3dRudder
Python Module

08/07/2017

Version 1.0 for Windows

This is the Release of the 3dRudder Python Module

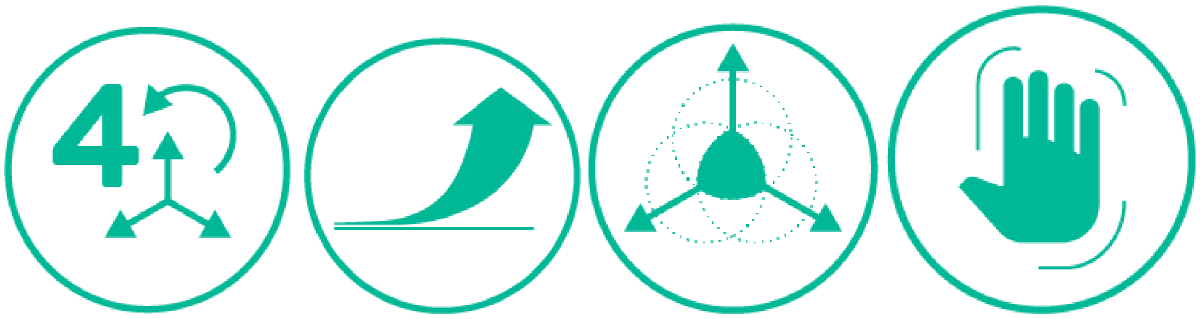
Warning: this version of the SDK works with firmware version 1.3.x.x and higher !

If you have 3dRudder version with the firmware 1.2.x.x or older, please contact us to get the updating software: support@3drudder.com

3dRudder Module

Version 0.75 for Windows (Windows 7 and later ...)

1 Description of the 3dRudder

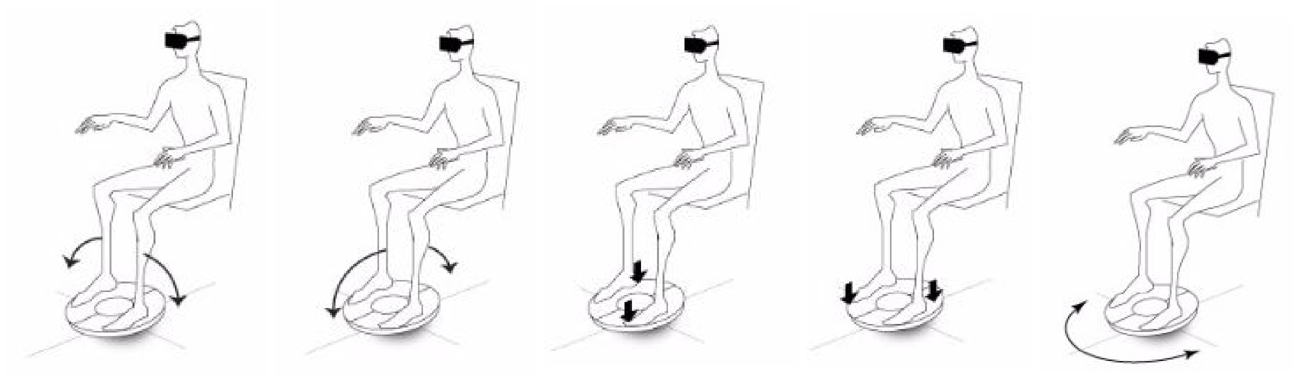


Before to start your development, you should not forget that the 3dRudder propose to you :

4 Axes - Progressive - Combined - Free Hand

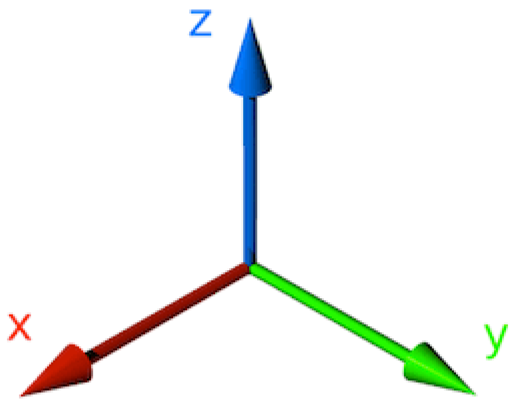
2 Axis definitions

The physical actions on the 3dRudder are converted from angle to move or rotation on 3D environment.
The physical actions are :

