Anibody

A 2d Canvas-based game engine

*Version: 1.1.9*

Inhaltsverzeichnis

[Inhaltsverzeichnis 2](#_Toc505032742)

[Preface 4](#_Toc505032743)

[About this document 4](#_Toc505032744)

[About the methodology 4](#_Toc505032745)

[Who wants to use Anibody? 4](#_Toc505032746)

[Idea of the Engine 4](#_Toc505032747)

[Installation 4](#_Toc505032748)

[Basic Concept 5](#_Toc505032749)

[The Game Cycle — Widget and EngineObject/ABO-Level 5](#_Toc505032750)

[Documentation 6](#_Toc505032751)

[Commonly used terminology 6](#_Toc505032752)

[CallbackObject 6](#_Toc505032753)

[Anibody 8](#_Toc505032754)

[Anibody.\* 13](#_Toc505032755)

[EngineObject 13](#_Toc505032756)

[ABO 15](#_Toc505032757)

[Widget 15](#_Toc505032758)

[DefaultCamera 15](#_Toc505032759)

[DefaultTerrain 15](#_Toc505032760)

[ECMAScriptExtension 15](#_Toc505032761)

[Anibody.debug.\* 15](#_Toc505032762)

[DumbObject 15](#_Toc505032763)

[Consolero 15](#_Toc505032764)

[DebugWindow 15](#_Toc505032765)

[Monitor 15](#_Toc505032766)

[Anibody.input.\* 15](#_Toc505032767)

[Anibody.nav.\* 16](#_Toc505032768)

[BoxMenu 16](#_Toc505032769)

[Gallery 16](#_Toc505032770)

[SlideMenu > Tab 16](#_Toc505032771)

[Anibody.shapes.\* 16](#_Toc505032772)

[Shape 16](#_Toc505032773)

[Rectangle 16](#_Toc505032774)

[Circle 16](#_Toc505032775)

[Triangle 16](#_Toc505032776)

[Anibody.ui.\* 16](#_Toc505032777)

[Alert 16](#_Toc505032778)

[Confirm 16](#_Toc505032779)

[Prompt 16](#_Toc505032780)

[MultipleChoice 16](#_Toc505032781)

[ColorPicker 16](#_Toc505032782)

[Button 16](#_Toc505032783)

[InputField 16](#_Toc505032784)

[Slider 16](#_Toc505032785)

[Switch 16](#_Toc505032786)

[Toaster 16](#_Toc505032787)

[Anibody.util.\* 16](#_Toc505032788)

[PriorityQueue 16](#_Toc505032789)

[Timer 16](#_Toc505032790)

[IntervalHandler 16](#_Toc505032791)

[Counter 16](#_Toc505032792)

[Flow 16](#_Toc505032793)

[MultiFlow 16](#_Toc505032794)

[Storyboard 16](#_Toc505032795)

[MediaManager 17](#_Toc505032796)

[Storage 17](#_Toc505032797)

[Task > Step 17](#_Toc505032798)

[Anibody.visual.\* 17](#_Toc505032799)

[ABOPresenter 17](#_Toc505032800)

[Animation 17](#_Toc505032801)

[ImageObject 17](#_Toc505032802)

[ABText 17](#_Toc505032803)

[CoordinateSystem 17](#_Toc505032804)

[Highlighting 17](#_Toc505032805)

[Spline 17](#_Toc505032806)

[Spotting 17](#_Toc505032807)

[Sprite 17](#_Toc505032808)

[Anibody.static.\* 17](#_Toc505032809)

[FAQ 17](#_Toc505032810)

[Where can I find an example project? 17](#_Toc505032811)

# Preface

## About this document

This documentation will cover the Anibody Engine in great detail. Please note, that not all of the components, methods and members are described here. This can have one of the following reasons:

1. The feature is not important or deprecated.
2. The feature should only be used internally (often marked by the underscore “\_” in front of the name)
3. It was simply forgotten to be added to the document. In that case, a reminder or a short message is more than welcomed.

## About the methodology

In this document, the methodology is a combination of listing the features with further description plus a short excerpt of an example (code snippets), in which the respective feature is being used.

|  |
| --- |
| // It is hoped that by doing so the learning effect will be increased |

Please see: FAQ for further examples.

