3dRudder

3dRudder plugin unity

3dRudder

03/31/2017

Version 0.0.0.9 for Unity

Warning this version of the SDK work only with the firmware 1.3.x.x and later !

If you have a old 3dRudder version with the firmware 1.2.x.x or older, please contact us to get the software to do the update. support-dev@3drudder.com

3dRudder Plugin

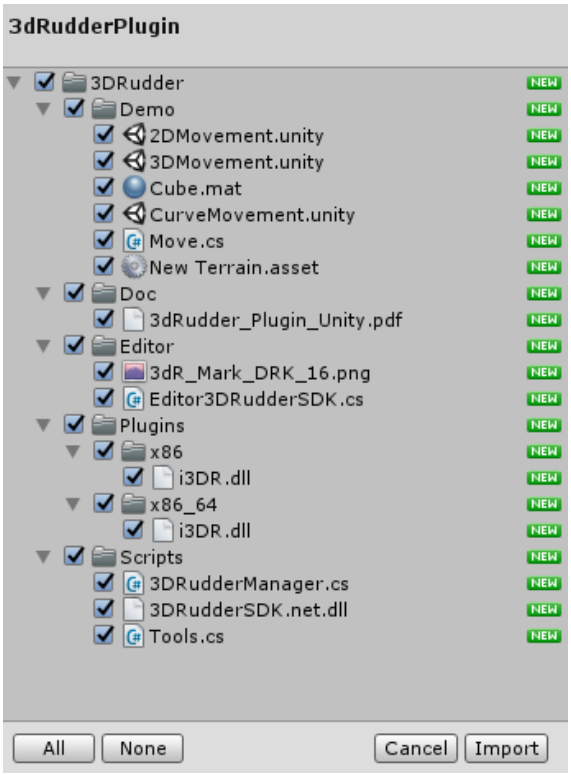
VERSION 0.0.0.9 FOR WINDOWS

Requires

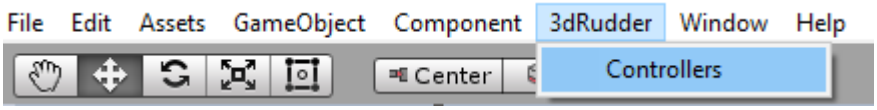
- Unity (free or pro) version 5 or higher (https://unity3d.com/)
- 3dRudder controller (http://www.3drudder.com/)
- 3dRudder SDK 0.0.7.5 already included in this package (https://github.com/3DRudder/3DRudderSDK)

Quick start

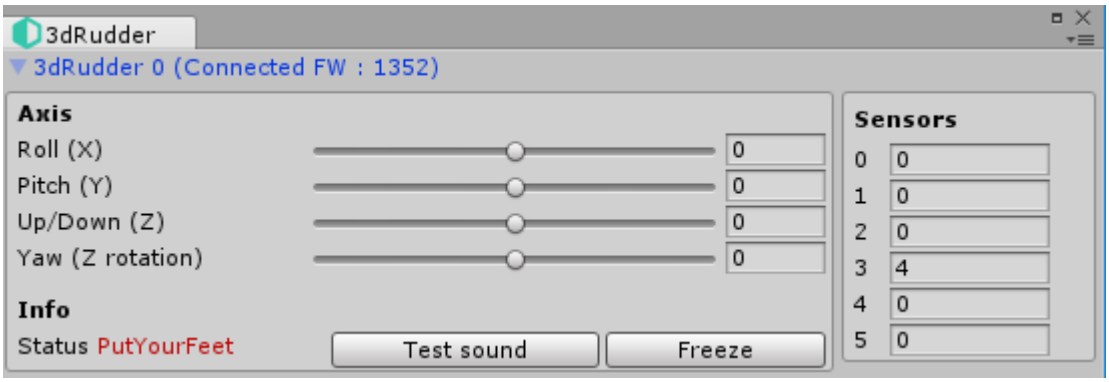
1. Import unity package. (Assets->*Import Package -> Custom Package : 3dRudder.unitypackage*)