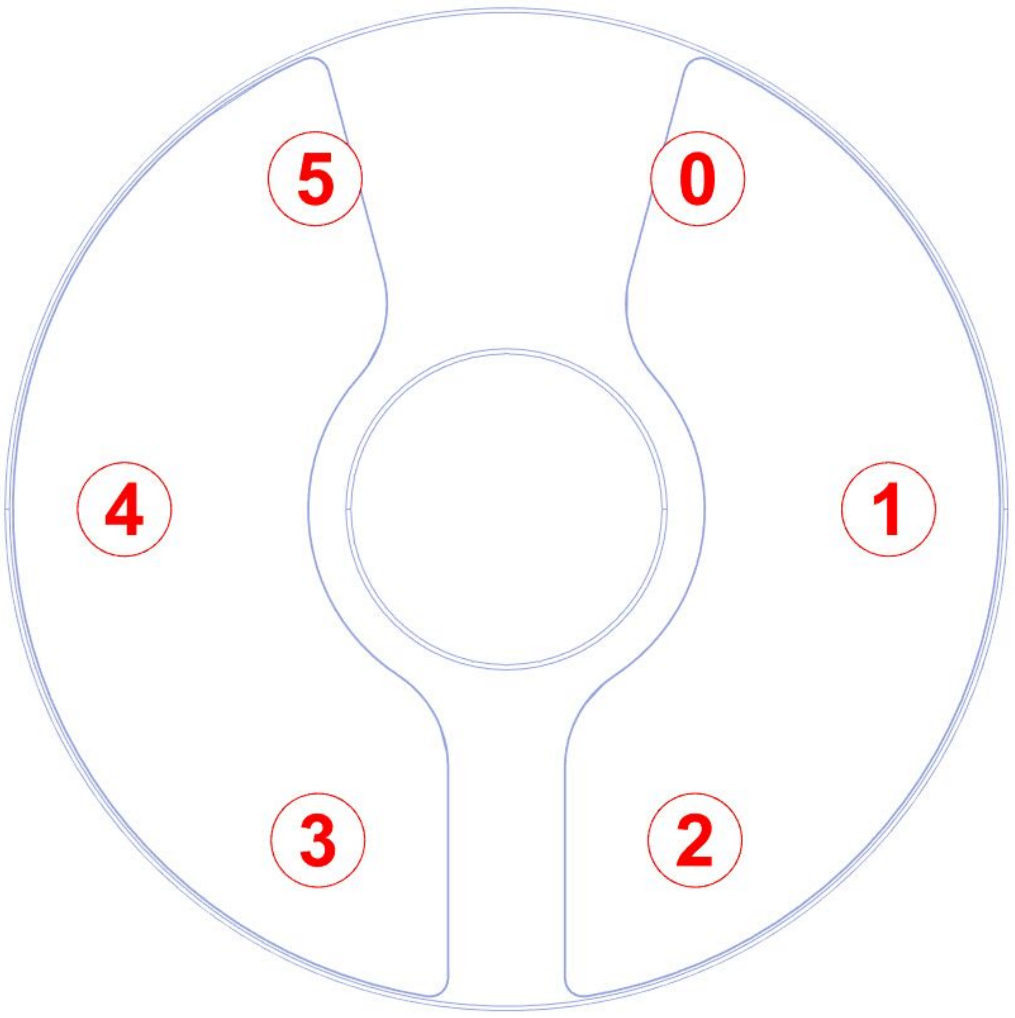


Roll **Pitch** **Up** **Down** **Yaw**

This is the 3D axis définition used by the 3dRudder to move or rotate on the 3D world :



The 3dRudder have 6 pressures sensors.