## Who wants to use Anibody?

Anyone who wants to bring his or her interactive animations or games to live and does not want to start from scratch. Anyone who wants to express them on the web and does not want to discriminate the viewer for what operation system or browser they use.

## Idea of the Engine

It provides the developer a possibility to implement game elements or interactive animations with less writing effort in a short period of time.

It is HTML5-conform and hence cross-browser friendly.

## Installation

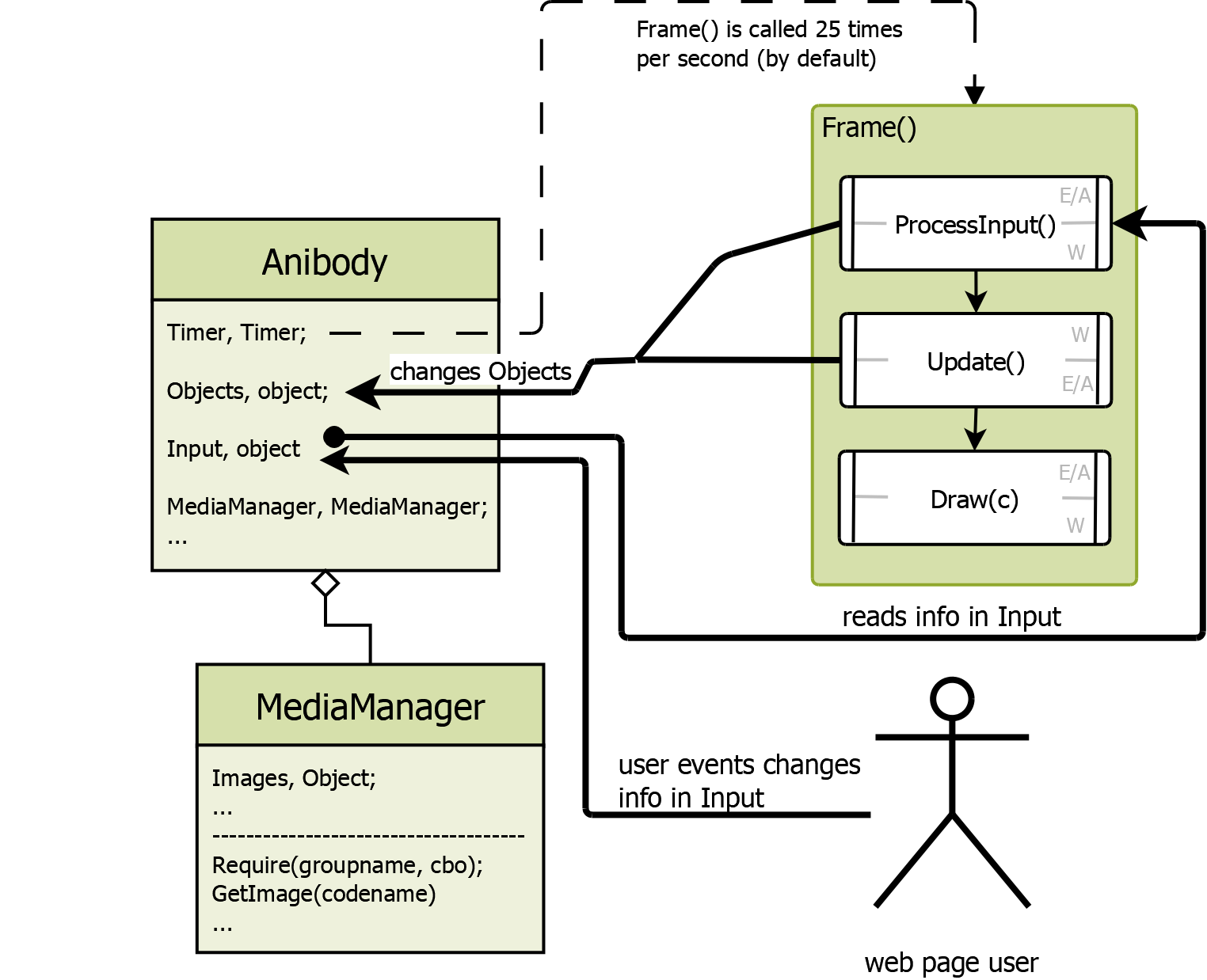
|  |
| --- |
| <!-- You do not have to use jquery but if you do than jquery should be included before anibody -->  <script type="text/javascript" src="jqueryXYZ.js"></script>  <script type="text/javascript" src="anibodyXYZ.js"></script> |

## Basic Concept

All game related objects, which are registered for the Anibody engine, are going through the “game cycle”, when the Frame() method is called. That means that their ProcessInput method, Update method and Draw method (as long as they have it) are being triggered in the respective phase of the game loop.

