

Namespace Inertia

Classes

[INERTIA](#)

2D physics simulation.

Class INERTIA

Namespace: [Inertia](#)

Assembly: Inertia.dll

2D physics simulation.

```
public class INERTIA : OBJECT
```

Inheritance

[object](#) ↴ ← [OBJECT](#) ← INERTIA

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↴ , [object.Equals\(object, object\)](#) ↴ , [object.GetHashCode\(\)](#) ↴ ,
[object.GetType\(\)](#) ↴ , [object.MemberwiseClone\(\)](#) ↴ , [object.ReferenceEquals\(object, object\)](#) ↴ ,
[object.ToString\(\)](#) ↴

Methods

ADDFORCE(int, double, double)

```
public void ADDFORCE(int id, double x, double y)
```

Parameters

`id` [int](#) ↴

`x` [double](#) ↴

`y` [double](#) ↴

CREATESPHERE(double, double, double, double)

```
public int CREATESPHERE(double _1, double _2, double _3, double _4)
```

Parameters

_1 [double](#)

_2 [double](#)

_3 [double](#)

_4 [double](#)

Returns

[int](#)

DELETEBODY(int)

```
public void DELETEBODY(int id)
```

Parameters

id [int](#)

GETPOSITIONX(int)

```
public double GETPOSITIONX(int id)
```

Parameters

id [int](#)

Returns

[double](#)

GETPOSITIONY(int)

```
public double GETPOSITIONY(int id)
```

Parameters

id [int](#)

Returns

[double](#)

GETSPEED(int)

```
public double GETSPEED(int id)
```

Parameters

id [int](#)

Returns

[double](#)

LINK(int, string, bool, bool)

```
public void LINK(int id, string object_name, bool _1, bool _2)
```

Parameters

id [int](#)

object_name [string](#)

_1 [bool](#)

_2 [bool](#)

LOAD(string)

```
public void LOAD(string filename)
```

Parameters

filename [string](#)

RESETTIMER()

```
public void RESETTIMER()
```

SETGRAVITY(double, double)

```
public void SETGRAVITY(double x, double y)
```

Parameters

x [double](#)

y [double](#)

SETLINEARDAMPING(double, double)

```
public void SETLINEARDAMPING(double x, double y)
```

Parameters

x [double](#)

y [double](#)

SETMATERIAL(int, string)

```
public void SETMATERIAL(int id, string material_name)
```

Parameters

id [int](#)

material_name [string](#)

SETPOSITION(int, double, double)

```
public void SETPOSITION(int id, double x, double y)
```

Parameters

id [int](#)

x [double](#)

y [double](#)

SETVELOCITY(int, double, double)

```
public void SETVELOCITY(int id, double x, double y)
```

Parameters

id [int](#)

x [double](#)

y [double](#)

TICK()

```
public void TICK()
```

UNLINK(int)

```
public void UNLINK(int id)
```

Parameters

id [int](#)

Namespace Matrix

Classes

[MATRIX](#)

2D Boulder Dash-like simulation.

Class MATRIX

Namespace: [Matrix](#)

Assembly: Matrix.dll

2D Boulder Dash-like simulation.

```
public class MATRIX : OBJECT
```

Inheritance

[object](#) ↴ ← [OBJECT](#) ← MATRIX

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↴ , [object.Equals\(object, object\)](#) ↴ , [object.GetHashCode\(\)](#) ↴ ,
[object.GetType\(\)](#) ↴ , [object.MemberwiseClone\(\)](#) ↴ , [object.ReferenceEquals\(object, object\)](#) ↴ ,
[object.ToString\(\)](#) ↴

Properties

BASEPOS

```
public (int, int) BASEPOS { init; }
```

Property Value

([int](#) ↴ , [int](#) ↴)

CELLHEIGHT

```
public int CELLHEIGHT { init; }
```

Property Value

[int](#)

CELLWIDTH

```
public int CELLWIDTH { init; }
```

Property Value

[int](#)

SIZE

```
public (int, int) SIZE { init; }
```

Property Value

([int](#), [int](#))

Methods

CALCENEMYMOVEDEST(int, int)

```
public int CALCENEMYMOVEDEST(int current_cell, int current_direction)
```

Parameters

current_cell [int](#)

current_direction [int](#)

Returns

[int](#)

CALCENEMYMOVEDIR(int, int)

```
public int CALCENEMYMOVEDIR(int current_cell, int current_direction)
```

Parameters

current_cell [int](#)

current_direction [int](#)

Returns

[int](#)

CANHEROGOTO(int)

```
public bool CANHEROGOTO(int cell_index)
```

Parameters

cell_index [int](#)

Returns

[bool](#)

GET(int)

```
public int GET(int cell_index)
```

Parameters

cell_index [int](#)

Returns

[int](#)

GETCELLOFFSET(int, int)

```
public int GETCELLOFFSET(int x, int y)
```

Parameters

x [int](#)

y [int](#)

Returns

[int](#)

GETCELLPOSX(int)

```
public int GETCELLPOSX(int cell_index)
```

Parameters

cell_index [int](#)

Returns

[int](#)

GETCELLPOSY(int)

```
public int GETCELLPOSY(int cell_index)
```

Parameters

cell_index [int](#)

Returns

[int](#)

GETCELLSNO(int)

```
public int GETCELLSNO(int cell_type)
```

Parameters

cell_type [int](#)

Returns

[int](#)

GETFIELDPOSX(int)

```
public int GETFIELDPOSX(int cell_index)
```

Parameters

cell_index [int](#)

Returns

[int](#)

GETFIELDPOSY(int)

```
public int GETFIELDPOSY(int cell_index)
```

Parameters

cell_index [int](#)

Returns

[int](#)

GETOFFSET(int, int)

```
public int GETOFFSET(int x, int y)
```

Parameters

x [int](#)

y [int](#)

Returns

[int](#)

ISGATEEMPTY()

```
public bool ISGATEEMPTY()
```

Returns

[bool](#)

ISINGATE(int)

```
public bool ISINGATE(int _)
```

Parameters

_ [int](#)

Returns

[bool](#)

MOVE(int, int)

```
public void MOVE(int _, int _2)
```

Parameters

_ [int](#)

_2 [int](#)

NEXT()

```
public int NEXT()
```

Returns

[int](#)

SET(int, int)

```
public void SET(int cell_index, int cell_type)
```

Parameters

cell_index [int](#)

cell_type [int](#)

SETGATE(int, int, int, int)

```
public void SETGATE(int _, int _2, int _3, int _4)
```

Parameters

_1 [int](#)

_2 [int](#)

_3 [int](#)

_4 [int](#)

SETROW(int, params int[])

```
public void SETROW(int row_index, params int[] cell_types)
```

Parameters

row_index [int](#)

cell_types [int](#)[]

TICK()

```
public void TICK()
```

Events

ONLATEST

```
public event SignalHandler? ONLATEST
```

Event Type

[SignalHandler](#)

ONNEXT

```
public event SignalHandler? ONNEXT
```

Event Type

[SignalHandler](#)

Namespace PIKLib

Classes

[AA|AA](#)

A set of "global" methods callable in isolation from any object using the @ syntax.

[ANIMO](#)

2D sprite animation.

[APPLICATION](#)

[ARRAY](#)

[BEHAVIOUR](#)

[BOOL](#)

Boolean value.

[BUTTON](#)

An interactable area which reacts to mouse cursor being hovered over it and clicking it.

[CANVAS_OBSERVER](#)

[CLASS](#)

[CNVLOADER](#)

[COMPLEXCONDITION](#)

[CONDITION](#)

[DATABASE](#)

[DOUBLE](#)

[EPISODE](#)

[EXPRESSION](#)

[FILTER](#)

[FONT](#)

[GROUP](#)

[IMAGE](#)

[INTEGER](#)

[KEYBOARD](#)

[MOUSE](#)

[MULTIARRAY](#)

[MUSIC](#)

[PATTERN](#)

[RAND](#)

[SCENE](#)

[SEQUENCE](#)

[SOUND](#)

[STATICFILTER](#)

[STRING](#)

[STRUCT](#)

[SYSTEM](#)

[TEXT](#)

[TIMER](#)

[VECTOR](#)

[VIRTUALGRAPHICSOBJECT](#)

Class AA\AA

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

A set of "global" methods callable in isolation from any object using the @ syntax.

```
public static class AA\AA
```

Inheritance

[object](#) ← AA\AA

Inherited Members

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

Methods

BOOL(string, bool)

Creates an object of type [BOOL](#).

```
public static void BOOL(string name, bool value)
```

Parameters

name [string](#)

The name of created object.

value [bool](#)

The initial value of created object.

BREAK()

```
public static void BREAK()
```

DOUBLE(string, double)

Creates an object of type [DOUBLE](#).

```
public static void DOUBLE(string name, double value)
```

Parameters

name [string](#)

The name of created object.

value [double](#)

The initial value of created object.

IF(string, string, string)

```
public static void IF(string condition, string code_if_true, string code_if_false)
```

Parameters

condition [string](#)

code_if_true [string](#)

code_if_false [string](#)

IF(string, string, string, string, string)

```
public static void IF(string left, string operand, string right, string
code_if_true, string code_if_false)
```

Parameters

left [string](#)

operand [string](#)

right [string](#)

code_if_true [string](#)

code_if_false [string](#)

INT(string, int)

Creates an object of type [INTEGER](#).

```
public static void INT(string name, int value)
```

Parameters

name [string](#)

The name of created object.

value [int](#)

The initial value of created object.

LOOP(string, int, int, int)

```
public static void LOOP(string behaviour, int init, int len, int step)
```

Parameters

behaviour [string](#)

init [int](#)

len [int](#)

step [int](#)

MSGBOX(string)

```
public static void MSGBOX(string message)
```

Parameters

message [string](#)

RETURN(variable)

```
public static variable RETURN(variable value)
```

Parameters

value [variable](#)

Returns

[variable](#)

STRING(string, string)

Creates an object of type [STRING](#).

```
public static string STRING(string name, string value)
```

Parameters

name [string](#)

The name of created object.

value [string](#)

The initial value of created object.

Returns

[string ↗](#)

WHILE(string, string, string, string)

```
public static void WHILE(string left, string condition, string right, string code)
```

Parameters

left [string ↗](#)

condition [string ↗](#)

right [string ↗](#)

code [string ↗](#)

Class ANIMO

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

2D sprite animation.

```
public class ANIMO : OBJECT
```

Inheritance

[object](#) ↴ ← [OBJECT](#) ← ANIMO

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↴ , [object.Equals\(object, object\)](#) ↴ , [object.GetHashCode\(\)](#) ↴ ,
[object.GetType\(\)](#) ↴ , [object.MemberwiseClone\(\)](#) ↴ , [object.ReferenceEquals\(object, object\)](#) ↴ ,
[object.ToString\(\)](#) ↴

Properties

FILENAME

A path to a file where the animation data is stored.

```
public string FILENAME { init; }
```

Property Value

[string](#) ↴

Remarks

The file must be in the ANN format.

See Also

[LOAD\(string\)](#)

FPS

Target animation speed (in frames per second).

```
public int? FPS { init; }
```

Property Value

[int](#)?

Default value: NULL.

Remarks

The property overrides settings from a file.

See Also

[SETFPS\(int\)](#)

MONITORCOLLISIONALPHA

Should the pixel-perfect collision algorithm be used instead of the default AABB one?

```
public bool MONITORCOLLISIONALPHA { init; }
```

Property Value

[bool](#)?

Default value: FALSE.

Remarks

The property is ignored if [MONITORCOLLISION](#) is set to FALSE.