3 Module Organization

This module is based on two files :

- ns3DRudder.py
- _ns3DRudder.pyd

4 Module Information

- Available in 32 and 64 bits for Python 3.5.2
- Based on the 3dRudder C++ SDK

5 Module Usage

```python
from  ns3DRudder import *

sdk=GetSDK()

sdk.Init()
```

6 SDK Reference

All the SDK is defined in the class `ns3DRudder.CSdk` With this SDK it's possible to manage up to four 3dRudder `_3DRUDDER_SDK_MAX_DEVICE` define the maximum number of port.

6.1 Get the sdk version

`ns3DRudder.CSdk.GetSDKVersion()`

Return the SDK version of the library. The version is a fixed point unsigned short in hexadecimal: 0x0040 mean version 0.4.

6.2 Get the number of connected 3DRudder

ns3DRudder.CSdk.GetNumberOfConnectedDevice()

Return the number of 3DRudder currently connected to the computer.

6.3 Check if a 3DRudder is connected to the port

`ns3DRudder.CSdk.IsDeviceConnected(nPortNumber)`

Return true if a 3DRudder is connected to the `nPortNumber` port.

6.4 Get the Firmware version of a 3dRudder

ns3DRudder.CSdk.GetVersion(nPortNumber)

Return version number of the firmware of the 3DRudder connected to the `nPortNumber` port. The version is a fixed point unsigned short in hexadecimal: 0x1152 mean version 1.1.5.2

Return 0xFFFF in case of error.

6.5 Play a sound on a 3DRudder

`ns3DRudder.CSdk.PlaySnd(nPortNumber,nFrequency,nDuration)`

It's possible to play a sound on a 3DRudder connected to the `nPortNumber` port.

`nFrequency` define the frequency of the sound in Hz (440 is a A).

`nDuration` define the duration of the sound in ms.

6.6 Play a sequence of sound on a 3dRudder

it’s possible to play a sequence of 12 tones on a 3dRudder with a firmware version superior or equal to 1.3.6.2

6.6.1 Using a memory array to define the Tones

`ErrorCode ns3DRudder.CSdk.PlaySndEx(nPortNumber,nSize,pTones,bWait=true ) const`

Play a sequence of sound on a 3dRudder connected to the `nPortNumber` port defined by `pTones` `array` with the size `nSize.`

`pTones , a array of Tone`

`bWait=true` makes the method wait until the end of the played Tones.

`ErrorCode` is the possible error code returned by this method.

The **Tone** class is this one :

```
class Tone
{
public:
        m_nFrequency
        m_nDurationOfTone
        m_nPauseAfterTone
};
```

with :

- m_nFrequency is the Frequency of the sound to be played.
- m_nDurationOfTone is the duration of the played tone
- m_nPauseAfterTone is the silence before playing the next tones

6.6.2 Using a string to define the Tones

`ErrorCode ns3DRudder.CSdk.PlaySndEx(nPortNumber,sTones,bWait=true ) const`

Play a sequence of sound on a 3dRudder connected to the `nPortNumber` port defined by the string `sTones.`

`bWait=true` make the method wait until the end of the played Tones.

`ErrorCode` is the possible error code returned by this method.

The String need to be written like this :

Note OctaveNumber (Duration of Tone, Pause After Tone)

So to play C of the octave 5 with a Duration of 200 and a Pause of 250 you can write like this : “C5(200,250)”. The sequence can be write in the same string like this : “C5(200,250)D#5(500,200)E5(300,100)”

6.7 Freeze/Unfreeze the device

```
ErrorCode ns3DRudder.CSdk.SetFreeze(nPortNumber, bEnable)
```

It may be useful to temporarily deactivate and reactivate the 3dRudder connected to `nPortNumber` without necessarily removing and replacing the feet. This makes it possible, for example, to freeze the displacement in the phases when they are more required in the 3D universe, without risk of drifting, and to avoiding freezing the user in his initial neutral position, and thus to relocate the device or move the legs for relax.
In Freeze mode, the values returned by the 3dRudder are identical to those returned when the device waits for the 2nd foot: the outputs are set to 0.
During an "unfreeze":

- The device switches directly to the "InUse" mode, without going through the required immobility step required in standard mode. This makes it possible to freeze the displacements and to restore them without latency, for a more fluid operation.
- The user offsets are recalculated when unfreezing: thus, during the freeze, the user can change his rest position.