2. Next, there are a new menu in the top toolbar. (*3dRudder -> Controllers*)

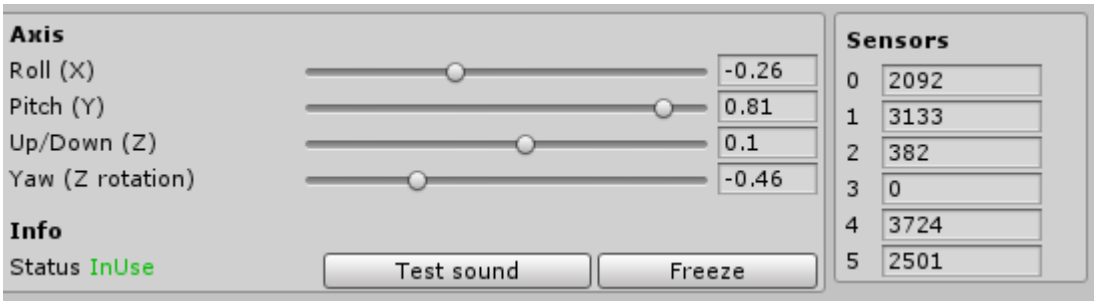


The informations are in real-time for all 3dRudder controllers.



Editor

If the controller is connected and it's using, the editor window should be like this.



The controller manage 4 **axis** :

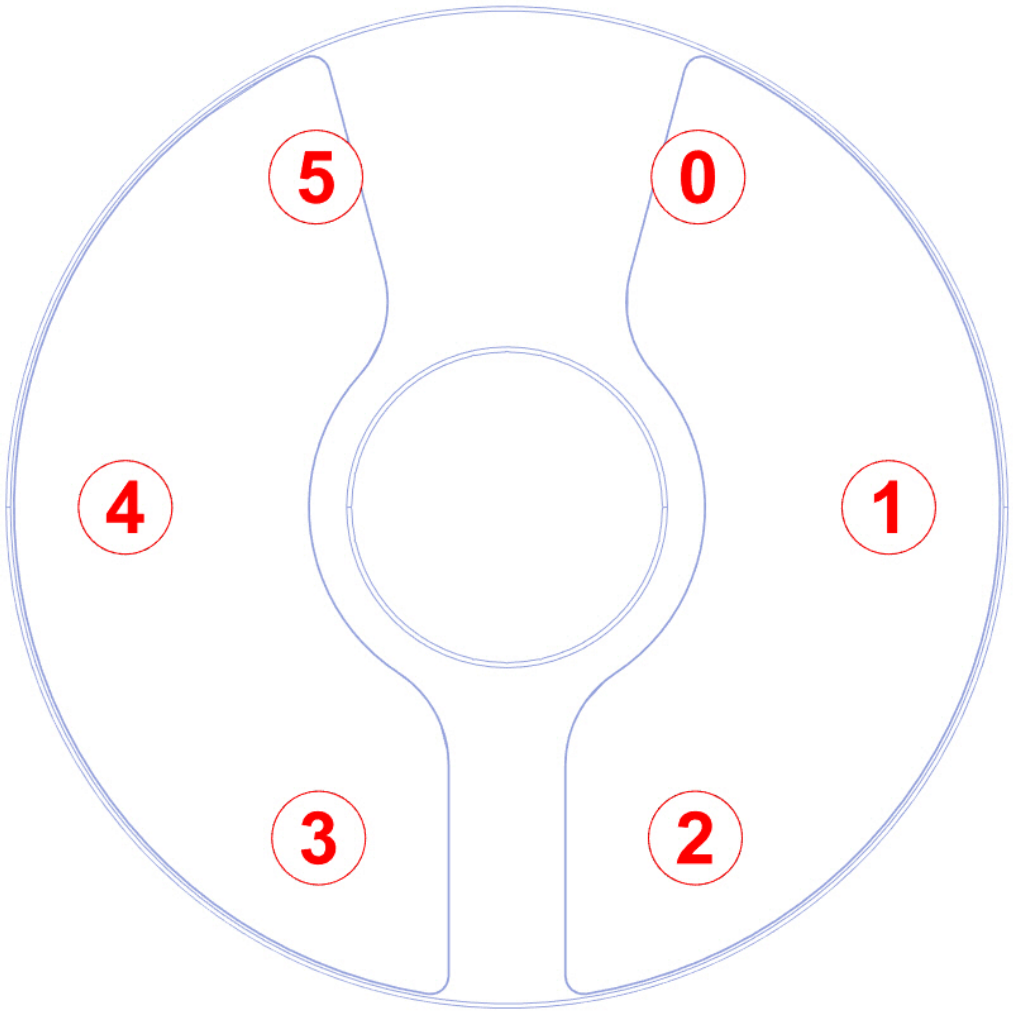
- X axis : **Roll** (Lateral Left/Right)

- Y Axis : **Pitch** (Forward/Backward)

- Z Axis : **Up/Down**

- Z rotation Axis : **Yaw** (Orbit/Rotation)