See Also

[MONITORCOLLISION](#), [MONITORCOLLISION\(bool\)](#), [REMOVEMONITORCOLLISION\(\)](#)

MONITORCOLLISION

Should 2D collisions be detected for the object?

```
public bool MONITORCOLLISION { init; }
```

Property Value

[bool](#)

Default value: FALSE.

Remarks

Setting the property to TRUE means [ONCOLLISION](#) callback is called.

By default, AABB collision algorithm is used.

See Also

[MONITORCOLLISIONALPHA](#), [MONITORCOLLISION\(bool\)](#), [REMOVEMONITORCOLLISION\(\)](#)

PRELOAD

Should the file specified in [FILENAME](#) be loaded instantly?

```
public bool PRELOAD { init; }
```

Property Value

[bool](#)

PRIORITY

The position of the object on the Z axis (directed out of the screen). Objects with greater priority are displayed over those with smaller priority.

```
public int PRIORITY { init; }
```

Property Value

[int](#)

Remarks

For the same priority value, objects declared later are displayed over those declared before.

See Also

[GETPRIORITY\(\)](#), [SETPRIORITY\(int\)](#)

RELEASE

Should the file handle be released after loading the file specified in [FILENAME](#)?

```
public bool RELEASE { init; }
```

Property Value

[bool](#)

TOCANVAS

```
public bool TOCANVAS { init; }
```

Property Value

[bool](#)

Default value: FALSE.

Remarks

Without setting the property to TRUE, the object remains invisible.

VISIBLE

Should the object be visible by default?

```
public bool VISIBLE { init; }
```

Property Value

[bool](#)

Default value: FALSE.

See Also

[TOCANVAS](#), [HIDE\(\)](#), [ISVISIBLE\(\)](#), [SHOW\(\)](#).

Methods

GETCENTERX()

Retrieves the horizontal position of the center of the object (in pixels).

```
public int GETCENTERX()
```

Returns

[int](#)

The horizontal position of the object's center (in pixels).

GETCENTERY()

Retrieves the vertical position of the center of the object (in pixels).

```
public int GETCENTERY()
```

Returns

[int](#)

The vertical position of the object's center (in pixels).

GETCFRAMEINEVENT()

Retrieves the 0-based index of the current frame within its parent public event (a sequence of frames).

```
public int GETCFRAMEINEVENT()
```

Returns

[int](#)

The index of the current frame within the current event.

See Also

[GETFRAME\(\)](#)

GETCURRFRAMEPOSX()

Retrieves the horizontal position of the current animation frame (in pixels).

```
public int GETCURRFRAMEPOSX()
```

Returns

[int](#)

The horizontal position of the current animation frame (in pixels).

GETCURRFRAMEPOSY()

Retrieves the vertical position of the current animation frame (in pixels).

```
public int GETCURRFRAMEPOSY()
```

Returns

[int](#)

The vertical position of the current animation frame (in pixels).

GETENDX()

Retrieves the horizontal position of the right edge of the object (in pixels).

```
public int GETENDX()
```

Returns

[int](#)

The horizontal position of the object's right edge (in pixels).

GETENDY()

Retrieves the vertical position of the bottom edge of the object (in pixels).

```
public int GETENDY()
```

Returns

[int](#)

The vertical position of the object's bottom edge (in pixels).

GETEVENTNAME()

Retrieves the name of the current public event (a sequence of frames).

```
public string GETEVENTNAME()
```

Returns

[string](#)

The name of the current event.

GETFRAME()

Retrieves the 0-based index of the compound image pointed to by the current animation frame.

```
public int GETFRAME()
```

Returns

[int](#)

The index of currently displayed animation image (not the index of a frame).

See Also

[SETFRAME\(int\)](#), [GETCFRAMEINEVENT\(\)](#)

GETFRAMENAME()

Retrieves the name of the current frame.

```
public string GETFRAMENAME()
```

Returns

[string](#)

The name of the current frame.

See Also

[SETFRAMENAME\(string\)](#)

GETHEIGHT()

Retrieves the height of the current animation frame (in pixels).

```
public int GETHEIGHT()
```

Returns

[int](#)

The height of the current animation frame (in pixels).

GETMAXWIDTH()

Checks the height of all compound images and returns the maximum (in pixels).

```
public int GETMAXWIDTH()
```

Returns

[int](#)

The maximal possible height of the displayed image (in pixels).

GETNOE()

Retrieves the total number of animation events (sequences of frames).

```
public int GETNOE()
```

Returns

[int](#)

The total number of animation events.

GETNOF()

Retrieves the total number of animation frames within the current public event (a sequence of frames).

```
public int GETNOF()
```

Returns

[int](#)

The total number of animation frames within the current event.

GETNOFINEVENT(string)

Retrieves the total number of animation frames within the public event (a sequence of frames) identified by the given name.

```
public int GETNOFINEVENT(string event_name)
```

Parameters

`event_name` [string](#)

The name of the public event to check.

Returns

[int](#)

The total number of animation frames within the specified event.

GETOPACITY()

Retrieves the general opacity of the animation.

```
public int GETOPACITY()
```

Returns

[int](#)

The general opacity of the animation.

See Also

[SETOPACITY\(int\)](#)

GETPOSITIONX()

Retrieves the horizontal position of the object (in pixels).

```
public int GETPOSITIONX()
```

Returns

[int](#)

The horizontal position of the object (in pixels).

See Also

[GETPOSITIONY\(\)](#), [SETPOSITION\(int, int\)](#), [MOVE\(int, int\)](#)

GETPOSITIONY()

Retrieves the base vertical position of the object (in pixels).

```
public int GETPOSITIONY()
```

Returns

[int](#)

The base vertical position of the object (in pixels).

See Also

[GETPOSITIONX\(\)](#), [SETPOSITION\(int, int\)](#), [MOVE\(int, int\)](#)

GETPRIORITY()

Retrieves the priority of the object.

```
public int GETPRIORITY()
```

Returns

[int](#)

The priority of the object.

See Also

[PRIORITY](#), [SETPRIORITY\(int\)](#)

GETWIDTH()

Retrieves the width of the current animation frame (in pixels).

```
public int GETWIDTH()
```

Returns

[int](#)

The width of the current animation frame (in pixels).

HIDE()

Makes the object invisible.

```
public void HIDE()
```

See Also

[VISIBLE](#), [ISVISIBLE\(\)](#), [SHOW\(\)](#)

INVALIDATE()

```
public void INVALIDATE()
```

ISAT(int, int, bool)

```
public bool ISAT(int x, int y, bool _)
```

Parameters

x [int](#)

y [int](#)

_ [bool](#)

Returns

[bool](#)

ISNEAR(string, string)

Checks if the object is near the [other](#) one.

```
public bool ISNEAR(string other, string iou_threshold)
```

Parameters

[other](#) [string](#)

Another graphics object for which nearness with the current object is checked.

[iou_threshold](#) [string](#)

Minimum IoU value to treat two objects as being near each other.

Returns

[bool](#)

A boolean value indicating if objects are near each other.

ISPLAYING()

Checks if any animation public event (a sequence of frames) is currently being played.

```
public bool ISPLAYING()
```

Returns

[bool](#)

Boolean value indicating if any animation public event is currently being played.

See Also

[PLAY\(string\)](#), [PLAY\(int\)](#), [STOP\(bool\)](#), [PAUSE\(\)](#), [RESUME\(\)](#)

ISVISIBLE()

Checks if the object is visible.

```
public bool ISVISIBLE()
```

Returns

[bool](#)

A boolean value indicating if the object is visible.

See Also

[VISIBLE](#), [HIDE\(\)](#), [SHOW\(\)](#)

LOAD(string)

Loads animation data from the file located at the given path.

```
public void LOAD(string filename)
```

Parameters

[filename](#) [string](#)

The path where the file is located.

MERGEALPHA()

```
public void MERGEALPHA()
```

MONITORCOLLISION(bool)

Enables the 2D collision detection for the object.

```
public void MONITORCOLLISION(bool pixel_perfect)
```

Parameters

`pixel_perfect bool`

Should the pixel-perfect collision algorithm be used instead of the default AABB one?

See Also

[MONITORCOLLISION\(\)](#), [MONITORCOLLISIONALPHA](#), [REMOVEMONITORCOLLISION\(\)](#)

MOVE(int, int)

Moves the animation object by the given offset (in pixels).

```
public void MOVE(int x_offset, int y_offset)
```

Parameters

`x_offset int`

A horizontal offset to be added to the current base object position.

`y_offset int`

A vertical offset to be added to the current base object position.

See Also

[GETPOSITIONX\(\)](#), [GETPOSITIONY\(\)](#), [SETPOSITION\(int, int\)](#)

NEXT()

```
public void NEXT()
```

NEXTFRAME()

```
public void NEXTFRAME()
```

NPLAY()

```
public void NPLAY()
```

PAUSE()

Pauses the current animation public event (a sequence of frames).

```
public void PAUSE()
```

See Also

[RESUME\(\)](#), [ISPLAYING\(\)](#), [PLAY\(string\)](#), [PLAY\(int\)](#), [STOP\(bool\)](#)

PLAY(int)

Plays animation public event identified by index `event_index`.

```
public void PLAY(int event_index)
```

Parameters

`event_index` [`int`](#)

The 0-based index of the animation public event to be played.

Remarks

Makes the object visible.

Animation can loop depending on its definition (see ANN format specification).

See Also

[PLAY\(string\)](#), [ISPLAYING\(\)](#), [STOP\(bool\)](#), [PAUSE\(\)](#), [RESUME\(\)](#)

PLAY(string)

Plays animation public event identified by name `event_name`.

```
public void PLAY(string event_name)
```

Parameters

`event_name` [string](#)

The name of the animation public event to be played.

Remarks

Makes the object visible.

Animation can loop depending on its definition (see ANN format specification).

See Also

[PLAY\(int\)](#), [ISPLAYING\(\)](#), [STOP\(bool\)](#), [PAUSE\(\)](#), [RESUME\(\)](#).

PREVFRAME()

```
public void PREVFRAME()
```

REMOVEMONITORCOLLISION()

Disables the 2D collision detection for the object.

```
public void REMOVEMONITORCOLLISION()
```

See Also

[MONITORCOLLISION](#), [MONITORCOLLISIONALPHA](#), [MONITORCOLLISION\(bool\)](#).

RESUME()

Unpauses the current animation public event (a sequence of frames).

```
public void RESUME()
```

See Also

[PAUSE\(\)](#), [ISPLAYING\(\)](#), [PLAY\(string\)](#), [PLAY\(int\)](#), [STOP\(bool\)](#)

SETANCHOR(anchor)

```
public void SETANCHOR(anchor anchor)
```

Parameters

anchor [anchor](#)

SETASBUTTON(bool, bool)

Enables or disables the object being interactable.

```
public void SETASBUTTON(bool as_button, bool with_cursor_pointer)
```

Parameters

as_button [bool](#)

Should interactability be enabled?

with_cursor_pointer [bool](#)

Should the mouse cursor icon be changed to pointer when it hovers over the interactable area? The argument is ignored if [as_button](#) is FALSE.

Remarks

Interactability means reacting to mouse cursor being hovered and mouse button being clicked over the displayed object. In reaction to these events, the following signals are fires: [ONFOCUSON](#), [ONFOCUSOFF](#), [ONCLICK](#), [ONRELEASE](#).

The interactable area is the AABB representing the frame at the time of calling the method.

SETBACKWARD()

```
public void SETBACKWARD()
```

SETCLIPPING()

```
public void SETCLIPPING()
```

SETFORWARD()

```
public void SETFORWARD()
```

SETFPS(int)

Sets the animation speed (in frames per second).

```
public void SETFPS(int fps)
```

Parameters

fps [int ↗](#)

Remarks

The property overrides settings from a file.