As a summary, the freeze / unfreeze function allows you to reposition yourself without creating unintentional movements in the game.
`bEnable` must be set to 0 to unfreeze, and to 1 to freeze.
`ErrorCode` is the possible error code returned by this method.

6.8 Hide the device

```
ErrorCode ns3DRudder.CSdk.HideSystemDevice(nPortNumber,bHide)
```

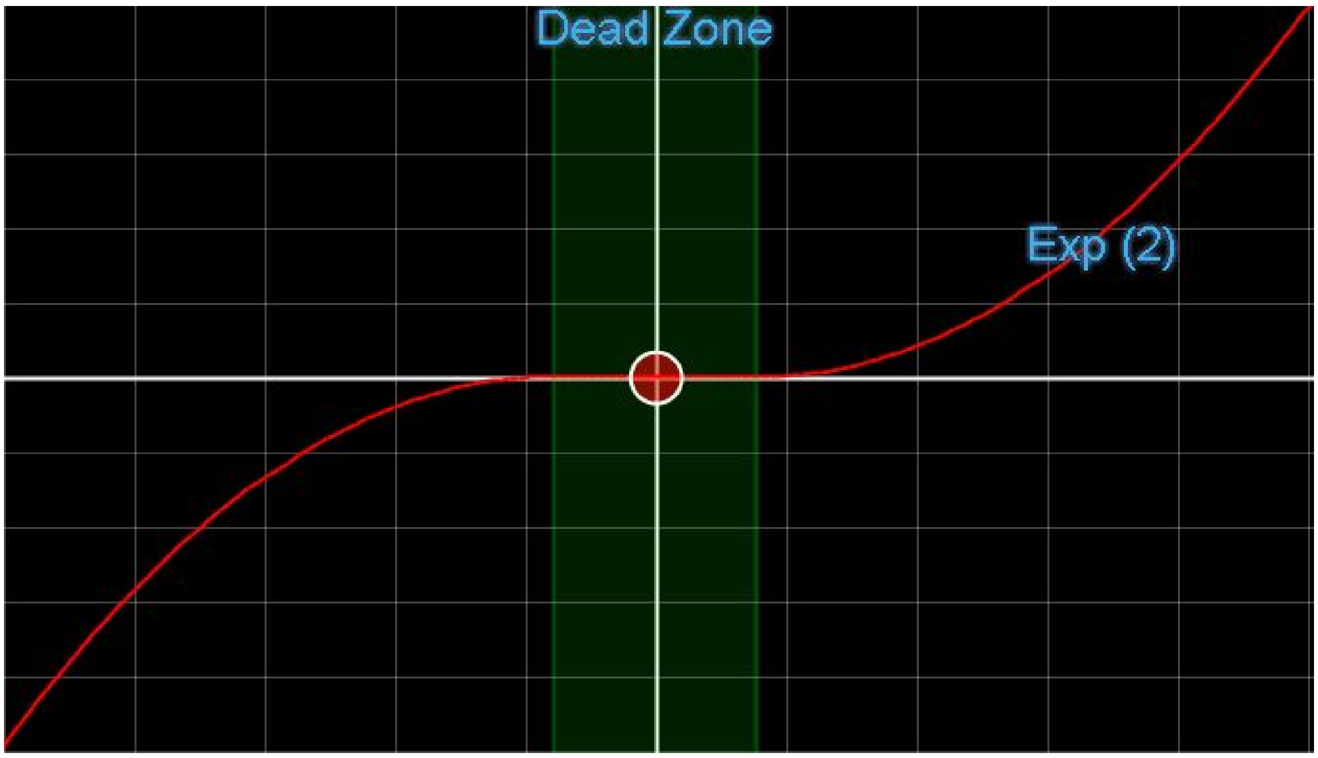
By default the 3dRudder is seen by the system as a Directinput device, a mouse or a keyboard (this can be changed in the dashboard).
The function HideSystemDevice allows to hide the 3dRudder from the system, so your game will not see it as a DirectInput device.
Please think to put it back in standard mode when you exit your game !
`ErrorCode` is the possible error code returned by this method.