All registered objects belong to one of two groups, which are called EngineObject/ABO (or just ABO) level and the Widget level.

Depending on to which level a game object belongs, its respective method is triggered in a different order. The priority does only have an effect on the order of the respective method within one level.



### The Game Cycle — Widget and EngineObject/ABO-Level

**ProcessInput()**

In this phase of the game cycle, it is usual practice to read the information, which is saved in Anibody.Input, and manipulate the representative object or the environment according to your game logic.

For example: If the W key is pressed then move the representative game object 10 pixel upwards.

|  |
| --- |
| //this.Engine is a reference to the Anibody engine  if (this.Engine.Input.Keys.W.FramesPressed === 1) {  this.Move(0, -10);  } |

**Update()**

In this phase, if it is necessary to your game logic, you can check the respective object or the general environment and handle the situation accordingly.

For example: If the mouse is over the graphical representation of the object, switch the cursor of the mouse to the “pointer”.

|  |
| --- |
| if(this.IsMouseOver)  this.Engine.Input.Mouse.Cursor.pointer(); |

Note: *At the end of ProcessInput(), the engine will resolve over which object the mouse is hovering, so now in the Update() you can safely check for this.*

**Draw(c)**

In this phase every object with a graphical representation will be drawn on the canvases context (c variable).

**Level**

# Documentation

## Commonly used terminology

### CallbackObject

This is an ordinary object to capsulate a function, its parameters and the object, to which the function refers (the this variable).

Due to the following conformation, a Callback-Object mainly consists of three attributes:

|  |  |
| --- | --- |
| function | The function, which will be capsulated |
| that | The this variable in the function (optional) |
| parameter | The Parameter for the function (optional) |

Example:

|  |
| --- |
| // an object with the method 'sayHello', which expects a name as a string  var anyobject = {  sayHello : function( name ){ console.log("Hello" + name);}  };  // this is the ca  var callbackobject = {  function : function(para){  // here should be important code, which justifies  // the use of an callback object  // BUT to keep it simple- just calling the method 'sayHello'  this.sayHello( para );  },  that : anyobject,  parameter : "World"  }; |

Building such an object can be tedious because you have to use the correct attribute names.

Therefore, the Anibody engine extends the Function class of the ECMAScript with a new method. The new method is:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | getCallbackObject | |
| Description | | Encapsulates the function an returns a callback object | |
| Parameter | | | |
| Nr. | Name | Type | Description |
| 1 | that | object | Is going to be the this variable in the function (optional) |
| 2 | parameter | any | Is going to be the parameter of the function (optional) |
| 3 | useApply | boolean | if true the conformation expects the parameter to be an array and uses Function.apply(that, parameter) instead of call() (default: false) |
| Returns | | object | CallbackObject |

Example

|  |
| --- |
| // an object with the method 'sayHello', which expects a name as a string  var anyobject = {  sayHello : function( name ){ console.log("Hello" + name);}  };  var func = function(para) {  this.sayHello( para );  };  var callbackobject = func.getCallbackObject(anyobject, "World"); |

Both examples produce the same Callback Object

If you want to trigger a callback object (yes, if you want to write own components and use best practice) than you may use the Anibody method CallObject.

Example (as a code extension to the above one)

|  |
| --- |
| Anibody.CallObject( callbackobject );  // console output: Hello World |

## Anibody

Description:

This is the main class. This is the engine. By default, it contains a timer, which calls the Frame()-Method 25 times per second (known as FPS, “frames per second”). Through an option parameter, the engine’s timer can be set to a different value or being canceled completely.