See Also

[FPS](#)

SETFRAME(int)

Changes the currently displayed frame to the image identified by the given index.

```
public void SETFRAME(int image_index)
```

Parameters

`image_index int`

Remarks

If any animation public event (a sequence of frames) is currently played, the image set by this method gets overwritten by the next update.

See Also

[GETFRAME\(\)](#), [SETFRAME\(string, int\)](#)

SETFRAME(string, int)

Changes the currently displayed frame to the one identified by the given index within the public event of name `event_name`.

```
public void SETFRAME(string event_name, int frame_index)
```

Parameters

`event_name string`

`frame_index int`

See Also

[GETFRAME\(\)](#), [SETFRAME\(int\)](#)

SETFRAMENAME(string)

Changes the name of the current frame.

```
public void SETFRAMENAME(string frame_name)
```

Parameters

`frame_name string`

The name to be set.

See Also

[GETFRAMENAME\(\)](#)

SETOPACITY(int)

Sets the general opacity of the animation.

```
public void SETOPACITY(int opacity)
```

Parameters

opacity [int](#)

The opacity to be set.

See Also

[GETOPACITY\(\)](#)

SETPOSITION(int, int)

Sets the base position of the animation.

```
public void SETPOSITION(int x, int y)
```

Parameters

x [int](#)

The horizontal position to be set (in pixels).

y [int](#)

The vertical position to be set (in pixels).

See Also

[GETPOSITIONX\(\)](#), [GETPOSITIONY\(\)](#), [MOVE\(int, int\)](#)

SETPRIORITY(int)

Sets the priority of the object.

```
public void SETPRIORITY(int priority)
```

Parameters

priority [int](#)

The priority to be set.

See Also

[PRIORITY](#), [GETPRIORITY\(\)](#).

SHOW()

Makes the object visible.

```
public void SHOW()
```

See Also

[VISIBLE](#), [HIDE\(\)](#), [ISVISIBLE\(\)](#).

STOP(bool)

Stops the animation public event (a sequence of frames) being currently played.

```
public void STOP(bool emit_on_finished = true)
```

Parameters

emit_on_finished [bool](#)

Should the [ONFINISHED](#) signal be emitted?

Events

ONCLICK

A handler for the signal emitted when the object is in interactive mode and a mouse button is pressed over it.

```
public event SignalHandler? ONCLICK
```

Event Type

[SignalHandler](#)

See Also

[SETASBUTTON\(bool, bool\)](#)

ONCOLLISION

A handler for the signal emitted when the object is in collision with another object.

```
public event SignalHandler<other_name>? ONCOLLISION
```

Event Type

[SignalHandler<other_name>](#)

Remarks

Can be specialized using the [other_name](#) parameter which is the name of the object with which the current object collides.

See Also

[MONITORCOLLISION\(\)](#), [MONITORCOLLISIONALPHA](#), [MONITORCOLLISION\(bool\)](#),
[REMOVEMONITORCOLLISION\(\)](#).

ONFINISHED

A handler for the signal emitted when an animation public event has finished playing.

```
public event SignalHandler<event_name>? ONFINISHED
```

Event Type

[SignalHandler<event_name>](#)

Remarks

Can be specialized using the [event_name](#) parameter which is the name of the finished event.

See Also

[STOP\(bool\)](#), [PLAY\(string\)](#)

ONFOCUSOFF

A handler for the signal emitted when the object is in interactive mode and the mouse cursors is moved outside its area.

```
public event SignalHandler? ONFOCUSOFF
```

Event Type

[SignalHandler](#)

See Also

[SETASBUTTON\(bool, bool\)](#)

ONFOCUSON

A handler for the signal emitted when the object is in interactive mode and the mouse cursors is moved onto it.

```
public event SignalHandler? ONFOCUSON
```

Event Type

[SignalHandler](#)

See Also

[SETASBUTTON\(bool, bool\)](#)

ONFRAMECHANGED

A handler for the signal emitted when the displayed frame changes as a result of an animation public event being played.

```
public event SignalHandler? ONFRAMECHANGED
```

Event Type

[SignalHandler](#)

Remarks

Can be specialized using the [event_name](#) parameter which is the name of the public event currently being played.

See Also

[PLAY\(string\)](#)

ONINIT

A handler for the signal emitted when the object is initialized.

```
public event SignalHandler? ONINIT
```

Event Type

[SignalHandler](#)

ONRELEASE

A handler for the signal emitted when the object is in interactive mode and a mouse button is released over it.

```
public event SignalHandler? ONRELEASE
```

Event Type

[SignalHandler](#)

See Also

[SETASBUTTON\(bool, bool\)](#)

ONSIGNAL

A handler for the signal emitted when a custom message is sent.

```
public event SignalHandler<signal_name>? ONSIGNAL
```

Event Type

[SignalHandler<signal_name>](#)

Remarks

Can be specialized using the [event_name](#) parameter which is the name of the public event currently being played.

ONSTARTED

A handler for the signal emitted when an animation public event has started playing.

```
public event SignalHandler<event_name>? ONSTARTED
```

Event Type

[SignalHandler<event_name>](#)

Remarks

Can be specialized using the [event_name](#) parameter which is the name of the started event.

See Also

[PLAY\(string\)](#)

Class APPLICATION

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class APPLICATION : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ APPLICATION

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

AUTHOR

```
public string AUTHOR { init; }
```

Property Value

[string](#)

BLOOMOO_VERSION

```
public string BLOOMOO_VERSION { init; }
```

Property Value

[string](#)

CREATIONTIME

```
public string CREATIONTIME { init; }
```

Property Value

[string ↗](#)

EPISODES

```
public string EPISODES { init; }
```

Property Value

[string ↗](#)

LASTMODIFYTIME

```
public string LASTMODIFYTIME { init; }
```

Property Value

[string ↗](#)

PATH

```
public string PATH { init; }
```

Property Value

[string ↗](#)

STARTWITH

```
public string STARTWITH { init; }
```

Property Value

[string](#)

VERSION

```
public string VERSION { init; }
```

Property Value

[string](#)

Methods

EXIT()

```
public void EXIT()
```

GETLANGUAGE()

```
public string GETLANGUAGE()
```

Returns

[string](#)

RUN(string, string, params variable[])

```
public variable? RUN(string object_name, string method_name, params  
variable[] arguments)
```

Parameters

`object_name` [string](#)

`method_name` [string](#)

`arguments` [variable](#)[]

Returns

[variable](#)

RUNENV(string, string)

```
public variable? RUNENV(string scene_name, string beh_name)
```

Parameters

`scene_name` [string](#)

`beh_name` [string](#)

Returns

[variable](#)

SETLANGUAGE(string)

```
public void SETLANGUAGE(string lang_id)
```

Parameters

`lang_id` [string](#)

Class ARRAY

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class ARRAY : OBJECT
```

Inheritance

[object](#) ↗ ← [OBJECT](#) ← ARRAY

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↗ , [object.Equals\(object, object\)](#) ↗ , [object.GetHashCode\(\)](#) ↗ ,
[object.GetType\(\)](#) ↗ , [object.MemberwiseClone\(\)](#) ↗ , [object.ReferenceEquals\(object, object\)](#) ↗ ,
[object.ToString\(\)](#) ↗

Methods

ADD()

```
public void ADD()
```

ADDAT(int, variable)

```
public void ADDAT(int index, variable summand)
```

Parameters

index [int](#) ↗

summand [variable](#)

CHANGEAT(int, variable)

```
public void CHANGEAT(int index, variable value)
```

Parameters

index [int](#)

value [variable](#)

CLAMPAT(int, variable, variable)

```
public void CLAMPAT(int index, variable min, variable max)
```

Parameters

index [int](#)

min [variable](#)

max [variable](#)

CONTAINS(variable)

```
public void CONTAINS(variable value)
```

Parameters

value [variable](#)

COPYTO()

```
public void COPYTO()
```

FIND()

```
public void FIND()
```

GET(int)

```
public void GET(int index)
```

Parameters

index [int](#)

GETSIZE()

```
public void GETSIZE()
```

GETSUMVALUE()

```
public void GETSUMVALUE()
```

INSERTAT(int, variable)

```
public void INSERTAT(int index, variable value)
```

Parameters

index [int](#)

value [variable](#)

LOAD()

```
public void LOAD()
```

LOADINI()

```
public void LOADINI()
```

MODAT()

```
public void MODAT()
```

MULAT()

```
public void MULAT()
```

REMOVE()

```
public void REMOVE()
```

REMOVEALL()

```
public void REMOVEALL()
```

REMOVEAT()

```
public void REMOVEAT()
```

REVERSEFIND()

```
public void REVERSEFIND()
```

SAVE()

```
public void SAVE()
```

SAVEINI()

```
public void SAVEINI()
```

SUB()

```
public void SUB()
```

SUBAT()

```
public void SUBAT()
```

SUM()

```
public void SUM()
```

Class BEHAVIOUR

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class BEHAVIOUR : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ BEHAVIOUR

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

CODE

```
public string CODE { init; }
```

Property Value

[string](#)

CONDITION

```
public string CONDITION { init; }
```

Property Value

[string](#)

Methods

RUN(params variable[])

```
public variable? RUN(params variable[] arguments)
```

Parameters

arguments [variable\[\]](#)

Returns

[variable](#)

RUNC(params variable[])

```
public variable? RUNC(params variable[] arguments)
```

Parameters

arguments [variable\[\]](#)

Returns

[variable](#)

RUNLOOPED(int, int, int)

```
public void RUNLOOPED(int start, int range_size, int step = 1)
```

Parameters

start [int](#)

range_size [int](#)

step [int](#)

Class BOOL

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

Boolean value.

```
public class BOOL : OBJECT
```

Inheritance

[object](#) ↴ ← [OBJECT](#) ← BOOL

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↴ , [object.Equals\(object, object\)](#) ↴ , [object.GetHashCode\(\)](#) ↴ ,
[object.GetType\(\)](#) ↴ , [object.MemberwiseClone\(\)](#) ↴ , [object.ReferenceEquals\(object, object\)](#) ↴ ,
[object.ToString\(\)](#) ↴

Properties

TOINI

```
public bool TOINI { init; }
```

Property Value

[bool](#) ↴

VALUE

```
public bool VALUE { init; }
```

Property Value

[bool](#)

Methods

SET(bool)

Sets the value of the object to `value`.

```
public void SET(bool value)
```

Parameters

`value` [bool](#)

New value for the object.

SWITCH(bool, bool)

Switches the value of the object between `TRUE` and `FALSE`.

```
public void SWITCH(bool _unused1, bool _unused2)
```

Parameters

`_unused1` [bool](#)

Unused.

`_unused2` [bool](#)

Unused.

Events

ONBRUTALCHANGED

```
public event SignalHandler<stringified_value>? ONBRUTALCHANGED
```

Event Type

[SignalHandler<stringified_value>](#)

ONCHANGED

`public event SignalHandler<stringified_value>? ONCHANGED`

Event Type

[SignalHandler<stringified_value>](#)

Class BUTTON

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

An interactable area which reacts to mouse cursor being hovered over it and clicking it.

```
public class BUTTON : OBJECT
```

Inheritance

[object](#) ↴ ← [OBJECT](#) ← BUTTON

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↴ , [object.Equals\(object, object\)](#) ↴ , [object.GetHashCode\(\)](#) ↴ ,
[object.GetType\(\)](#) ↴ , [object.MemberwiseClone\(\)](#) ↴ , [object.ReferenceEquals\(object, object\)](#) ↴ ,
[object.ToString\(\)](#) ↴

Properties

DRAGGABLE

A value specifying if the object should support dragging.

