



# hinput

v1.3.0

## Documentation

[download](#) | [install](#) | [learn](#)

*hinput is a simple gamepad manager for Unity - a [hiloqo](#) project from [henri](#)*

## Summary

The hinput package is made out of the following classes :

- Core classes :
  - **hinput** (static) : The main class from which you access the gamepads
  - **hGamepad** : Represents a gamepad
  - **hStick** : Represents a left stick, a right stick or a D-pad.
  - **hPressable** (abstract) : Represents anything that can be considered pressed and released. Five classes are derived from it :
    - **hButton** : Represents a gamepad button, a bumper or a stick click.
    - **hTrigger** : Represents a gamepad trigger.
    - **hDirection** : Represents a **hStick** direction. It is considered pressed if the **hStick** is pushed in the right direction.
    - **hAnyInput** : Represents every control of a gamepad at once.
    - **hStickPressedZone** : Represents a **hStick**, interpreted as a button. It is considered pressed if the **hStick** is pushed in any direction.
  - **hAxis** : Used to calculate the position of a **hStick**.
  - **hVibration** : Handles the vibration of a gamepad.
- Other classes :
  - **hSetup** (static) : Handles the installation of hinput.
  - **hSettings** : Handles the user parameters of hinput. Instantiated automatically at runtime, but you can create it manually to change its values.
  - **hUpdater** : Handles gamepad refresh. Instantiated automatically at runtime.
  - **hUtils** (static) : Gathers useful methods regarding operating systems, internal settings, etc.

**hAxis**, **hSetup**, **hUpdater**, **hUtils**, and **hVibration** are not mentioned in the rest of this document because they are utility classes that you don't need to interact with.

# hinput

The main static class of the hinput package, from which you can access gamepads.

## Static properties

- *gamepad* (**hGamepad** array)
  - An array of 8 gamepads, labelled 0 to 7.
  - Gamepad disconnects are handled by the driver, and as such will yield different results depending on your operating system.
- *anyGamepad* (**hGamepad**)
  - A virtual gamepad that returns the inputs of every gamepad at once.
  - To be more accurate, this gamepad returns the biggest absolute value for each input (and each axis in the case of **hSticks**). For instance :
    - If player 1 pushed their A button and player 2 pushed their B button, both the A and the B button of anyGamepad will be *pressed*.
    - If player 1 pushed their left trigger by 0.24 and player 2 pushed theirs by 0.46, the left trigger of anyGamepad will have a *position* of 0.46.
    - If player 1 positioned their right stick at (-0.21, -0.78) and player 2 has theirs at (0.47, 0.55), the right stick of anyGamepad will have a *position* of (0.47, -0.78).



# hSettings

hinput class responsible for handling settings.

You can attach it to a gameobject to expose settings. If you don't, it will automatically be instantiated at runtime when needed, with default settings.

**hSettings** calls DontDestroyOnLoad when created.

## Static properties (serialized in the editor)

- *buildAllOnStartUp* (**bool**, default : false)
    - If enabled, hinput will start tracking every control of every gamepad from startup. Otherwise, each control will only start being registered the first time you ask for it.
  - *worldCamera* (**Camera**, default : null)
    - The **Camera** on which the *worldPositionCamera* and *worldPositionCameraRaw* properties of **hStick** should be calculated. If no **Camera** is set, hinput will try to find one on your scene.
    - hinput will first try to get the gameobject tagged "MainCamera". If there isn't one, hinput will get the first gameobject on the game scene that has a **Camera** component.
    - If there is no **Camera** on the scene, hinput will return an error whenever you call a *worldPositionCamera* or *worldPositionCameraRaw* property.
  - *stickDeadZone* (**float**, range (0,1), default : 0.2)
    - The distance from the origin beyond which stick inputs start being registered (except for raw inputs).
  - *stickPressedZone* (**float**, range (0,1), default : 0.5)
    - The distance from the end of the dead zone beyond which stick inputs are considered pushed.
  - *directionAngle* (**float**, range (45,90), default : 90)
    - The size of the angle that defines a stick direction.
    - If it is higher than 45 degrees, directions like *up* and *upLeft* will overlap. Likewise, if it is lower than 90 degrees, there will be a gap between directions like *up* and *left*.
  - *triggerDeadZone* (**float**, range (0,1), default : 0.1)
    - The distance from the origin beyond which trigger inputs start being registered (except for raw inputs).
  - *triggerPressedZone* (**float**, range (0,1), default : 0.5)
-

- The distance from the end of the dead zone beyond which trigger inputs are considered pushed.
- *doublePressDuration* (**float**, range (0,2), default : 0.3)
  - The maximum duration between the start of two presses for them to be considered a double press.
- *longPressDuration* (**float**, range (0,2), default : 0.3)
  - The minimum duration of a press for it to be considered a long press.
- *vibrationDuration* (**float**, range (0,2), default : 0.5)
  - The default duration of gamepad vibration.
- *leftVibrationIntensity* (**float**, range (0,1), default : 1)
  - The default intensity of the left motor when controllers vibrate.
- *rightVibrationIntensity* (**float**, range (0,1), default : 1)
  - The default intensity of the left motor when controllers vibrate.