6.9 Test if the device is hidden

```
ns3DRudder.CSdk.IsSystemDeviceHidden(nPortNumber)
```

Check if the device connected to `nPortNumber` is hidden

6.10 Curves



Note : before release 0.6 of the SDK, this functionality wasn't enabled.
The SDK manages the moving curves to help you to do the gameplay of your game. You have 4 parameters by curve but you will generally use mainly 2 as the others are the limits in and out.
The parameters are :

- **Dead Zone:** zone where the input has no impact on the output.
- **Exp:** exponent of the curve, Exponent 1 is linear, Exponent 2 is a square curve, etc.
- **xSat:** input limit, it's generally linked to the physical limit of the 3dRudder (for the Roll/X Axis and the Pitch/Y Axis)   or of the human body (for the Yaw/Z Rotation).
- **yMax:** output limit, as we work in normalized values, this value is generally fixed to 1.0

There is one curve for each axis, defined in `CurveType` :

`CurveXAxis` or `CurveRoll` X Axis curve
`CurveYAxis` or `CurvePitch` Y Axis curve
`CurveZAxis` or `CurveUpDown` Z Axis Curve
`CurveZRotation` or `CurveYaw` Z Rotation Curve

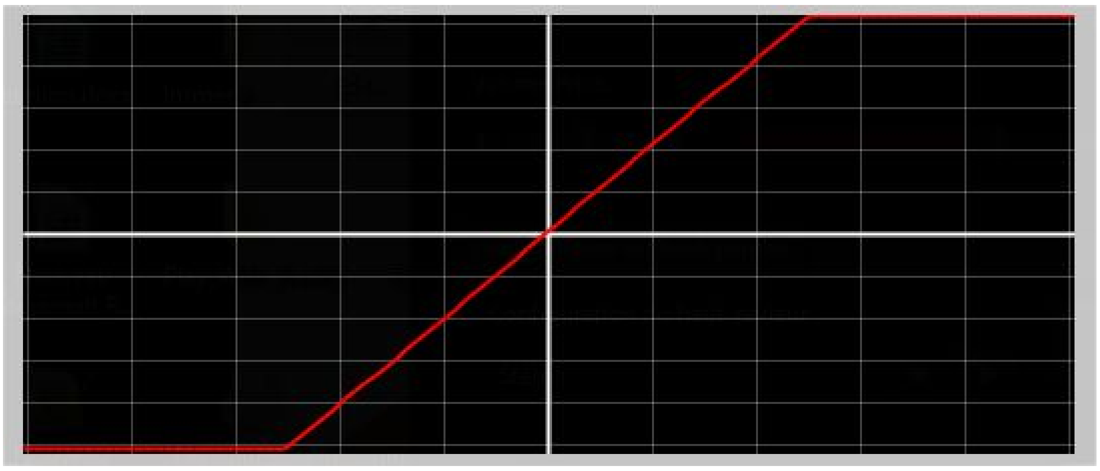
6.10.1 Linear And Factory Curves

This is an example to apply a linear or Factory curves.

```
#Init SDk 3dRudder

sdk=GetSDK()

sdk.Init()

#create Curves

curves=CurveArray()

curves.InitLinear() or Curves.InitFactory()

#apply Curves

sdk.GetAxis(0,ValueWithCurve,axis,curves)
```