Extends from: none

Parameter:

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | | | |
| Nr. | Name | Type | Description |
| 1 | html\_id | string | The id of the targeted canvas as a string |
| 2 | opt | object | Object that contains further engine configuration. (See example) |

|  |
| --- |
| var engine = new Anibody("ExampleID",{  flagConstantLoop : true,  flagPreventKeyboardStrokesToBubbleUp : true,  flagPreventContextClickBubbleToUp : true,  flagMediaManager : true,  flagMouseInput : true,  flagKeyboardInput : true,  flagTouchHandler : true,  flagTouch2FakeMouseClick : true,  flagStorage : true,  flagIntervalHandler : true,  fps : 25  }); |

Members

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| EI | integer | Engine Index of this engine |
| CurrentFrame | integer | The current frame |
| Objects | object | Object that holds the game related Objects. |
| Canvas | <canvas> | Reference to the canvas html element |
| Context | CanvasRenderingContext2D | Reference to the context of this canvas |

Methods:

|  |  |  |
| --- | --- | --- |
| Name | Start | |
| Description | Starts the Game Cycle/Loop | |
| Parameter - none | | |
| Returns | undefined | undefined |

|  |  |  |
| --- | --- | --- |
| Name | Stop | |
| Description | Stops the Game Cycle/Loop | |
| Parameter - none | | |
| Returns | undefined | undefined |

|  |  |  |
| --- | --- | --- |
| Name | Pause | |
| Description | Pauses the Game Cycle/Loop (keeps processing input and drawing) | |
| Parameter - none | | |
| Returns | undefined | undefined |

|  |  |  |
| --- | --- | --- |
| Name | Continue | |
| Description | Continues the Game Cycle/Loop | |
| Parameter - none | | |
| Returns | undefined | undefined |

|  |  |  |
| --- | --- | --- |
| Name | Frame | |
| Description | Calls one loop through the Game Cycle/Loop – Calls ProcessInput(), Update(), Draw() | |
| Parameter - none | | |
| Returns | undefined | undefined |

|  |  |  |
| --- | --- | --- |
| Name | ProcessInput | |
| Description | Calls ProcessInput() of every registered object. More Info see The Game Cycle — here Frame() | |
| Parameter - none | | |
| Returns | undefined | undefined |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | Update | |
| Description | | Calls Update() of every registered object. More Info see The Game Cycle — here Frame() | |
| Parameter - none | | | |
| Returns | | undefined | undefined |
|  | | | |
| Name | | Draw | |
| Description | | Calls Draw() of every registered object. More Info see The Game Cycle — here Frame() | |
| Parameter | | | |
| Nr. | Name | Type | Description |
| 1 | c | CanvasRenderingContext2D | The Canvas’ context, on which will be drawn (usually it is Anibody.Context) |
| Returns | | undefined | undefined |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | AddProcessInputFunctionObject | |
| Description | | Adds a Callback object to the ProcessInput part of the game loop on the Widget level. | |
| Parameter | | | |
| Nr. | Name | Type | Description |
| 1 | pio | object | Process Input function Object (is an ordinary callback object) |
| 2 | prio | number | Priority of the callback (optional) |
| Returns | | number | Reference number, which can be used to remove the callback object |

Example

|  |
| --- |
| var f = function () {  // creating an object, which describes a rectangle of some content  var area = {  x: this.ContentBox.x,  y: this.ContentBox.y,  width: this.ContentBox.width,  height: this.ContentBox.height,  type: "rect"  };  // at the end of the ProcessInput-part of the game loop  // the engine resolves over which 'area' the mouse hovers  this.Engine.Input.MouseHandler.AddHoverRequest(area, this, "IsMouseOverContent");  };  var cbo = f.getCallbackObject(this);  this.\_refnum = this.Engine.AddProcessInputFunctionObject(cbo); |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | RemoveProcessInputFunctionObject | |
| Description | | Removes a specified Callback object from the ProcessInput-part of the game loop | |
| Parameter | | | |
| Nr. | Name | Type | Description |
| 1 | ref | number | Reference number of the callback object |
| Returns | | undefined | undefined |

Example

|  |
| --- |
| // ...  this.Engine.RemoveProcessInputFunctionObject(this.\_refnum); |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | Add | |
| Description | | Adds | |
| Parameter | | | |
| Nr. | Name | Type | Description |
| 1 | pio | object |  |
| 2 | prio | number | Priority of the callback (optional) |
| Returns | | number | Reference number, which can be used to remove the callback object |