```
public bool DRAGGABLE { init; }
```

Property Value

[bool](#) ↴

ENABLE

A value specifying if the object should be activated by default.

```
public bool ENABLE { init; }
```

Property Value

[bool](#) ↗

Remarks

A disabled button hides associated objects supplied using the [GFXSTANDARD](#), [GFXONMOVE](#) and [GFXONCLICK](#) properties.

GFXONCLICK

The name of an [ANIMO](#) or [IMAGE](#) object to be shown when the mouse button is being pressed over the button.

```
public string? GFXONCLICK { init; }
```

Property Value

[string](#) ↗

Remarks

This property is overridden by the [RECT](#) property.

GFXONMOVE

The name of an [ANIMO](#) or [IMAGE](#) object to be shown when the mouse cursor is being hovered over the button.

```
public string? GFXONMOVE { init; }
```

Property Value

[string](#) ↗

Remarks

This property is overridden by the [RECT](#) property.

GFXSTANDARD

The name of an [ANIMO](#) or [IMAGE](#) object to be shown when the button is neither pressed nor hovered upon.

```
public string GFXSTANDARD { init; }
```

Property Value

[string](#)

Remarks

This property is overridden by the [RECT](#) property.

RECT

A literal rect described by four coordinates or a reference being the name of a graphical object to base the rect on, capturing its current state.

```
public rect RECT { init; }
```

Property Value

[rect](#)

Remarks

This property overrides the [GFXSTANDARD](#), [GFXONMOVE](#) and [GFXONCLICK](#) properties.

If a reference is used, the rect only reflects the state of the referenced object at the time of the method call. For example changing the referenced animation frame does not result in the interactive area of the button being resized/moved.

SNDONMOVE

The name of a [SOUND](#) object to be played when mouse cursor hovers over the button.

```
public string? SNDONMOVE { init; }
```

Property Value

[string](#) ↗

Methods

DISABLE()

```
public void DISABLE()
```

DISABLEBUTVISIBLE()

```
public void DISABLEBUTVISIBLE()
```

ENABLE()

```
public void ENABLE()
```

GETSTD()

```
public string GETSTD()
```

Returns

[string](#) ↗

SETONCLICK(string)

```
public void SETONCLICK(string object_name)
```

Parameters

object_name [string](#)

SETONMOVE(string)

```
public void SETONMOVE(string object_name)
```

Parameters

object_name [string](#)

SETPRIORITY(int)

```
public void SETPRIORITY(int priority)
```

Parameters

priority [int](#)

SETRECT(rect)

Sets the interactive area of the button to the given [rect](#).

```
public void SETRECT(rect rect)
```

Parameters

rect [rect](#)

A literal rect described by four coordinates or a reference being the name of a graphical object to base the rect on, capturing its current state.

Remarks

Calling this method does not make the button visible or enabled.

Setting a rect makes the [GFXSTANDARD](#), [GFXONMOVE](#) and [GFXONCLICK](#) properties as well as any further calls to the [SETSTD\(string\)](#), [SETONMOVE\(string\)](#) and [SETONCLICK\(string\)](#) methods to be ignored.

If a reference is used, the rect only reflects the state of the referenced object at the time of the method call. For example changing the referenced animation frame does not result in the interactive area of the button being resized/moved.

SETSTD(string)

```
public void SETSTD(string object_name)
```

Parameters

object_name [string](#)

Events

ONACTION

```
public event SignalHandler? ONACTION
```

Event Type

[SignalHandler](#)

ONCLICKED

```
public event SignalHandler? ONCLICKED
```

Event Type

[SignalHandler](#)

ONDRAZZING

```
public event SignalHandler? ONDRAGGING
```

Event Type

[SignalHandler](#)

ONENDDRAZZING

```
public event SignalHandler? ONENDDRAZZING
```

Event Type

[SignalHandler](#)

ONFOCUSOFF

```
public event SignalHandler? ONFOCUSOFF
```

Event Type

[SignalHandler](#)

ONFOCUSON

```
public event SignalHandler? ONFOCUSON
```

Event Type

[SignalHandler](#)

ONINIT

```
public event SignalHandler? ONINIT
```

Event Type

[SignalHandler](#)

ONRELEASED

```
public event SignalHandler? ONRELEASED
```

Event Type

[SignalHandler](#)

ONSTARTDRAGGING

```
public event SignalHandler? ONSTARTDRAGGING
```

Event Type

[SignalHandler](#)

Class CANVAS_OBSERVER

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class CANVAS_OBSERVER : OBJECT
```

Inheritance

[object](#) ← [OBJECT](#) ← CANVAS_OBSERVER

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Methods

ADD()

```
public void ADD()
```

ENABLENOTIFY()

```
public void ENABLENOTIFY()
```

GETGRAPHICSAT(int, int)

```
public string? GETGRAPHICSAT(int x_position, int y_position)
```

Parameters

x_position [int](#)

y_position [int](#)

Returns

[string](#)

GETGRAPHICSAT(int, int, bool, int, int, bool)

```
public string? GETGRAPHICSAT(int x_position, int y_position, bool _unknown, int min_priority, int max_priority, bool pixel_perfect)
```

Parameters

x_position [int](#)

y_position [int](#)

_unknown [bool](#)

min_priority [int](#)

max_priority [int](#)

pixel_perfect [bool](#)

Returns

[string](#)

MOVEBKG(int, int)

```
public void MOVEBKG(int x_offset, int y_offset)
```

Parameters

x_offset [int](#)

y_offset int

PASTE()

```
public void PASTE()
```

REDRAW()

```
public void REDRAW()
```

REFRESH()

```
public void REFRESH()
```

REMOVE()

```
public void REMOVE()
```

SAVE(string)

```
public void SAVE(string filename)
```

Parameters

filename string

SETBACKGROUND(string)

```
public void SETBACKGROUND(string object_name_or_filename)
```

Parameters

object_name_or_filename [string](#)

SETBKGPOS(int, int)

```
public void SETBKGPOS(int x, int y)
```

Parameters

x [int](#)

y [int](#)

Events

ONWINDOWFOCUSOFF

```
public event SignalHandler? ONWINDOWFOCUSOFF
```

Event Type

[SignalHandler](#)

ONWINDOWFOCUSON

```
public event SignalHandler? ONWINDOWFOCUSON
```

Event Type

[SignalHandler](#)

Class CLASS

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class CLASS : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ CLASS

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

BASE

```
public string BASE { init; }
```

Property Value

[string](#)

DEF

```
public string DEF { init; }
```

Property Value

[string](#)

Methods

NEW(string, params variable[])

```
public void NEW(string object_name, params variable[] arguments)
```

Parameters

object_name [string](#)

arguments [variable](#)[]

Class CNVLOADER

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class CNVLOADER : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ CNVLOADER

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Methods

LOAD()

```
public void LOAD()
```

RELEASE()

```
public void RELEASE()
```

Class COMPLEXCONDITION

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class COMPLEXCONDITION : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ COMPLEXCONDITION

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

CONDITION1

```
public string CONDITION1 { init; }
```

Property Value

[string](#)

CONDITION2

```
public string CONDITION2 { init; }
```

Property Value

[string](#)

OPERATOR

```
public complex_operator OPERATOR { init; }
```

Property Value

[complex_operator](#)

Methods

BREAK(bool)

```
public void BREAK(bool _)
```

Parameters

_ [bool](#)

CHECK(bool)

```
public bool CHECK(bool _)
```

Parameters

_ [bool](#)

Returns

[bool](#)

ONE_BREAK(bool)

```
public void ONE_BREAK(bool _)
```

Parameters

— [bool](#) ↗

Events

ONRUNTIMEFAILED

```
public event SignalHandler? ONRUNTIMEFAILED
```

Event Type

[SignalHandler](#)

ONRUNTIMESUCCESS

```
public event SignalHandler? ONRUNTIMESUCCESS
```

Event Type

[SignalHandler](#)

Class CONDITION

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class CONDITION : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ CONDITION

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

OPERAND1

```
public string OPERAND1 { init; }
```

Property Value

[string](#)

OPERAND2

```
public string OPERAND2 { init; }
```

Property Value

[string](#)

OPERATOR

```
public condition_operator OPERATOR { init; }
```

Property Value

[condition_operator](#)

Methods

BREAK(bool)

```
public void BREAK(bool _)
```

Parameters

_ [bool](#)

CHECK(bool)

```
public bool CHECK(bool _)
```

Parameters

_ [bool](#)

Returns

[bool](#)

ONE_BREAK(bool)

```
public void ONE_BREAK(bool _)
```

Parameters

— [bool](#) ↗

Events

ONRUNTIMEFAILED

```
public event SignalHandler? ONRUNTIMEFAILED
```

Event Type

[SignalHandler](#)

ONRUNTIMESUCCESS

```
public event SignalHandler? ONRUNTIMESUCCESS
```

Event Type

[SignalHandler](#)

Class DATABASE

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class DATABASE : OBJECT
```

Inheritance

[object](#) ↗ ← [OBJECT](#) ← DATABASE

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↗ , [object.Equals\(object, object\)](#) ↗ , [object.GetHashCode\(\)](#) ↗ ,
[object.GetType\(\)](#) ↗ , [object.MemberwiseClone\(\)](#) ↗ , [object.ReferenceEquals\(object, object\)](#) ↗ ,
[object.ToString\(\)](#) ↗

Properties

MODEL

```
public string MODEL { init; }
```

Property Value

[string](#) ↗

Methods

ADD(string)

```
public void ADD(string object_name)
```

Parameters

object_name [string](#)

FIND(string, variable, int)

```
public int FIND(string column_name, variable value, int start_row_index)
```

Returns

column_name [string](#)

value [variable](#)

start_row_index [int](#)

Returns

[int](#)

GETCURSORPOS()

```
public int GETCURSORPOS()
```

Returns

[int](#)

GETROWSNO()

```
public int GETROWSNO()
```

Returns

[int](#)

LOAD(string)

```
public void LOAD(string filename)
```

Parameters

filename [string](#)

NEXT()

```
public void NEXT()
```

REMOVEALL()

```
public void REMOVEALL()
```

SAVE(string)

```
public void SAVE(string filename)
```

Parameters

filename [string](#)

SELECT(int)

```
public void SELECT(int row_index)
```

Parameters

row_index [int](#)

Class DOUBLE

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class DOUBLE : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ DOUBLE

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

TOINI

```
public bool TOINI { init; }
```

Property Value

[bool](#)

VALUE

```
public double VALUE { init; }
```

Property Value

[double](#)

Methods

ADD(double)

```
public double ADD(double summand)
```

Parameters

summand [double](#)

Returns

[double](#)

ARCTAN(double)

```
public double ARCTAN(double degrees)
```

Parameters

degrees [double](#)

Returns

[double](#)

ARCTANEX(double, double, int)

```
public double ARCTANEX(double y, double x, int summand = 0)
```

Parameters

y [double](#)

x [double](#)

summand [int](#)

Returns

[double](#)

CLAMP(double, double)

```
public double CLAMP(double min, double max)
```

Parameters

min [double](#)

max [double](#)

Returns

[double](#)

COSINUS(double)

```
public double COSINUS(double degrees)
```

Parameters

degrees [double](#)

Returns

[double](#)

DIV(double)

```
public void DIV(double divisor)
```

Parameters

`divisor double`

LENGTH(double, double)

```
public double LENGTH(double horizontal_distance, double vertical_distance)
```

Parameters

`horizontal_distance double`

`vertical_distance double`

Returns

`double`

MAXA(params double[])