The are 6 **sensors** :



o

The **axis** are normalized, and the **sensors** are in Newton (mass)

The **status** give the current status of controller :

`NoFootStayStill`: Puts the 3dRudder on the floor, curved side below, without putting your feet on the device. The user waits for approx. 5 seconds for the 3dRudder to boot up until 3 short beeps are heard.
`Initialisation`: The 3dRudder initialize for about 2 seconds. Once done a long beep will be heard from the device. The 3dRudder is then operational.
`PutYourFeet`: Put your first feet on the 3dRudder.
`PutSecondFoot`: Put your second Foot on the 3dRudder.
`StayStill`: 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.
`InUse`: The 3dRudder is in use.
`ExtendedMode:` The 3dRudder is in use and is fully operational with all the features enabled.

s3DRudderManager

It's a singleton, you can get like this : `s3DrudderManager manager = s3DRudderManager.Instance`

The **`Unity3DRudder`**`.s3DRuddermanager` provides this methods :

1. **`void Init()`**
   Init the SDK

2. **`Status GetState(uint portNumber)`**
   This method return the state of the 3dRudder connected to the **`portNumber`** port.

3. **`void PlaySnd(uint portNumber,ushort frequencyHz,ushort durationMillisecond)`**
   It's possible to play a sound on a 3dRudder connected to the **`portNumber`** port.
   `frequencyHz` define the frequency of the sound in Hz (440 is a A).
   `durationMillisecond` define the duration of the sound in millisecond.

4. **`void SetFreeze(uint portNumber, bool enable)`**
   Freeze or unfreeze the 3dRudder

5. **`ushort GetSDKVersion()`**
   Return the SDK version of the library.

6. **`int GetNumberOfConnectedDevice()`**
   Return the number of 3dRudder currently connected to the computer.

7. **`bool IsDeviceConnected(uint portNumber)`**
   Return true if a 3dRudder is connected to the `portNumber` port.

8. **`ushort GetVersion(uint portNumber)`**
   Return version number of the firmware of the 3dRudder connected to the `portNumber` port. The version is a fixed point unsigned short in hexadecimal: 0x1152 mean version 1.1.5.2

9. **`void HideSystemDevice(uint portNumber,bool bHide)`**

   Set or not the device as a joystick

   By default the 3dRudder is seen by the system as a Directinput device, a mouse or a keyboard (this can be changed thanks to 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 !

10. **`Axis GetUserOffset(uint portNumber)`**
    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.

11. `float GetAxis(uint portNumber, ModeAxis mode, Axis pAxis, CurveArray pCurve)`
    Returns the axis X,Y,Z and Rz (Rotation Z)

    The values are only valid if the status is `InUse` or `ExtendedMode`.

    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)

12. `float GetSensor(uint portNumber, uint index)`
    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.

13. `void ShutDown()`
    ShutDown the SDK

It's possible to be notify when a 3dRudder is connected or disconnected with property Events :

```
s3DRudderManager.Instance.Events.OnConnectEvent += (portNumber) => Debug.LogFormat("3dRudder {0} connected", portNumber);
```

Rudder

The `Unity3DRudder.Rudder` provides this methods :

- `Status GetStatus()`
  Returns the status of 3Drudder (see status page 5)

- `bool IsDeviceConnected()`
  Returns if rudder is connected

- `uint GetVersion()`
  Same manager

- `void HideSystemDevice(bool hide)`
  Same manager

- `bool IsSystemDeviceHidden()`
  Returns if the rudder is show as joystick or hide

- `void PLaySnd(ushort frequencyHz,ushort durationMillisecond)`
  Same manager

- `void SetFreeze(bool enable)`
  Same manager

- `Axis GetUserOffset()`
  Same manager

- `Vector3 GetAxis3D(ns3DRudder.Axis pAxis = null)`
  Returns axis in unity's format Vector3

- `float  GetAxis(ModeAxis mode)`
  Returns the axis X,Y,Z

- `float  GetAxisWithCurve(ModeAxis mode, Curve curve)`
  Returns the axis X,Y,Z computed with the curve

- `float  GetSensor(uint index)`
  Returns the value of sensor

Curve

You can override the default curves for all axis of 3dRudder.