Example

|  |
| --- |
| g |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | Remove | |
| Description | | Removes a Callback object, which is triggered in the | |
| Parameter | | | |
| Nr. | Name | Type | Description |
| 1 | ref | number |  |
| Returns | | undefined | undefined |

Example

|  |
| --- |
| // ...  this.Engine.Remove (this.\_refnum); |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | Add | |
| Description | | Adds an Callback object to the ProcessInput part of the game loop | |
| Parameter | | | |
| Nr. | Name | Type | Description |
| 1 | pio | object (cbo) | Process Input function Object (is an ordinary callback object) |
| 2 | prio | number | Priority of the callback (optional) |
| Returns | | number | Reference number, which can be used to remove the callback object |

Example

|  |
| --- |
| var f = function () {  // creating an object, which describes a rectangle of some content  var area = {  x: this.ContentBox.x,  y: this.ContentBox.y,  width: this.ContentBox.width,  height: this.ContentBox.height,  type: "rect"  };  // at the end of the ProcessInput-part of the game loop  // the engine resolves over which 'area' the mouse hovers  this.Engine.Input.MouseHandler.AddHoverRequest(area, this, "IsMouseOverContent");  };  var cbo = f.getCallbackObject(this);  this.\_refnum = this.Engine.AddProcessInputFunctionObject(cbo); |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | RemoveProcessInputFunctionObject | |
| Description | | Removes a Callback object, which is triggered in the ProcessInput part of the game loop | |
| Parameter | | | |
| Nr. | Name | Type | Description |
| 1 | ref | number | Process Input function Object (is an ordinary callback object) |
| Returns | | undefined | undefined |

Example

|  |
| --- |
| // ...  this.Engine.RemoveProcessInputFunctionObject(this.\_refnum); |

## Anibody.\*

### EngineObject

Description:

Every object used in the AniBody-Engine should derive from this class. EngineObjects are used in the background of the engine, not visible to the user.

Extends from: none

Parameter: none

Members

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Engine | Anibody | A reference to the engine, to which the EngineObject belongs. |
| EI | integer | Engine Index, the index of the engine, to which this |
| UniqueID | integer | Every EngineObject receives an Id, which is unique in the area of Anibody |

Methods:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | Register | |
| Description | | Adds the EngineObject to an engine | |
| Parameter | | | |
| Nr. | Name | Type | Description |
| 1 | prior | number | The priority in the Anibody.Objects.Queue (optional) |
| 2 | ei | integer | The Engine index of the engine, to which it should belong (default = 0) |
| Returns | | integer | Reference number – can be used for Anibody.RemoveObject() |

|  |  |  |
| --- | --- | --- |
| Name | Deregister | |
| Description | Removes a registered object.  This function should be called if the instance won't be used anymore | |
| Parameter | | |
| Returns | undefined |  |

Example

|  |
| --- |
| var s = new Anibody.shapes.Shape(5,1,8,3,1,3); // triangle  s.Register(); // you may use null if you don't need to use priorities  // ... let's say the Shape served its purpose  s.Deregister(); |

### ABO

### Widget

### DefaultCamera

### DefaultTerrain

## ECMAScriptExtension

## Anibody.debug.\*

### DumbObject

### Consolero

### DebugWindow

### Monitor

## Anibody.input.\*

- not the classes are important but the structure and the meaning of the data here

## Anibody.nav.\*

### BoxMenu

### Gallery

### SlideMenu > Tab

## Anibody.shapes.\*

### Shape

### Rectangle

### Circle

### Triangle

## Anibody.ui.\*

### Alert

### Confirm

### Prompt

### MultipleChoice

### ColorPicker

### Button

### InputField

### Slider

### Switch

### Toaster

## Anibody.util.\*

### PriorityQueue

### Timer

### IntervalHandler

### Counter

### Flow

### MultiFlow

### Storyboard

### MediaManager

### Storage

### Task > Step

## Anibody.visual.\*

### ABOPresenter

### Animation

### ImageObject

### ABText

### CoordinateSystem

### Highlighting

### Spline

### Spotting

### Sprite

## Anibody.static.\*

# FAQ

## Where can I find an example project?

The Github repository is a big example. You can clone it and test it on your own server or click the link found in the README.md to test it without any further time to waste.