```
public double MAXA(params double[] values)
```

Parameters

`values double[]`

Returns

`double`

MINA(params double[])

```
public double MINA(params double[] values)
```

Parameters

values double[]

Returns

double[]

MUL(double)

```
public void MUL(double multiplier)
```

Parameters

multiplier double[]

SET(double)

```
public void SET(double value)
```

Parameters

value double[]

SINUS(double)

```
public double SINUS(double degrees)
```

Parameters

degrees double[]

Returns

double[]

SQRT()

```
public double SQRT()
```

Returns

double

SUB(double)

```
public double SUB(double subtrahend)
```

Parameters

subtrahend double

Returns

double

Class EPISODE

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class EPISODE : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ EPISODE

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

AUTHOR

```
public string AUTHOR { init; }
```

Property Value

[string](#)

CREATIONTIME

```
public string CREATIONTIME { init; }
```

Property Value

[string](#)

LASTMODIFYTIME

```
public string LASTMODIFYTIME { init; }
```

Property Value

[string](#)

PATH

```
public string PATH { init; }
```

Property Value

[string](#)

SCENES

```
public string[] SCENES { init; }
```

Property Value

[string](#)[]

STARTWITH

```
public string STARTWITH { init; }
```

Property Value

[string](#)

VERSION

```
public string VERSION { init; }
```

Property Value

[string](#)

Methods

BACK()

```
public void BACK()
```

GETCURRENTSCENE()

```
public string GETCURRENTSCENE()
```

Returns

[string](#)

GETLATESTSCENE()

```
public string GETLATESTSCENE()
```

Returns

[string](#)

GOTO(string)

```
public void GOTO(string scene_name)
```

Parameters

scene_name [string](#) ↗

Class EXPRESSION

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class EXPRESSION : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ EXPRESSION

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

OPERAND1

```
public string OPERAND1 { init; }
```

Property Value

[string](#)

OPERAND2

```
public string OPERAND2 { init; }
```

Property Value

[string](#)

OPERATOR

```
public expression_operator OPERATOR { init; }
```

Property Value

[expression_operator](#)

Class FILTER

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class FILTER : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ FILTER

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

ACTION

```
public string ACTION { init; }
```

Property Value

[string](#)

Class FONT

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class FONT : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ FONT

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

DEF_family_style_size

```
public string DEF_family_style_size { init; }
```

Property Value

[string](#)

Class GROUP

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class GROUP : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ GROUP

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Methods

ADD(string)

```
public void ADD(string object_name)
```

Parameters

object_name [string](#)

ADDCLONES()

```
public void ADDCLONES()
```

GETSIZE()

```
public void GETSIZE()
```

NEXT()

```
public void NEXT()
```

PREV()

```
public void PREV()
```

REMOVE(string)

```
public void REMOVE(string object_name)
```

Parameters

object_name string

REMOVEALL()

```
public void REMOVEALL()
```

RESETMARKER()

```
public void RESETMARKER()
```

SETMARKERPOS(int)

```
public void SETMARKERPOS(int index)
```

Parameters

index [int](#)

Events

ONINIT

```
public event SignalHandler? ONINIT
```

Event Type

[SignalHandler](#)

Class IMAGE

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class IMAGE : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ IMAGE

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

FILENAME

```
public string FILENAME { init; }
```

Property Value

[string](#)

MONITORCOLLISION

```
public bool MONITORCOLLISION { init; }
```

Property Value

[bool](#)

MONITORCOLLISIONALPHA

```
public bool MONITORCOLLISIONALPHA { init; }
```

Property Value

[bool](#) ↗

PRELOAD

```
public bool PRELOAD { init; }
```

Property Value

[bool](#) ↗

PRIORITY

```
public int PRIORITY { init; }
```

Property Value

[int](#) ↗

RELEASE

```
public bool RELEASE { init; }
```

Property Value

[bool](#) ↗

TOCANVAS

```
public bool TOCANVAS { init; }
```

Property Value

bool ↗

VISIBLE

```
public bool VISIBLE { init; }
```

Property Value

bool ↗

Methods

GETALPHA()

```
public void GETALPHA()
```

GETHEIGHT()

```
public void GETHEIGHT()
```

GETPIXEL()

```
public void GETPIXEL()
```

GETPOSITIONX()

```
public void GETPOSITIONX()
```

GETPOSITIONY()

```
public void GETPOSITIONY()
```

GETWIDTH()

```
public void GETWIDTH()
```

HIDE()

```
public void HIDE()
```

INVALIDATE()

```
public void INVALIDATE()
```

ISVISIBLE()

```
public void ISVISIBLE()
```

LOAD()

```
public void LOAD()
```

MERGEALPHA(int, int, string)

```
public void MERGEALPHA(int x_offset, int y_offset, string object_name)
```

Parameters

x_offset [int ↗](#)

y_offset [int ↗](#)

object_name [string ↗](#)

MOVE(int, int)

```
public void MOVE(int x_offset, int y_offset)
```

Parameters

x_offset [int ↗](#)

y_offset [int ↗](#)

SETASBUTTON()

```
public void SETASBUTTON()
```

SETCLIPPING(int, int, int, int)

```
public void SETCLIPPING(int left_x, int top_y, int _width, int _height)
```

Parameters

left_x [int ↗](#)

top_y [int ↗](#)

_width [int ↗](#)

_height [int](#)

SETOPACITY()

```
public void SETOPACITY()
```

SETPOSITION(int, int)

```
public void SETPOSITION(int x, int y)
```

Parameters

x [int](#)

y [int](#)

SETPRIORITY()

```
public void SETPRIORITY()
```

SHOW()

```
public void SHOW()
```

Events

ONCLICK

```
public event SignalHandler? ONCLICK
```

Event Type

[SignalHandler](#)

ONFOCUSOFF

```
public event SignalHandler? ONFOCUSOFF
```

Event Type

[SignalHandler](#)

ONFOCUSON

```
public event SignalHandler? ONFOCUSON
```

Event Type

[SignalHandler](#)

ONINIT

```
public event SignalHandler? ONINIT
```

Event Type

[SignalHandler](#)

Class INTEGER

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class INTEGER : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ INTEGER

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↳ , [object.Equals\(object, object\)](#) ↳ , [object.GetHashCode\(\)](#) ↳ ,
[object.GetType\(\)](#) ↳ , [object.MemberwiseClone\(\)](#) ↳ , [object.ReferenceEquals\(object, object\)](#) ↳ ,
[object.ToString\(\)](#) ↳

Properties

TOINI

```
public bool TOINI { init; }
```

Property Value

[bool](#) ↳

VALUE

```
public int VALUE { init; }
```

Property Value

[int](#) ↳

VARTYPE

```
public string VARTYPE { init; }
```

Property Value

[string](#)

Methods

ABS(int)

Sets the modulus of [value](#) as the value of the object.

```
public int ABS(int value)
```

Parameters

[value](#) [int](#)

The value of which modulus is to be set as the value of the object.

Returns

[int](#)

ADD(int)

```
public int ADD(int summand)
```

Parameters

[summand](#) [int](#)

Returns

[int](#)

AND(int)

```
public int AND(int operand)
```

Parameters

operand [int](#)

Returns

[int](#)

CLAMP(int, int)

```
public int CLAMP(int min, int max)
```

Parameters

min [int](#)

max [int](#)

Returns

[int](#)

DEC()

```
public void DEC()
```

DIV(int)

```
public void DIV(int divisor)
```

Parameters

divisor [int ↗](#)

INC()

```
public void INC()
```

LENGTH(int, int)

```
public int LENGTH(int horizontal_distance, int vertical_distance)
```

Parameters

horizontal_distance [int ↗](#)

vertical_distance [int ↗](#)

Returns

[int ↗](#)

MOD(int)

```
public void MOD(int divisor)
```

Parameters

divisor [int ↗](#)

MUL(int)

```
public void MUL(int multiplier)
```

Parameters

multiplier [int](#)

OR(int)

```
public int OR(int operand)
```

Parameters

operand [int](#)

Returns

[int](#)

RANDOM(int)

```
public int RANDOM(int max_exclusive)
```

Parameters

max_exclusive [int](#)

Returns

[int](#)

RANDOM(int, int)

```
public int RANDOM(int summand, int max_exclusive)
```

Parameters

summand [int](#)

`max_exclusive` [int ↗](#)

Returns

[int ↗](#)

RESETINI()

`public void RESETINI()`

SET(int)

`public void SET(int value)`

Parameters

`value` [int ↗](#)

SUB(int)

`public int SUB(int subtrahend)`

Parameters

`subtrahend` [int ↗](#)

Returns

[int ↗](#)

SWITCH(int, int)

`public void SWITCH(int value1, int value2)`

Parameters

value1 [int](#)

value2 [int](#)

Events

ONBRUTALCHANGED

```
public event SignalHandler<stringified_value>? ONBRUTALCHANGED
```

Event Type

[SignalHandler<stringified_value>](#)

ONCHANGED

```
public event SignalHandler<stringified_value>? ONCHANGED
```

Event Type

[SignalHandler<stringified_value>](#)

ONINIT

```
public event SignalHandler? ONINIT
```

Event Type

[SignalHandler](#)

ONSIGNAL

```
public event SignalHandler<signal_name>? ONSIGNAL
```

Event Type

[SignalHandler<signal_name>](#)

Class KEYBOARD

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class KEYBOARD : OBJECT
```

Inheritance

[object](#) ← [OBJECT](#) ← KEYBOARD

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Methods

DISABLE()

```
public void DISABLE()
```

ENABLE()

```
public void ENABLE()
```

GETLATESTKEY()

```
public void GETLATESTKEY()
```

IENABLED()

```
public bool IENABLED()
```

Returns

[bool](#)

ISKEYDOWN()

```
public bool ISKEYDOWN()
```

Returns

[bool](#)

SETAUTOREPEAT()

```
public void SETAUTOREPEAT()
```

Events

ONCHAR

```
public event SignalHandler<char_name>? ONCHAR
```

Event Type

[SignalHandler<char_name>](#)

ONKEYDOWN

```
public event SignalHandler? ONKEYDOWN
```

Event Type

[SignalHandler](#)

ONKEYUP

```
public event SignalHandler? ONKEYUP
```

Event Type

[SignalHandler](#)

Class MOUSE

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class MOUSE : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ MOUSE

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

RAW

```
public int? RAW { init; }
```

Property Value

[int](#)?