<u>Linear Curve :</u>

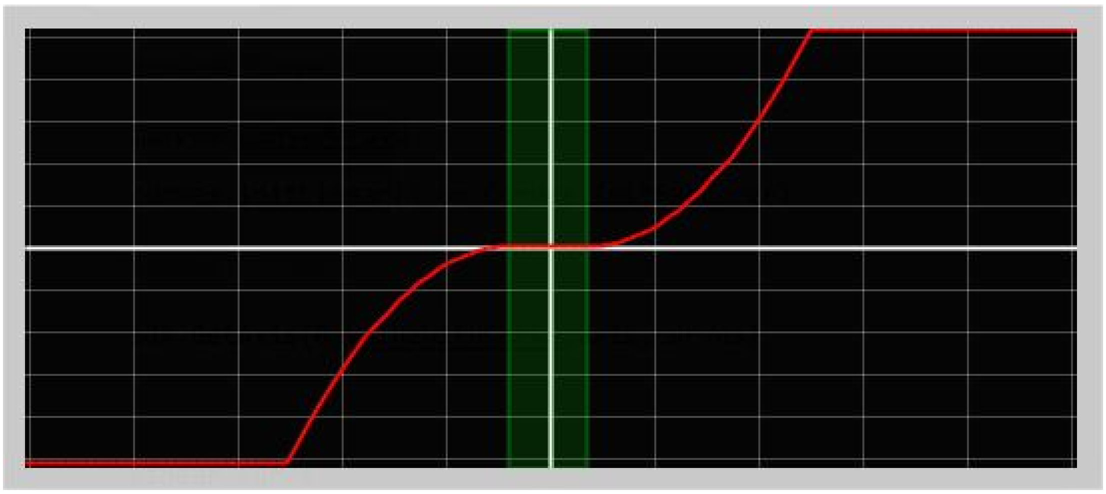
Example Yaw



Axe	DeadZone	XSat	YMax	Exp.
Yaw	0.0	1.0	1.0	1.0
Pitch	0.0	1.0	1.0	1.0
roll	0.0	1.0	1.0	1.0
UpDown	0.0	1.0	1.0	1.0

Factory Curve:

Example Yaw



Axe	DeadZone	XSat	YMax	Exp.
Yaw	3.0/25.0	20.0/25.0	1.0	2.0
Pitch	2.0/18.0	14.0/18.0	1.0	2.0
roll	2.0/18.0	12.0/18.0	1.0	2.0
UpDown	0.08	0.6	1.0	4.0

6.10.2 CurveArray()

```
By default, the CurveArray() is initialized with the FactoryCurves.
```

6.11 Get the 3DRudder Values

status `ns3DRudder.GetStatus(nPortNumber)`
status give the current status of the 3DRudder.

This status could have the values :

`1:` Puts the 3dRudder on the floor, curved side below, without putting your feet on the device. The user waits for 2 seconds for the 3DRudder to boot up until 3 short beeps are heard.
`2:` The 3dRudder initialize for about 2 seconds. Once done a long beep will be heard from the device. The 3DRudder is then operational.
`3:` Put your first feet on the 3dRudder.
`4:` Put your second Foot on the 3dRudder.
`5:` The user must wait still for half a second for calibration until a last short beep is heard from the device. The 3DRudder is ready to be used.
`6:` The 3dRudder is in use.
`7:` The 3dRudder is in use and is fully operational with all the features enabled.

6.12 Read the User Offset Value

`ErrorCode ns3DRudder.CSdk.GetUserOffset(nPortNumber,pAxis)`
This function reads the User Offset Value, i.e. the value (saved in `Axis`) of the yaw, pitch, roll and updown when the user is in its neutral position : those values could be used to calculate the Non Symmetrical Pitch value for instance.