# hGamepad

hinput class representing a gamepad.

## Properties

- **fullName (string)**
    - The full name of a gamepad, like "Linux\_Gamepad4".
  - **index (int)**
    - The index of a gamepad in the *gamepad* array of hinput, like 3 for `hinput.gamepad[3].index`.
    - `hinput.anyGamepad.index` will return -1.
  - **type (int)**
    - The type of a gamepad, like "Xbox One For Windows"
  - **sticks (List<hStick>)**
    - The list containing a gamepad's sticks, in the following order : { leftStick, rightStick, dPad }
  - **buttons (List<hPressable>)**
    - The list containing a gamepad's buttons, in the following order : { A, B, X, Y, left bumper, right bumper, left trigger, right trigger, back, start, left stick click, right stick click, XBox button }
  - **anyInput (hAnyInput)**
    - A virtual button that returns every input of a gamepad at once.
  - **leftStick (hStick)**
    - The left stick of a gamepad.
  - **rightStick (hStick)**
    - The right stick of a gamepad.
  - **dPad (hStick)**
    - The D-pad of a gamepad.
  - **leftTrigger (hTrigger)**
    - The left trigger of a gamepad.
  - **rightTrigger (hTrigger)**
    - The right trigger of a gamepad.
  - **A (hButton)**
-

- The A button of a gamepad.
- ***B* (hButton)**
  - The B button of a gamepad.
- ***X* (hButton)**
  - The X button of a gamepad.
- ***Y* (hButton)**
  - The Y button of a gamepad.
- ***back* (hButton)**
  - The Back button of a gamepad.
- ***start* (hButton)**
  - The Start button of a gamepad.
- ***leftBumper* (hButton)**
  - The left bumper of a gamepad.
- ***rightBumper* (hButton)**
  - The right bumper of a gamepad.
- ***leftStickClick* (hButton)**
  - The left stick click of a gamepad.
- ***rightStickClick* (hButton)**
  - The right stick click of a gamepad.
- ***xBoxButton* (hButton)**
  - The XBox button of a gamepad.
  - Windows and Linux drivers can't detect the value of this button. Therefore it will be considered released at all times on these operating systems.

## Methods

- ***Vibrate*** (no arguments)
  - Vibrate a gamepad. Default duration and intensity can be tweaked in settings.
- ***Vibrate*** (argument : duration (**float**))
  - Vibrate a gamepad for duration seconds. Default intensity can be tweaked in hinput settings.
- ***Vibrate*** (arguments : leftIntensity (**float**), rightIntensity (**float**))

- Vibrate the left motor a gamepad with an intensity of leftIntensity, and the right motor with an intensity of rightIntensity. Default intensity can be tweaked in settings.
- *Vibrate* (arguments : leftIntensity (**float**), rightIntensity (**float**), duration (**float**))
  - Vibrate the left motor a gamepad with an intensity of leftIntensity, and the right motor with an intensity of rightIntensity, for duration seconds.
- *VibrateAdvanced* (arguments : leftIntensity (**float**), rightIntensity (**float**))
  - Vibrate the left motor a gamepad with an intensity of leftIntensity, and the right motor with an intensity of rightIntensity, FOREVER.
  - Don't forget to call *StopVibration* !
- *StopVibration* (argument : duration (**float**))
  - Stop all vibrations on a gamepad.



# hPressable

hinput abstract class representing anything that can be considered pressed and released. It can be an actual button, a stick click, a trigger, a stick direction...

## Implicit Cast

If no property of the **hPressable** is used, it will automatically be cast to a boolean with the value *pressed*.

For instance, `hinput.gamepad[0].A` will be interpreted as `hinput.gamepad[0].A.pressed`.

## Abstract properties (overridden by derived classes)

- *pressed* (**bool**)
  - Returns true if the input is pressed. Returns false otherwise.
- *position* (**float**)
  - Returns the current position of the input.
- *inDeadZone* (**bool**)
  - Returns true if the input is in its dead zone. Returns false otherwise.