Methods

DISABLE()

```
public void DISABLE()
```

DISABLESIGNAL()

```
public void DISABLESIGNAL()
```

ENABLE()

```
public void ENABLE()
```

ENABLESIGNAL()

```
public void ENABLESIGNAL()
```

GETPOSX()

```
public int GETPOSX()
```

Returns

[int](#)

GETPOSY()

```
public int GETPOSY()
```

Returns

[int](#)

HIDE()

```
public void HIDE()
```

ISLBUTTONDOWN()

```
public bool ISLBUTTONDOWN()
```

Returns

[bool](#)

SET()

```
public void SET()
```

SETCLIPRECT()

```
public void SETCLIPRECT()
```

SETPOSITION(int, int)

```
public void SETPOSITION(int x, int y)
```

Parameters

x [int](#)

y [int](#)

SHOW()

```
public void SHOW()
```

Events

ONCLICK

```
public event SignalHandler<button_name>? ONCLICK
```

Event Type

[SignalHandler<button_name>](#)

ONDBLCLICK

```
public event SignalHandler? ONDBLCLICK
```

Event Type

[SignalHandler](#)

ONINIT

```
public event SignalHandler? ONINIT
```

Event Type

[SignalHandler](#)

ONMOVE

```
public event SignalHandler? ONMOVE
```

Event Type

[SignalHandler](#)

ONRELEASE

```
public event SignalHandler? ONRELEASE
```

Event Type

[SignalHandler](#)

Class MULTIARRAY

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class MULTIARRAY : OBJECT
```

Inheritance

[object](#) ↗ ← [OBJECT](#) ← MULTIARRAY

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↗ , [object.Equals\(object, object\)](#) ↗ , [object.GetHashCode\(\)](#) ↗ ,
[object.GetType\(\)](#) ↗ , [object.MemberwiseClone\(\)](#) ↗ , [object.ReferenceEquals\(object, object\)](#) ↗ ,
[object.ToString\(\)](#) ↗

Properties

DIMENSIONS

```
public int DIMENSIONS { init; }
```

Property Value

[int](#) ↗

Methods

GET(params int[])

```
public variable? GET(params int[] indices)
```

Parameters

`indices int[]`

Returns

variable

SET(variable, params int[])

```
public void SET(variable value, params int[] indices)
```

Parameters

`value variable`

`indices int[]`

Class MUSIC

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class MUSIC : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ MUSIC

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

FILENAME

```
public string FILENAME { init; }
```

Property Value

[string](#)

Methods

PLAY()

```
public void PLAY()
```

Class PATTERN

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class PATTERN : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ PATTERN

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↳ , [object.Equals\(object, object\)](#) ↳ , [object.GetHashCode\(\)](#) ↳ ,
[object.GetType\(\)](#) ↳ , [object.MemberwiseClone\(\)](#) ↳ , [object.ReferenceEquals\(object, object\)](#) ↳ ,
[object.ToString\(\)](#) ↳

Properties

GRIDX

```
public int GRIDX { init; }
```

Property Value

[int](#) ↳

GRIDY

```
public int GRIDY { init; }
```

Property Value

[int](#) ↳

HEIGHT

```
public int HEIGHT { init; }
```

Property Value

[int ↗](#)

LAYERS

```
public int LAYERS { init; }
```

Property Value

[int ↗](#)

PRIORITY

```
public int PRIORITY { init; }
```

Property Value

[int ↗](#)

TOCANVAS

```
public bool TOCANVAS { init; }
```

Property Value

[bool ↗](#)

VISIBLE

```
public bool VISIBLE { init; }
```

Property Value

[bool](#)

WIDTH

```
public int WIDTH { init; }
```

Property Value

[int](#)

Methods

ADD(string, int, int, string, int)

```
public void ADD(string _, int x, int y, string object_name, int _2)
```

Parameters

_ [string](#)

x [int](#)

y [int](#)

object_name [string](#)

_2 [int](#)

GETGRAPHICSAT(int, int, bool, bool, int)

```
public string GETGRAPHICSAT(int x, int y, bool _, bool _2, int _3)
```

Parameters

x [int](#)

y [int](#)

_ [bool](#)

_2 [bool](#)

_3 [int](#)

Returns

[string](#)

MOVE(int, int)

```
public void MOVE(int x, int y)
```

Parameters

x [int](#)

y [int](#)

Class RAND

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class RAND : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ RAND

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↳ , [object.Equals\(object, object\)](#) ↳ , [object.GetHashCode\(\)](#) ↳ ,
[object.GetType\(\)](#) ↳ , [object.MemberwiseClone\(\)](#) ↳ , [object.ReferenceEquals\(object, object\)](#) ↳ ,
[object.ToString\(\)](#) ↳

Methods

GET(int)

```
public int GET(int max_exclusive)
```

Parameters

max_exclusive [int](#) ↳

Returns

[int](#) ↳

GET(int, int)

```
public int GET(int summand, int max_exclusive)
```

Parameters

summand [int](#)

max_exclusive [int](#)

Returns

[int](#)

GETPLENTY(string, int, int, int, bool)

```
public void GETPLENTY(string arr_name, int _, int _2, int _3, bool _4)
```

Parameters

arr_name [string](#)

_ [int](#)

_2 [int](#)

_3 [int](#)

_4 [bool](#)

Class SCENE

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class SCENE : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ SCENE

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

AUTHOR

```
public string AUTHOR { init; }
```

Property Value

[string](#)

BACKGROUND

```
public string BACKGROUND { init; }
```

Property Value

[string](#)

CREATIONTIME

```
public string CREATIONTIME { init; }
```

Property Value

[string](#)

DLLS

```
public string[] DLLS { init; }
```

Property Value

[string](#)[]

LASTMODIFYTIME

```
public string LASTMODIFYTIME { init; }
```

Property Value

[string](#)

MUSIC

```
public string MUSIC { init; }
```

Property Value

[string](#)

PATH

```
public string PATH { init; }
```

Property Value

[string](#)

VERSION

```
public string VERSION { init; }
```

Property Value

[string](#)

Methods

GETMAXHSPRIORITY()

```
public void GETMAXHSPRIORITY()
```

GETMINHSPRIORITY()

```
public void GETMINHSPRIORITY()
```

GETPLAYINGANIMO()

```
public void GETPLAYINGANIMO()
```

GETPLAYINGSEQ()

```
public void GETPLAYINGSEQ()
```

PAUSE()

```
public void PAUSE()
```

REMOVECLONES()

```
public void REMOVECLONES()
```

RESUME()

```
public void RESUME()
```

RUN(string, string, params variable[])

```
public variable? RUN(string object_name, string method_name, params  
variable[] arguments)
```

Parameters

object_name [string](#)

method_name [string](#)

arguments [variable](#)[]

Returns

[variable](#)

RUNCLONES()

```
public void RUNCLONES()
```

SETMAXHSPRIORITY()

```
public void SETMAXHSPRIORITY()
```

SETMINHSPRIORITY()

```
public void SETMINHSPRIORITY()
```

SETMUSICVOLUME(int)

```
public void SETMUSICVOLUME(int volume)
```

Parameters

volume [int](#)

STARTMUSIC()

```
public void STARTMUSIC()
```

STOPMUSIC()

```
public void STOPMUSIC()
```

Class SEQUENCE

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class SEQUENCE : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ SEQUENCE

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

FILENAME

```
public string FILENAME { init; }
```

Property Value

[string](#)

Methods

GETEVENTNAME()

```
public string GETEVENTNAME()
```

Returns

string ↴

HIDE()

```
public void HIDE()
```

ISPLAYING()

```
public bool ISPLAYING()
```

Returns

bool ↴

PAUSE()

```
public void PAUSE()
```

PLAY(string)

```
public void PLAY(string parameter)
```

Parameters

parameter string ↴

RESUME()

```
public void RESUME()
```

STOP(bool)

```
public void STOP(bool emit_on_finished = true)
```

Parameters

emit_on_finished [bool](#)

Events

ONFINISHED

```
public event SignalHandler<parameter>? ONFINISHED
```

Event Type

[SignalHandler<parameter>](#)

ONINIT

```
public event SignalHandler? ONINIT
```

Event Type

[SignalHandler](#)

ONSTARTED

```
public event SignalHandler<parameter>? ONSTARTED
```

Event Type

[SignalHandler<parameter>](#)

Class SOUND

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class SOUND : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ SOUND

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

FILENAME

```
public string FILENAME { init; }
```

Property Value

[string](#)

FLUSHAFTERPLAYED

```
public bool FLUSHAFTERPLAYED { init; }
```

Property Value

[bool](#)

PRELOAD

```
public bool PRELOAD { init; }
```

Property Value

[bool](#) ↗

RELEASE

```
public bool RELEASE { init; }
```

Property Value

[bool](#) ↗

Methods

ISPLAYING()

```
public bool ISPLAYING()
```

Returns

[bool](#) ↗

LOAD(string)

```
public void LOAD(string filename)
```

Parameters

[filename](#) [string](#) ↗

PAUSE()

```
public void PAUSE()
```

PLAY()

```
public void PLAY()
```

RESUME()

```
public void RESUME()
```

SETVOLUME(int)

```
public void SETVOLUME(int volume)
```

Parameters

volume [int](#)

STOP()

```
public void STOP()
```

Events

ONFINISHED

```
public event SignalHandler? ONFINISHED
```

Event Type

[SignalHandler](#)

ONINIT

```
public event SignalHandler? ONINIT
```

Event Type

[SignalHandler](#)

ONSTARTED

```
public event SignalHandler? ONSTARTED
```

Event Type

[SignalHandler](#)

Class STATICFILTER

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class STATICFILTER : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ STATICFILTER

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

ACTION

```
public string ACTION { init; }
```

Property Value

[string](#)

Methods

LINK(string)

```
public void LINK(string graphics_name)
```

Parameters

`graphics_name` [string](#)

SETPROPERTY(string, variable)

`public void SETPROPERTY(string key, variable value)`

Parameters

`key` [string](#)

`value` [variable](#)

UNLINK(string)

`public void UNLINK(string graphics_name)`

Parameters

`graphics_name` [string](#)

Class STRING

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class STRING : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ STRING

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

TOINI

```
public bool TOINI { init; }
```

Property Value

[bool](#)

VALUE

```
public string VALUE { init; }
```

Property Value

[string](#)

Methods

ADD(string)

```
public string ADD(string suffix)
```

Parameters

suffix [string](#)

Returns

[string](#)

COPYFILE(string, string)

```
public bool COPYFILE(string filename, string copied_filename)
```

Parameters

filename [string](#)

copied_filename [string](#)

Returns

[bool](#)

CUT(int, int)

```
public void CUT(int index, int length)
```

Parameters

index [int](#)

length [int](#)

FIND(string, int)

```
public int FIND(string needle, int start_index = 0)
```

Parameters

needle [string](#)

start_index [int](#)

Returns

[int](#)

GET(int)

```
public string GET(int start_index)
```

Parameters

start_index [int](#)

Returns

[string](#)

GET(int, int)

```
public string GET(int start_index, int length)
```

Parameters

start_index [int](#)

length [int](#)

Returns

[string](#)

LENGTH()

```
public int LENGTH()
```

Returns

[int](#)

REPLACE(string, string)

```
public void REPLACE(string search, string replace)
```

Parameters

search [string](#)

replace [string](#)

REPLACEAT(int, string)

```
public void REPLACEAT(int index, string replace)
```

Parameters

index [int](#)

replace [string](#)

RESETINI()

```
public void RESETINI()
```

SET(string)

```
public void SET(string value)
```

Parameters

value [string](#)

SUB(int, int)

```
public void SUB(int index, int length)
```

Parameters

index [int](#)

length [int](#)

UPPER()

```
public void UPPER()
```

Events

ONBRUTALCHANGED

```
public event SignalHandler<stringified_value>? ONBRUTALCHANGED
```

Event Type

[SignalHandler<stringified_value>](#)

ONCHANGED

`public event SignalHandler<stringified_value>? ONCHANGED`

Event Type

[SignalHandler<stringified_value>](#)

ONINIT

`public event SignalHandler? ONINIT`

Event Type

[SignalHandler](#)

Class STRUCT

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class STRUCT : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ STRUCT

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

FIELDS

```
public (string, string)[] FIELDS { init; }
```

Property Value

[\(string\)](#) , [string](#))[]

Methods

GETFIELD(string)

```
public variable GETFIELD(string field_name)
```

Parameters

`field_name` [string](#)

Returns

[variable](#)

SET(string)

```
public void SET(string struct_name)
```

Parameters

`struct_name` [string](#)

SETFIELD(string, variable)

```
public void SETFIELD(string field_name, variable value)
```

Parameters

`field_name` [string](#)

`value` [variable](#)

Class SYSTEM

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class SYSTEM : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ SYSTEM

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Methods

GETDATE()

```
public string GETDATE()
```

Returns

[string](#)

GETMHZ()

