Reference

Static Public Member Functions

- static uint **GetColor** (string htmlColor)

  This translates a Tiled color string to a uint that can be used as a GXPEngine Sprite color.
- static bool **GetMirrorX** (uint tileID)

  This method converts a raw tile number read from a tile layer to a mirrorX value.
- static float **GetRotation** (uint tileID)

  This method converts a raw tile number read from a tile layer to a rotation value in degrees.
- static int **GetTileFrame** (uint tileID)

  This method converts a raw tile number read from a tile layer to a pure image ID, that can be used with Map.GetTileSet.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.48 TiledMapParser.TileSet Class Reference

Public Member Functions

- override string **ToString** ()

Public Attributes

- int **TileWidth**
- int **TileHeight**
- int **TileCount**
- int **Columns**
- int FirstGId

  This is the number of the first tile. Usually 1 (so 0 means empty/no tile).
- string **Name**
- Image **Image**

Properties

- int **Rows** `[get]`

5.48.1 Member Data Documentation

5.48.1.1 FirstGId

```
int TiledMapParser.TileSet.FirstGId
```

This is the number of the first tile. Usually 1 (so 0 means empty/no tile).

When multiple tilesets are used, this is the total number of previous tiles + 1.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/TiledMapParser.cs

5.49 GXPEngine.Time Class Reference

Contains various time related functions.

Properties

- static int **now** `[get]`

  Returns the current system time in milliseconds
- static int time `[get]`

  Returns this time in milliseconds since the program started
- static int **deltaTime** `[get]`

5.49.1 Detailed Description

Contains various time related functions.

5.49.2 Property Documentation

5.49.2.1 time

```
int GXPEngine.Time.time  [static], [get]
```

Returns this time in milliseconds since the program started

The time.

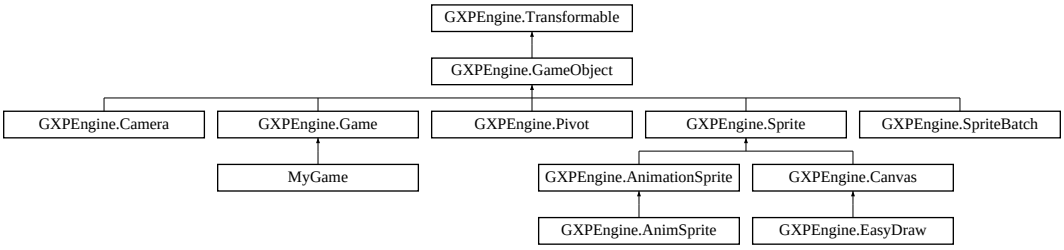
The documentation for this class was generated from the following file:

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Utils/Time.cs

5.50 GXPEngine.Transformable Class Reference

The Transformable class contains all positional data of GameObjects.

Inheritance diagram for GXPEngine.Transformable:



Public Member Functions

- void SetXY (float x, float y)

  Sets the X and Y position.
- virtual Vector2 InverseTransformPoint (float x, float y)

  Transforms the point from the game's global space to this object's local space.
- virtual Vector2 InverseTransformDirection (float x, float y)

  Transforms the direction vector (x,y) from the game's global space to this object's local space. This means that rotation and scaling is applied, but translation is not.
- float **DistanceTo** (Transformable other)

  Returns the distance to another Transformable
- virtual Vector2 TransformPoint (float x, float y)

  Transforms the point from this object's local space to the game's global space.
- virtual Vector2 TransformDirection (float x, float y)

  Transforms a direction vector (x,y) from this object's local space to the game's global space. This means that rotation and scaling is applied, but translation is not.
- void Turn (float angle)

  Turn the specified object with a certain angle in degrees.
- void Move (float stepX, float stepY)

  Move the object, based on its current rotation.
- void Translate (float stepX, float stepY)

  Move the object, in world space. (Object rotation is ignored)
- void SetScaleXY (float scaleX, float scaleY)

  Sets the object's scaling
- void SetScaleXY (float scale)

  Sets the object's scaling
- Transformable **Inverse** ()

  Returns the inverse matrix transformation, if it exists. (Use this e.g. for cameras used by sub windows)

Protected Attributes

- float[ ] _matrix
- float **_rotation** = 0.0f
- float **_scaleX** = 1.0f
- float **_scaleY** = 1.0f

Properties

- float[ ] matrix `[get]`

  Returns the gameobject's 4x4 matrix.
- float x `[getset]`

  Gets or sets the x position.
- float y `[getset]`

  Gets or sets the y position.
- float rotation `[getset]`

  Gets or sets the object's rotation in degrees.
- float scaleX `[getset]`

  Sets the object's x-axis scale
- float scaleY `[getset]`

  Sets the object's y-axis scale
- float scale `[getset]`

  Sets the object's x-axis and y-axis scale Note: This getter/setter is included for convenience only Reading this value will return scaleX, not scaleY!!

5.50.1 Detailed Description

The Transformable class contains all positional data of GameObjects.

5.50.2 Member Function Documentation

5.50.2.1 InverseTransformDirection()

```
virtual Vector2 GXPEngine.Transformable.InverseTransformDirection (
            float x,
            float y )  [virtual]
```

Transforms the direction vector (x,y) from the game's global space to this object's local space. This means that rotation and scaling is applied, but translation is not.

Reimplemented in GXPEngine.GameObject.

5.50.2.2 InverseTransformPoint()

```
virtual Vector2 GXPEngine.Transformable.InverseTransformPoint (
            float x,
            float y )  [virtual]
```

Transforms the point from the game's global space to this object's local space.

Returns

The point.

Parameters

x	The x coordinate.
y	The y coordinate.

Reimplemented in GXPEngine.GameObject.