6.13 Read Force Sensor Value

`ns3DRudder.CSdk.GetSensor(nPortNumber,nIndex)`
This function reads the values of the 6 force sensors indexed by `nIndex` of the 3dRudder connected on `nPortNumber`. The unit of 16 bits returned value is given in grams.

5.3.5 Get Axis Value
`ErrorCode ns3DRudder.CSdk.GetAxis(nPortNumber,nMode,pAxispCurve)`

This function reads the values of the `Axis,` with the current `ModeAxis` with the optional usage of the curves defined by `CurveArray` for the 3dRudder Connected to `nPortNumber.` The values are only valid if the status is `InUse` or `ExtendedMode`.
`ErrorCode` is the possible error code returned by this method.
The class `Axis` contains the value of the Axis :

you can get the X Axis with `GetXAxis()` or `GetPhysicalRoll()`
you can get the Y Axis with `GetYAxis()` or `GetPhysicalPitch()`
you can get the Z Axis with `GetZAxis()` or `GetUpDown()`
you can get the Z Rotation with `GetZRotation()` or `GetPhysicalYaw()`

The class `CurveArray` defines the setting of curve of each axis.

WARNING : since SDK version 0.7, the input values of the curves are normalized
This means that the x values should be in the range -1/1, corresponding to the following full scales of angles :

- yaw full scale : -25/+25 degrees, which is the maximum acceptable yaw angle for long-lasting play
- pitch full scale : -18/+18 degrees, which is the maximum reachable pitch angle, due to 3dRudder shape
- roll full scale : -18/+18 degrees, which is the maximum reachable pitch angle, due to 3dRudder shape
- updown full scale : -1/1 (unchanged, no unit)

`ModeAxis` defines the current mode to get the value :
In standard use, we strongly recommend to use on of the 2 ModeAxis :

- `NormalizedValueNonSymmetricalPitch`
- `ValueWithCurveNonSymmetricalPitch`

Which allows the user to reach the maximum value for backward movement whatever it's initial position.

`UserRefAngle:`

Returns 4 values, depending on the status of the 3dRudder:

`If status is InUse or ExtendedMode :`

- yaw : angle in degrees related to neutral user position (i.e. feet position at init)
- pitch : angle in degrees related to neutral user position (i.e. feet position at init)
- roll : angle in degrees related to neutral user position (i.e. feet position at init)
- updown : raw up/down value between -1 and 1

`In all other status:`

- yaw : heading angle in degrees related to magnetic North
- pitch : value in degrees related to vertical (Earth gravity)
- roll : value in degrees related to vertical (Earth gravity)
- updown : raw up/down value between -1 and 1

This mode doesn't use the curves

`NormalizedValue:`

This function returns normalized values of each 4 axis between -1 and 1

It returns 4 values, depending on the status of the 3dRudder:

`If status is InUse or ExtendedMode :`

- yaw : heading value between -1 and 1, related to neutral user position (i.e. feet position at init).
  As physical Full Scale is 25 degrees, the returned value is yaw UserRefAngle/25

- pitch : pitch value between -1 and 1, related to neutral user position (i.e. feet position at init)
  As physical Full Scale is 18 degrees, the returned value is yaw UserRefAngle/18

- roll : roll value between -1 and 1, related to neutral user position (i.e. feet position at init)
  As physical Full Scale is 18 degrees, the returned value is yaw UserRefAngle/18

- updown : raw up/down value between -1 and 1

`In all other status:`

- Returns 0

This mode doesn't use the curves

`NormalizedValueNonSymmetricalPitch:`

With this ModeAxis value, the returned values are the same as in `NormalizedValue`, except for the pitch, for which the value is magnified, depending on the initial user position, to ensure to be able to reach the full scale. This is especially usefull when the user has a significant initial pitch to the rear, which is a standard position.

In this case, as the 18° angle (in reference to the user initial position) that leads to full scale in `NormalizedValue` mode cannot be reached, the value is magnified so that the full scale is reached when the pitch reaches 18° in reference to the vertical, which is the maximum value that the shape of the 3dRudder allows), without changing the neutral position.
In other words, the pitch value is magnified in the direction where the available angle is the smaller.