## Properties

- *name* (**string**)
    - Returns the name of the input , like “A”, “LeftTrigger” or “DPad\_Up”.
  - *fullName* (**string**)
    - Returns the full name of the input , like “Mac\_Gamepad2\_RightStickClick”
  - *gamepadIndex* (**int**)
    - Returns the index of the gamepad this input is attached to.
  - *gamepad* (**hGamepad**)
    - Returns the gamepad this input is attached to.
  - *positionRaw* (**float**)
    - Returns the current raw position of the input, i.e. not taking the dead zone into account.
  - *released* (**bool**)
    - Returns true if the input is not *pressed*. Returns false otherwise.
  - *justPressed* (**bool**)
-



- Returns true if the input is currently *pressed* and was *released* last frame. Returns false otherwise.
  - *justReleased* (**bool**)
    - Returns true if the input is currently *released* and was *pressed* last frame. Returns false otherwise.
  - *doublePress* (**bool**)
    - Returns true if the input is currently *pressed*, and the last two presses started less than `hSettings.doublePressDuration` seconds apart. Returns false otherwise.
  - *doublePressJustPressed* (**bool**)
    - Returns true if the input is currently *justPressed*, and the last two presses started less than `hSettings.doublePressDuration` seconds apart. Returns false otherwise.
  - *doublePressJustReleased* (**bool**)
    - Returns true if the input is currently *justReleased*, and the last two presses started less than `hSettings.doublePressDuration` seconds apart. Returns false otherwise.
  - *lastPressWasDouble* (**bool**)
    - Returns true if the last two presses started less than `hSettings.doublePressDuration` seconds apart (including current press if the input is *pressed*). Returns false otherwise.
  - *longPress* (**bool**)
    - Returns true if the input is currently *pressed* and the press has lasted longer than `hSettings.longPressDuration` seconds. Returns false otherwise.
  - *longPressJustReleased* (**bool**)
    - Returns true if the input is currently *justReleased*, and the last press has lasted longer than `hSettings.longPressDuration` seconds. Returns false otherwise.
  - *lastPressWasLong* (**bool**)
    - Returns true if the last press has lasted longer than `hSettings.longPressDuration` seconds (including current press if the input is *pressed*). Returns false otherwise.
  - *pressDuration* (**float**)
    - If the input is *pressed*, returns the amount of time that has passed since it is *pressed*. Returns 0 otherwise.
-

- *releaseDuration* (**float**)
  - If the input is *released*, returns the amount of time that has passed since it is *released*. Returns 0 otherwise
- *lastPressed* (**float**)
  - Returns the date the input was last *pressed* (in seconds from the beginning of the game). Returns 0 if it hasn't been *pressed*.
- *lastPressStart* (**float**)
  - Returns the date the input was last *justPressed* (in seconds from the beginning of the game). Returns 0 if it hasn't been *pressed*.
- *lastReleased* (**float**)
  - Returns the date the input was last *released* (in seconds from the beginning of the game). Returns zero if it hasn't been *pressed*.



# hButton : hPressable

hinput class representing a physical button of the controller, such as the A button, the bumpers or the stick clicks.

Inherits **hPressable** and redefines the values of *pressed*, *position*, *positionRaw*, and *inDeadZone*.

## Properties

- ***index* (int)**
  - Returns the index of the button on its gamepad.

## Override properties

- ***positionRaw* (float)**
  - Returns 1 if the button is currently pressed. Returns 0 otherwise.
- ***position* (float)**
  - Returns 1 if the button is currently pressed. Returns 0 otherwise.
- ***pressed* (bool)**
  - Returns true if the button is currently pressed. Returns false otherwise.
- ***inDeadZone* (bool)**
  - Returns true if the input is currently released. Returns false otherwise.



# hTrigger : hPressable

hinput class representing the left or right trigger of a controller.

Inherits **hPressable** and redefines the values of *pressed*, *position*, *positionRaw*, and *inDeadZone*.

## Properties

- *index* (**int**)
  - Returns the index of the trigger on its gamepad.

## Override properties

- *positionRaw* (**float**)
  - Returns the position of the trigger, between 0 and 1. The dead zone is not taken into account.
- *position* (**float**)
  - Returns the position of the trigger, between 0 and 1.
- *pressed* (**bool**)
  - Returns true if the position of the trigger is beyond `hSettings.pressedZone`. Returns false otherwise.
- *inDeadZone* (**bool**)
  - Returns true if the position of the trigger is within `hSettings.triggerDeadZone`. Returns false otherwise.