```
public int GETMHZ()
```

Returns

[int](#)

GETMINUTES()

```
public int GETMINUTES()
```

Returns

[int](#)

GETSECONDS()

```
public int GETSECONDS()
```

Returns

[int](#)

GETSYSTEMTIME()

```
public int GETSYSTEMTIME()
```

Returns

[int](#)

Class TEXT

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class TEXT : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ TEXT

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

FONT

```
public string FONT { init; }
```

Property Value

[string](#)

HJUSTIFY

```
public bool HJUSTIFY { init; }
```

Property Value

[bool](#)

HYPertext

```
public bool HYPertext { init; }
```

Property Value

[bool](#) ↗

MONITORCOLLISION

```
public bool MONITORCOLLISION { init; }
```

Property Value

[bool](#) ↗

MONITORCOLLISIONALPHA

```
public bool MONITORCOLLISIONALPHA { init; }
```

Property Value

[bool](#) ↗

RECT

```
public rect RECT { init; }
```

Property Value

[rect](#)

TEXT[]

```
public string TEXT { init; }
```

Property Value

[string](#)

TOCANVAS

```
public bool TOCANVAS { init; }
```

Property Value

[bool](#)

VISIBLE

```
public bool VISIBLE { init; }
```

Property Value

[bool](#)

VJUSTIFY

```
public bool VJUSTIFY { init; }
```

Property Value

[bool](#)

Methods

HIDE()

```
public void HIDE()
```

SETCOLOR()

```
public void SETCOLOR()
```

SETJUSTIFY()

```
public void SETJUSTIFY()
```

SETPOSITION()

```
public void SETPOSITION()
```

SETTEXT(string)

```
public void SETTEXT(string text)
```

Parameters

text string

SHOW()

```
public void SHOW()
```

Events

ONINIT

```
public event SignalHandler? ONINIT
```

Event Type

[SignalHandler](#)

Class TIMER

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class TIMER : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ TIMER

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

ELAPSE

```
public int ELAPSE { init; }
```

Property Value

[int](#)

ENABLED

```
public bool ENABLED { init; }
```

Property Value

[bool](#)

TICKS

```
public int TICKS { init; }
```

Property Value

[int](#)

Methods

DISABLE()

```
public void DISABLE()
```

ENABLE()

```
public void ENABLE()
```

GETTICKS()

```
public int GETTICKS()
```

Returns

[int](#)

RESET()

```
public void RESET()
```

SET(int)

```
public void SET(int _)
```

Parameters

_ [int](#)

SETELAPSE(int)

```
public void SETELAPSE(int _)
```

Parameters

_ [int](#)

Events

ONINIT

```
public event SignalHandler? ONINIT
```

Event Type

[SignalHandler](#)

ONTICK

```
public event SignalHandler<stringified_tick_number>? ONTICK
```

Event Type

[SignalHandler<stringified_tick_number>](#)

Class VECTOR

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class VECTOR : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ VECTOR

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

SIZE

```
public int SIZE { init; }
```

Property Value

[int](#) ↳

VALUE

```
public double[] VALUE { init; }
```

Property Value

[double](#) ↳ []

Methods

ADD(string)

```
public void ADD(string summand_name)
```

Parameters

summand_name [string](#)

ASSIGN(params double[])

```
public void ASSIGN(params double[] values)
```

Parameters

values [double](#)[]

GET(int)

```
public double GET(int index)
```

Parameters

index [int](#)

Returns

[double](#)

LEN()

```
public double LEN()
```

Returns

double ↴

MUL(double)

```
public void MUL(double multiplier)
```

Parameters

multiplier double ↴

NORMALIZE()

```
public void NORMALIZE()
```

REFLECT(string, string)

```
public void REFLECT(string normal_name, string result_name)
```

Parameters

normal_name string ↴

result_name string ↴

Class VIRTUALGRAPHICSOBJECT

Namespace: [PIKLib](#)

Assembly: PIKLib.dll

```
public class VIRTUALGRAPHICSOBJECT : OBJECT
```

Inheritance

[object](#) ↳ [OBJECT](#) ↳ VIRTUALGRAPHICSOBJECT

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Properties

ASBUTTON

```
public bool ASBUTTON { init; }
```

Property Value

[bool](#)

MASK

```
public string MASK { init; }
```

Property Value

[string](#)

MONITORCOLLISION

```
public bool MONITORCOLLISION { init; }
```

Property Value

[bool](#) ↗

MONITORCOLLISIONALPHA

```
public bool MONITORCOLLISIONALPHA { init; }
```

Property Value

[bool](#) ↗

PRIORITY

```
public int PRIORITY { init; }
```

Property Value

[int](#) ↗

SOURCE

```
public string SOURCE { init; }
```

Property Value

[string](#) ↗

TOCANVAS

```
public bool TOCANVAS { init; }
```

Property Value

[bool](#)

VISIBLE

```
public bool VISIBLE { init; }
```

Property Value

[bool](#)

Methods

GETHEIGHT()

```
public int GETHEIGHT()
```

Returns

[int](#)

GETPOSITIONX()

```
public int GETPOSITIONX()
```

Returns

[int](#)

GETPOSITIONY()

```
public int GETPOSITIONY()
```

Returns

[int](#)

GETWIDTH()

```
public int GETWIDTH()
```

Returns

[int](#)

MOVE(int, int)

```
public void MOVE(int x_offset, int y_offset)
```

Parameters

x_offset [int](#)

y_offset [int](#)

SETMASK(string)

```
public void SETMASK(string graphics_name)
```

Parameters

graphics_name [string](#)

SETPOSITION(int, int)

```
public void SETPOSITION(int x, int y)
```

Parameters

x [int](#)

y [int](#)

SETPRIORITY(int)

```
public void SETPRIORITY(int priority)
```

Parameters

priority [int](#)

SETSOURCE(string)

```
public void SETSOURCE(string graphics_name)
```

Parameters

graphics_name [string](#)

Namespace World

Classes

WORLD

3D physics simulation.

Class WORLD

Namespace: [World](#)

Assembly: World.dll

3D physics simulation.

```
public class WORLD : OBJECT
```

Inheritance

[object](#) ↴ ← [OBJECT](#) ← WORLD

Inherited Members

[OBJECT.DESCRIPTION](#) , [OBJECT.TYPE](#) , [OBJECT.ADDBEHAVIOUR\(string, string\)](#) ,
[OBJECT.CLONE\(int\)](#) , [OBJECT.GETCLONEINDEX\(\)](#) , [OBJECT.GETNAME\(\)](#) ,
[OBJECT.MSGBOX\(string\)](#) , [OBJECT.REMOVEBEHAVIOUR\(string\)](#) , [OBJECT.RESETCLONES\(\)](#) ,
[object.Equals\(object\)](#) ↴ , [object.Equals\(object, object\)](#) ↴ , [object.GetHashCode\(\)](#) ↴ ,
[object.GetType\(\)](#) ↴ , [object.MemberwiseClone\(\)](#) ↴ , [object.ReferenceEquals\(object, object\)](#) ↴ ,
[object.ToString\(\)](#) ↴

Properties

FILENAME

```
public string FILENAME { init; }
```

Property Value

[string](#) ↴

Methods

ADDBODY()

```
public void ADDBODY()
```

ADDFORCE()

```
public void ADDFORCE()
```

ADDGRAVITYEX()

```
public void ADDGRAVITYEX()
```

FINDPATH()

```
public void FINDPATH()
```

FOLLOWPATH()

```
public void FOLLOWPATH()
```

GETANGLE()

```
public void GETANGLE()
```

GETBKGPOSX()

```
public void GETBKGPOSX()
```

GETBKGPOSY()

```
public void GETBKGPOSY()
```

GETMOVEDISTANCE()

```
public void GETMOVEDISTANCE()
```

GETPOSITIONX()

```
public void GETPOSITIONX()
```

GETPOSITIONY()

```
public void GETPOSITIONY()
```

GETPOSITIONZ()

```
public void GETPOSITIONZ()
```

GETROTATIONZ()

```
public void GETROTATIONZ()
```

GETSPEED()

```
public void GETSPEED()
```

JOIN()

```
public void JOIN()
```

LINK()

```
public void LINK()
```

LOAD()

```
public void LOAD()
```

MOVEOBJECTS()

```
public void MOVEOBJECTS()
```

REMOVEOBJECT()

```
public void REMOVEOBJECT()
```

SETACTIVE()

```
public void SETACTIVE()
```

SETBKGSIZE()

```
public void SETBKGSIZE()
```

SETBODYDYNAMICS()

```
public void SETBODYDYNAMICS()
```

SETG()

```
public void SETG()
```

SETGRAVITY()

```
public void SETGRAVITY()
```

SETGRAVITYCENTER()

```
public void SETGRAVITYCENTER()
```

SETLIMIT()

```
public void SETLIMIT()
```

SETMAXSPEED()

```
public void SETMAXSPEED()
```

SETMOVEFLAGS()

```
public void SETMOVEFLAGS()
```

SETPOSITION()

```
public void SETPOSITION()
```

SETREFOBJECT()

```
public void SETREFOBJECT()
```

SETVELOCITY()

```
public void SETVELOCITY()
```

START()

```
public void START()
```

STOP()

```
public void STOP()
```

UNLINK()

```
public void UNLINK()
```

Namespace abstractions

Classes

[BoolVariable](#)

[DoubleVariable](#)

[IntVariable](#)

[LiteralRect](#)

[OBJECT](#)

[ReferenceRect](#)

[StringVariable](#)

[button_name](#)

[char_name](#)

[event_name](#)

[other_name](#)

[parameter](#)

[rect](#)

[signal_name](#)

[stringified_tick_number](#)

[stringified_value](#)

[variable](#)

Enums

[anchor](#)

[complex_operator](#)

[condition_operator](#)

[expression_operator](#)

Delegates

[SignalHandler](#)

A plain non-parametrized signal handler. It can receive an arbitrary number of arguments.

[SignalHandler<P>](#)

A specialized signal handler identified by its parametr. It can receive an arbitrary number of arguments.

Class BoolVariable

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record BoolVariable : variable, IEquatable<variable>,
IEquatable<BoolVariable>
```

Inheritance

[object](#) ← [variable](#) ← BoolVariable

Implements

[IEquatable](#)<[variable](#)>, [IEquatable](#)<[BoolVariable](#)>

Inherited Members

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

Constructors

BoolVariable(bool)

```
public BoolVariable(bool value)
```

Parameters

value [bool](#)

Properties

value

```
public bool value { get; init; }
```

Property Value

bool

Class DoubleVariable

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record DoubleVariable : variable, IEquatable<variable>,
IEquatable<DoubleVariable>
```

Inheritance

[object](#) ← [variable](#) ← DoubleVariable

Implements

[IEquatable](#)<[variable](#)>, [IEquatable](#)<[DoubleVariable](#)>

Inherited Members

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

Constructors

DoubleVariable(double)

```
public DoubleVariable(double value)
```

Parameters

value [double](#)

Properties

value

```
public double value { get; init; }
```

Property Value

double ↗

Class IntVariable

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record IntVariable : variable, IEquatable<variable>, IEquatable<IntVariable>
```

Inheritance

[object](#) ← [variable](#) ← IntVariable

Implements

[IEquatable](#)<[variable](#)>, [IEquatable](#)<[IntVariable](#)>

Inherited Members

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

Constructors

IntVariable(int)

```
public IntVariable(int value)
```

Parameters

value [int](#)