The calculation of the NonSymetricalPitch uses the unsymmetrical offset of the user calculated in `InUse` and `ExtendedMode`.

This mode doesn't use the curves.

With this ModeAxis value, the returned values are the same as in `NormalizedValue`, except for the pitch, for which the value is magnified, depending on the initial user position, to ensure to be able to reach the full scale. This is especially usefull when the user has a significant initial pitch to the rear, which is a standard position.

In this case, as the 18° angle (in reference to the user initial position) that leads to full scale in `NormalizedValue` mode cannot be reached, the value is magnified so that the full scale is reached when the pitch reaches 18° in reference to the vertical, which is the maximum value that the shape of the 3dRudder allows), without changing the neutral position.
In other words, the pitch value is magnified in the direction where the available angle is the smaller.

The calculation of the NonSymetricalPitch uses the unsymmetrical offset of the user calculated in `InUse` and `ExtendedMode`.

This mode doesn't use the curves.

`ValueWithCurve:`

returns the value of the axis, using the curves.
The input value of the curve is the `NormalizedValue`, the output of the function is the corresponding output of the curve, for each axis.
The curve should have an input range of -1/+1, and can include deadzone and progressivity.

`ValueWithCurveNonSymmetricalPitch:`

returns the value of the axis, using the curves.
The input value of the curve is the `NormalizedValueNonSymmetricalPitch`, the output of the function is the corresponding output of the curve, for each axis.
The curve should have an input range of -1/+1, and can include deadzone and progressivity.

6.14 Custom Curve

It’s possible to define your own custom curve by doing a derivation of the method :

```
class MyCurve(Curve):

        def __init__(self,fDeadZone,fxSat,fyMax,fExp):
                Curve.__init__(self,fDeadZone,fxSat,fyMax,fExp)

        def CalcCurveValue(self,fValue):
                #.../…
                return fRetValue
```

of the class `Curve.`

The methode is right only with the `ModeAxis ValueWithCurve`

6.15 Events

`void ns3DRudder.CSdk.SetEvent(pEvent) const`
For version 0.6 and further of the SDK, it’s possible to get events. Currently the SDK manages two events, one for the connection and the other one for the disconnection.
to use it, you should create a class derived from `IEvent` and define two method from the virtual one :

```
                class CEvent(IEvent):

                        def __Init__(self):
                                IEvent.__Init(self)
                        def OnConnect(self,nDeviceNumber):
                        #.../…
                        def OnDisconnect(self,nDeviceNumber):
                        #.../…
```

Warning: Those events are called from another thread !

6.16 Error Code

`ErrorCode` define the error code used by the SDK:

`Success:` No error
`NotConnected:` The 3dRudder is not connected.
`Fail:` Fail to execute the method.
`IncorrectCommand:` Incorrect command.
`Timeout:` Communication with the 3dRudder timeout.
`WrongSignature:` Wrong signature of the version of the Firmware.
`NotReady:` The data you try to read is not ready.

6.17 Get the text of the error

```
GetErrorText( nError)
```

Translates the error code to human-readable value.

6.18 Description of the Samples

sample_01.py : This sample presents the basic Function of the SDK.
sample_02.py : This sample presents the Custom Curves.
sample_03.py : This sample presents the Event processing.
sample_04.py : First sample presents the Tone Function.
sample_05.py : Second sample presents the Tone Function.

For all questions contact us :

- web site : http://www.3drudder.com/download/
 http://www.3drudder.com/developers/
- github : https://github.com/3DRudder
- mail : support@3drudder.com

And follow us on :

- facebook : https://www.facebook.com/3drudder
- twitter : https://twitter.com/3DRudder
- youtube : https://www.youtube.com/channel/UCq5xGN4UsDN1VO6ii9q05uw
- google+ : https://plus.google.com/106907277277246174396
- linkedin : https://www.linkedin.com/company/3drudder