5.50.2.3 Move()

```
void GXPEngine.Transformable.Move (
            float stepX,
            float stepY )
```

Move the object, based on its current rotation.

Parameters

stepX	Step x.
stepY	Step y.

5.50.2.4 SetScaleXY() [1/2]

```
void GXPEngine.Transformable.SetScaleXY (
            float scale )
```

Sets the object's scaling

Parameters

scale	Scale x and y.

5.50.2.5 SetScaleXY() [2/2]

```
void GXPEngine.Transformable.SetScaleXY (
            float scaleX,
            float scaleY )
```

Sets the object's scaling

Parameters

scaleX	Scale x.
scaleY	Scale y.

5.50.2.6 SetXY()

```
void GXPEngine.Transformable.SetXY (
            float x,
            float y )
```

Sets the X and Y position.

Parameters

x	The x coordinate.
y	The y coordinate.

5.50.2.7 TransformDirection()

```
virtual Vector2 GXPEngine.Transformable.TransformDirection (
            float x,
            float y )  [virtual]
```

Transforms a direction vector (x,y) from this object's local space to the game's global space. This means that rotation and scaling is applied, but translation is not.

Reimplemented in GXPEngine.GameObject.

5.50.2.8 TransformPoint()

```
virtual Vector2 GXPEngine.Transformable.TransformPoint (
            float x,
            float y )  [virtual]
```

Transforms the point from this object's local space to the game's global space.

Returns

The point.

Parameters

x	The x coordinate.
y	The y coordinate.

Reimplemented in GXPEngine.GameObject.

5.50.2.9 Translate()

```
void GXPEngine.Transformable.Translate (
            float stepX,
            float stepY )
```

Move the object, in world space. (Object rotation is ignored)

Parameters

stepX	Step x.
stepY	Step y.

5.50.2.10 Turn()

```
void GXPEngine.Transformable.Turn (
            float angle )
```

Turn the specified object with a certain angle in degrees.

Parameters

angle	Angle.

5.50.3 Member Data Documentation

5.50.3.1 _matrix

```
float [] GXPEngine.Transformable._matrix  [protected]
```

Initial value:

```
= new float[16] {
            1.0f, 0.0f, 0.0f, 0.0f,
            0.0f, 1.0f, 0.0f, 0.0f,
            0.0f, 0.0f, 1.0f, 0.0f,
            0.0f, 0.0f, 0.0f, 1.0f }
```

5.50.4 Property Documentation

5.50.4.1 matrix

```
float [] GXPEngine.Transformable.matrix  [get]
```

Returns the gameobject's 4x4 matrix.

The matrix.

5.50.4.2 rotation

```
float GXPEngine.Transformable.rotation  [get], [set]
```

Gets or sets the object's rotation in degrees.

The rotation.

5.50.4.3 scale

```
float GXPEngine.Transformable.scale  [get], [set]
```

Sets the object's x-axis and y-axis scale Note: This getter/setter is included for convenience only Reading this value will return scaleX, not scaleY!!

The scale.

5.50.4.4 scaleX

```
float GXPEngine.Transformable.scaleX  [get], [set]
```

Sets the object's x-axis scale

The scale x.

5.50.4.5 scaleY

```
float GXPEngine.Transformable.scaleY  [get], [set]
```

Sets the object's y-axis scale

The scale y.

5.50.4.6 x

```
float GXPEngine.Transformable.x  [get], [set]
```

Gets or sets the x position.

The x.

5.50.4.7 y

```
float GXPEngine.Transformable.y  [get], [set]
```

Gets or sets the y position.

The y.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/Transformable.cs

5.51 GXPEngine.Managers.UpdateManager Class Reference

Public Member Functions

- void **Step** ()
- void **Add** (GameObject gameObject)
- Boolean **Contains** (GameObject gameObject)
- void **Remove** (GameObject gameObject)
- string **GetDiagnostics** ()

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Managers/UpdateManager.cs

5.52 GXPEngine.Core.Vector2 Struct Reference

Public Member Functions

- **Vector2** (float x, float y)
- override string **ToString** ()

Public Attributes

- float **x**
- float **y**

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/Vector2.cs

5.53 GXPEngine.Window Class Reference

A class that can be used to create "sub windows" (e.g. mini-map, splitscreen, etc). This is not a gameobject. Instead, subscribe the RenderWindow method to the main game's OnAfterRender event.

Public Member Functions

- **Window** (int x, int y, int width, int height, GameObject camera)

  Creates a render window in the rectangle given by x,y,width,height. The camera determines the focal point, rotation and scale of this window.
- void **RenderWindow** (GLContext glContext)

  To render the scene in this window, subscribe this method to the main game's OnAfterRender event.

Public Attributes

- GameObject **camera**

  The game object (which should be in the hierarchy!) that determines the focus point, rotation and scale of the viewport window.

Properties

- int **windowX** `[getset]`

  The x coordinate of the window's left side
- int **windowY** `[getset]`

  The y coordinate of the window's top
- float **centerX** `[get]`

  The x coordinate of the window center
- float **centerY** `[get]`

  The y coordinate of the window center
- int **width** `[getset]`

  The window's width
- int **height** `[getset]`

  The window's height

5.53.1 Detailed Description

A class that can be used to create "sub windows" (e.g. mini-map, splitscreen, etc). This is not a gameobject. Instead, subscribe the RenderWindow method to the main game's OnAfterRender event.

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/AddOns/Window.cs

5.54 GXPEngine.Core.WindowSize Class Reference

Public Attributes

- int **width**
- int **height**

Static Public Attributes

- static WindowSize **instance** = new WindowSize()

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

- C:/Users/Saxion/Documents/Courses/GameProg/GXPEngine2022BB/GXPEngine2022BB/GXPEngine/↩ GXPEngine/Core/GLContext.cs

Index