1. Create `CurveArray` object
2. Create `CurveRudder` (2 constructor) for each axis

   The curves are normalized in input and output [-1, 1]

   a. `CurveRudder(float fDeadZone, float fxSat, float fyMax, float fExp)` **basic curve**

      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

   b. `CurveRudder(AnimationCurve curve, bool useUnityCurve = true)` **unity curve**

3. Bind the `CurveRudder` in the `CurveArray` with the axis:

```
curvearray.SetCurve(ns3DRudder.CurveType.CurveXAxis, curverudder);
```

4. Get axis value with `ModeAxis.ValueWithCurve` or `ModeAxis.ValueWithCurveNonSymmetricalPitch`:

   `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

```
axis = rudder.GetAxisWithCurve(ModeAxis, curvearray);
```

`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.**

Freeze/UnFreeze The Device

```
void SetFreeze(uint portNumber, bool enable)
```

It may be useful to temporarily deactivate and reactivate the 3dRudder connected to **`portNumber`** 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.

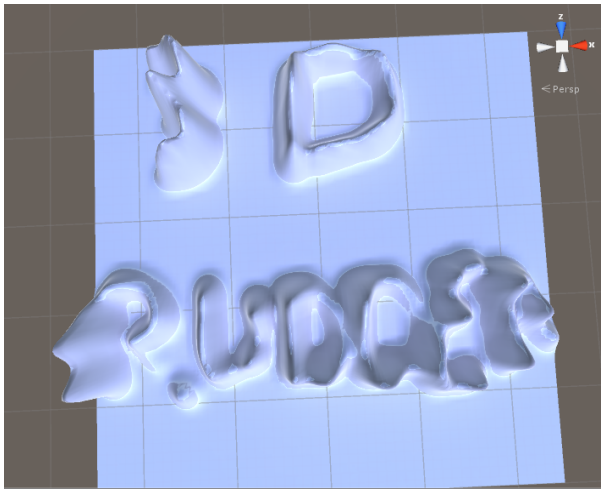
During an "unfreeze":

- 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.

Sample



There are 3 demo in package :

- 2DMovement : how to move and rotate a simple cube.
- 3DMovement : how to move in 3D environnement.
- CurveMovement: how to move with Curve.

3 demo use the same script (Move.cs).

The difference are the parameters under Inpector of Unity.

Example : For the use of the responsive curves, under the "3DMovement" , you can change the Move.cs as the following code.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void Start()
{
    cube = transform;
    translation = transform.position;
    rotation = transform.rotation;
    // Mode Curve
    curves = new ns3DRudder.CurveArray();  // Use the factory curve
    /* if (UseCurve)
    {
      ModeAxis = ns3DRudder.ModeAxis.ValueWithCurve;
      // SET ANIMATION CURVE
      Roll = new CurveRudder(RollCurve);
      Pitch = new CurveRudder(PitchCurve);
      UpDown = new CurveRudder(UpDownCurve);
      Yaw = new CurveRudder(YawCurve);
      // SET NEW RUDDER CURVES
      curves.SetCurve(ns3DRudder.CurveType.CurveXAxis, Roll);
      curves.SetCurve(ns3DRudder.CurveType.CurveYAxis, Pitch);
      curves.SetCurve(ns3DRudder.CurveType.CurveZAxis, UpDown);
      curves.SetCurve(ns3DRudder.CurveType.CurveZRotation, Yaw);
    } */
 }
```

```
void GetAxis()
{
   // Get state of Controller with port number : 0
   rudder = s3DRudderManager.Instance.GetRudder(IndexRudder);
   if (UseCurve)
    // TO DO  axis = rudder.GetAxisWithCurve(ModeAxis, curves);
      axis = rudder.GetAxisWithCurve(ns3DRudder.ModeAxis.ValueWithCurve, curves);
   else
      axis = rudder.GetAxis(ModeAxis);

         // Get the direction of Controller and multiply by deltatime and speed
    if (CanMove)
   {
      if (Move3D)
         translation = Vector3.Scale(rudder.GetAxis3D(axis), SpeedTranslation * Time.deltaTime);
      else
      {
         translation.x = axis.GetXAxis() * SpeedTranslation.x * Time.deltaTime;
         translation.z = axis.GetYAxis() * SpeedTranslation.z * Time.deltaTime;
      }
         }
     if (CanRotate)
      rotation *= Quaternion.AngleAxis(axis.GetZRotation() * SpeedRotation * Time.deltaTime, Vector3.up);
    }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Under “Inspector” check the box “Use Curve”

IMPORTANT :

Must call the function `ShutDown()` of `s3DRudderManager` when application quits.

```
OnApplicationQuit()
{
        s3DRudderManager.Instance.ShutDown();
}
```

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
- asset store : https://www.assetstore.unity3d.com/en/#!/content/72626

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