Properties

value

```
public int value { get; init; }
```

Property Value

Class LiteralRect

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record LiteralRect : rect, IEquatable<rect>, IEquatable<LiteralRect>
```

Inheritance

[object](#) ↴ ← [rect](#) ← LiteralRect

Implements

[IEquatable](#) ↴ <[rect](#)>, [IEquatable](#) ↴ <[LiteralRect](#)>

Inherited Members

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

Constructors

LiteralRect(int, int, int, int)

```
public LiteralRect(int left_x, int top_y, int right_x, int bottom_y)
```

Parameters

left_x [int](#) ↴

top_y [int](#) ↴

right_x [int](#) ↴

bottom_y [int](#) ↴

Properties

bottom_y

```
public int bottom_y { get; init; }
```

Property Value

[int ↗](#)

left_x

```
public int left_x { get; init; }
```

Property Value

[int ↗](#)

right_x

```
public int right_x { get; init; }
```

Property Value

[int ↗](#)

top_y

```
public int top_y { get; init; }
```

Property Value

[int ↗](#)

Class OBJECT

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public abstract class OBJECT
```

Inheritance

[object](#) ← OBJECT

Derived

[INERTIA](#), [MATRIX](#), [ANIMO](#), [APPLICATION](#), [ARRAY](#), [BEHAVIOUR](#), [BOOL](#), [BUTTON](#),
[CANVAS_OBSERVER](#), [CLASS](#), [CNVLOADER](#), [COMPLEXCONDITION](#), [CONDITION](#), [DATABASE](#),
[DOUBLE](#), [EPISODE](#), [EXPRESSION](#), [FILTER](#), [FONT](#), [GROUP](#), [IMAGE](#), [INTEGER](#), [KEYBOARD](#),
[MOUSE](#), [MULTIARRAY](#), [MUSIC](#), [PATTERN](#), [RAND](#), [SCENE](#), [SEQUENCE](#), [SOUND](#), [STATICFILTER](#),
[STRING](#), [STRUCT](#), [SYSTEM](#), [TEXT](#), [TIMER](#), [VECTOR](#), [VIRTUALGRAPHICSOBJECT](#), [WORLD](#)

Inherited Members

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

Properties

DESCRIPTION

A short description of the object.

```
public string? DESCRIPTION { init; }
```

Property Value

[string](#)

TYPE

The type of the object.

```
public string TYPE { init; }
```

Property Value

[string](#)

Methods

ADDBEHAVIOUR(string, string)

Adds/overwrites a handler for the given signal.

```
public void ADDBEHAVIOUR(string signal_name, string code)
```

Parameters

[signal_name](#) [string](#)

The (parametrized) name of the signal.

[code](#) [string](#)

A handler (code block or a BEHAVIOUR name).

CLONE(int)

Sets the clone count for the object. If it is greater than current clone count, new clones are created.

```
public void CLONE(int count = 1)
```

Parameters

[count](#) [int](#)

Desired clone count.

GETCLONEINDEX()

Returns the clone index of the object.

```
public int GETCLONEINDEX()
```

Returns

[int](#)

0 if the object is the original, unique sequential 1-based index otherwise.

GETNAME()

Retrieves the name of the object.

```
public string GETNAME()
```

Returns

[string](#)

The object name.

MSGBOX(string)

Displays a message box with the given contents.

```
public void MSGBOX(string message)
```

Parameters

[message](#) [string](#)

A message to be displayed.

Remarks

Does nothing in the release build of the original engine.

REMOVEBEHAVIOUR(string)

Removes the handler for the given signal (if it exists).

```
public void REMOVEBEHAVIOUR(string signal_name)
```

Parameters

signal_name [string](#)

The (parametrized) name of the signal.

RESETCLONES()

Resets the clone count.

```
public void RESETCLONES()
```

Class ReferenceRect

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record ReferenceRect : rect, IEquatable<rect>, IEquatable<ReferenceRect>
```

Inheritance

[object](#) ↴ ← [rect](#) ← ReferenceRect

Implements

[IEquatable](#) ↴ <[rect](#)>, [IEquatable](#) ↴ <[ReferenceRect](#)>

Inherited Members

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

Constructors

ReferenceRect(string)

```
public ReferenceRect(string object_name)
```

Parameters

object_name [string](#) ↴

Properties

object_name

```
public string object_name { get; init; }
```

Property Value

[string ↗](#)

Delegate SignalHandler

Namespace: [abstractions](#)

Assembly: PIKLib.dll

A plain non-parametrized signal handler. It can receive an arbitrary number of arguments.

```
public delegate void SignalHandler(params variable[] arguments)
```

Parameters

arguments [variable\[\]](#)

Arguments passed to the handler.

Delegate SignalHandler<P>

Namespace: [abstractions](#)

Assembly: PIKLib.dll

A specialized signal handler identified by its parameter. It can receive an arbitrary number of arguments.

```
public delegate void SignalHandler<P>(P parameter, params variable[] arguments)
```

Parameters

parameter P

The parameter identifying the handler.

arguments [variable](#)[]

Arguments passed to the handler.

Type Parameters

P

Type of the parameter (for documentation purposes).

Class StringVariable

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record StringVariable : variable, IEquatable<variable>,
IEquatable<StringVariable>
```

Inheritance

[object](#) ← [variable](#) ← StringVariable

Implements

[IEquatable](#)<[variable](#)>, [IEquatable](#)<[StringVariable](#)>

Inherited Members

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

Constructors

StringVariable(string)

```
public StringVariable(string value)
```

Parameters

value [string](#)

Properties

value

```
public string value { get; init; }
```

Property Value

[string](#) ↗

Enum anchor

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public enum anchor
```

Fields

BOTTOM = 8

CENTER = 0

LEFT = 5

LEFTLOWER = 3

LEFTUPPER = 1

RIGHT = 6

RIGHTLOWER = 4

RIGHTUPPER = 2

TOP = 7

Class button_name

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record button_name : IEquatable<button_name>
```

Inheritance

[object](#) ← button_name

Implements

[IEquatable](#)<[button_name](#)>

Inherited Members

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

Constructors

button_name(string)

```
public button_name(string value)
```

Parameters

Value [string](#)

Properties

Value

```
public string Value { get; init; }
```

Property Value

[string](#)

Operators

implicit operator button_name(string)

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

Parameters

s [string](#)

Returns

[button_name](#)

implicit operator string(button_name)

```
public static implicit operator string(button_name n)
```

Parameters

n [button_name](#)

Returns

[string](#)

Class char_name

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record char_name : IEquatable<char_name>
```

Inheritance

[object](#) ← char_name

Implements

[IEquatable](#)<char_name>

Inherited Members

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

Constructors

char_name(string)

```
public char_name(string Value)
```

Parameters

Value [string](#)

Properties

Value

```
public string Value { get; init; }
```

Property Value

[string](#)

Operators

implicit operator char_name(string)

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

Parameters

s [string](#)

Returns

[char_name](#)

implicit operator string(char_name)

```
public static implicit operator string(char_name n)
```

Parameters

n [char_name](#)

Returns

[string](#)

Enum complex_operator

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public enum complex_operator
```

Fields

AND = 0

OR = 1

Enum condition_operator

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public enum condition_operator
```

Fields

EQUAL = 0

GREATER = 3

GREATEREQUAL = 5

LESS = 2

LESSEQUAL = 4

NOTEQUAL = 1

Class event_name

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record event_name : IEquatable<event_name>
```

Inheritance

[object](#) ← event_name

Implements

[IEquatable](#)<[event_name](#)>

Inherited Members

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

Constructors

event_name(string)

```
public event_name(string Value)
```

Parameters

Value [string](#)

Properties

Value

```
public string Value { get; init; }
```

Property Value

[string](#)

Operators

implicit operator event_name(string)

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

Parameters

s [string](#)

Returns

[event_name](#)

implicit operator string(event_name)

```
public static implicit operator string(event_name n)
```

Parameters

n [event_name](#)

Returns

[string](#)

Enum expression_operator

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public enum expression_operator
```

Fields

ADD = 0

DIV = 3

MOD = 4

MUL = 2

SUB = 1

Class other_name

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record other_name : IEquatable<other_name>
```

Inheritance

[object](#) ← other_name

Implements

[IEquatable](#)<other_name>

Inherited Members

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

Constructors

other_name(string)

```
public other_name(string Value)
```

Parameters

Value [string](#)

Properties

Value

```
public string Value { get; init; }
```

Property Value

[string](#)

Operators

implicit operator other_name(string)

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

Parameters

s [string](#)

Returns

[other_name](#)

implicit operator string(other_name)

```
public static implicit operator string(other_name n)
```

Parameters

n [other_name](#)

Returns

[string](#)

Class parameter

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record parameter : IEquatable<parameter>
```

Inheritance

[object](#) ← parameter

Implements

[IEquatable](#)<parameter>

Inherited Members

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

Constructors

parameter(string)

```
public parameter(string Value)
```

Parameters

Value [string](#)

Properties

Value

```
public string Value { get; init; }
```

Property Value

[string](#)

Operators

implicit operator parameter(string)

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

Parameters

s [string](#)

Returns

[parameter](#)

implicit operator string(parameter)

```
public static implicit operator string(parameter n)
```

Parameters

n [parameter](#)

Returns

[string](#)

Class rect

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public abstract record rect : IEquatable<rect>
```

Inheritance

[object](#) ← rect

Implements

[IEquatable](#) <rect>

Derived

[LiteralRect](#), [ReferenceRect](#)

Inherited Members

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

Class signal_name

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record signal_name : IEquatable<signal_name>
```

Inheritance

[object](#) ← signal_name

Implements

[IEquatable](#)<[signal_name](#)>

Inherited Members

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

Constructors

signal_name(string)

```
public signal_name(string value)
```

Parameters

Value [string](#)

Properties

Value

```
public string Value { get; init; }
```

Property Value

[string](#)

Operators

implicit operator signal_name(string)

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

Parameters

s [string](#)

Returns

[signal_name](#)

implicit operator string(signal_name)

```
public static implicit operator string(signal_name n)
```

Parameters

n [signal_name](#)

Returns

[string](#)

Class stringified_tick_number

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record stringified_tick_number : IEquatable<stringified_tick_number>
```

Inheritance

[object](#) ← stringified_tick_number

Implements

[IEquatable](#)<[stringified_tick_number](#)>

Inherited Members

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

Constructors

stringified_tick_number(string)

```
public stringified_tick_number(string Value)
```

Parameters

Value [string](#)

Properties

Value

```
public string Value { get; init; }
```

Property Value

[string](#)

Operators

implicit operator stringified_tick_number(string)

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

Parameters

s [string](#)

Returns

[stringified_tick_number](#)

implicit operator string(stringified_tick_number)

```
public static implicit operator string(stringified_tick_number n)
```

Parameters

n [stringified_tick_number](#)

Returns

[string](#)

Class stringified_value

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public record stringified_value : IEquatable<stringified_value>
```

Inheritance

[object](#) ← stringified_value

Implements

[IEquatable](#)<[stringified_value](#)>

Inherited Members

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

Constructors

stringified_value(string)

```
public stringified_value(string value)
```

Parameters

Value [string](#)

Properties

Value

```
public string Value { get; init; }
```

Property Value

[string](#)

Operators

implicit operator stringified_value(string)

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

Parameters

s [string](#)

Returns

[stringified_value](#)

implicit operator string(stringified_value)

```
public static implicit operator string(stringified_value n)
```

Parameters

n [stringified_value](#)

Returns

[string](#)

Class variable

Namespace: [abstractions](#)

Assembly: PIKLib.dll

```
public abstract record variable : IEquatable<variable>
```

Inheritance

[object](#) ← variable

Implements

[IEquatable](#) <[variable](#)>

Derived

[BoolVariable](#), [DoubleVariable](#), [IntVariable](#), [StringVariable](#)

Inherited Members

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