# hDirection : hPressable

hinput class representing a given direction of a stick or D-pad, such as the up or down-left directions.

Inherits **hPressable** and redefines the values of *pressed*, *position*, *positionRaw*, and *inDeadZone*.

## Properties

- ***stickIndex* (int)**
  - Returns the index of the stick this direction is attached to (0 for a left stick, 1 for a right stick, 2 for a D-pad).
- ***stick* (hStick)**
  - Returns the stick this direction is attached to.
- ***angle* (float)**
  - Returns the value of the angle that defines this direction (In degrees : left=180, up=90, right=0, down=-90).

## Override properties

- ***positionRaw* (float)**
  - Returns the position of the stick along the direction, between -1 and 1. The dead zone is not taken into account.
- ***position* (float)**
  - Returns the position of the stick along the direction, between -1 and 1.
- ***pressed* (bool)**
  - Returns true if the stick is *inPressedZone*, and within hSettings.directionAngle degrees of *angle*. Returns false otherwise.
- ***inDeadZone* (bool)**
  - Returns true if the stick is *inDeadZone*, or beyond hSettings.directionAngle degrees of *angle*. Returns false otherwise.



# hAnyInput : hPressable

hinput class representing every input of a controller at once.

Inherits **hPressable** and redefines the values of *pressed*, *position*, *positionRaw*, and *inDeadZone*.

## Override properties

- *positionRaw* (**float**)
  - Returns the raw position of the most pushed gamepad button, between 0 and 1.
- *position* (**float**)
  - Returns the position of the most pushed gamepad button, between 0 and 1.
- *pressed* (**bool**)
  - Returns true if a gamepad button is currently pressed. Returns false otherwise.
- *inDeadZone* (**bool**)
  - Returns true if all gamepad buttons are currently in dead zone. Returns false otherwise.



# hStickPressedZone : hPressable

hinput class representing a stick or D-pad as a button. It is considered pressed if the stick is pushed in any direction.

Inherits **hPressable** and redefines the values of *pressed*, *position*, *positionRaw*, and *inDeadZone*.

## Properties

- **stickIndex (int)**
  - Returns the index of the stick this button is attached to (0 for a left stick, 1 for a right stick, 2 for a D-pad).
- **stick (hStick)**
  - Returns the stick this button is attached to.

## Override properties

- **positionRaw (float)**
  - Returns the relative distance between the current stick's raw position and the end of its pressed zone, between 0 and 1. Returns 1 if it is in its pressed zone.
- **position (float)**
  - Returns the relative distance between the current stick's position and the end of its pressed zone, between 0 and 1. Returns 1 if it is in its pressed zone.
- **pressed (bool)**
  - Returns true if the stick is *inPressedZone*. Returns false otherwise.
- **inDeadZone (bool)**
  - Returns true if the stick is *inDeadZone*. Returns false otherwise.



# hStick

hinput class representing a gamepad stick, such as the left stick, the right stick, or the D-pad.

## Implicit Cast

If no property of the **hStick** is used, it will automatically be cast to a **Vector2** with the value *position*.

For instance, `hinput.gamepad[0].leftStick` will be interpreted as `hinput.gamepad[0].leftStick.position`.

## Properties

- **name (string)**
  - Returns the name of the stick, like “LeftStick” or “DPad”.
- **fullName (string)**
  - Returns the full name of the stick, like “Mac\_Gamepad2\_RightStick”
- **gamepadIndex (int)**
  - Returns the index of the gamepad this stick is attached to.
- **gamepad (hGamepad)**
  - Returns the gamepad this stick is attached to.
- **index (int)**
  - Returns the index of the stick on its gamepad (0 for a left stick, 1 for a right stick, 2 for a D-pad).
- **up (hDirection)**
  - Returns a virtual button defined by the stick’s projected position along a direction that has a 90 degree angle with the horizontal axis.
- **down (hDirection)**
  - Returns a virtual button defined by the stick’s projected position along a direction that has a -90 degree angle with the horizontal axis.
- **left (hDirection)**
  - Returns a virtual button defined by the stick’s projected position along a direction that has a 180 degree angle with the horizontal axis.
- **right (hDirection)**





- Returns a virtual button defined by the stick's projected position along the horizontal axis.
  - *upLeft* (**hDirection**)
    - Returns a virtual button defined by the stick's projected position along a direction that has a 135 degree angle with the horizontal axis.
  - *downLeft* (**hDirection**)
    - Returns a virtual button defined by the stick's projected position along a direction that has a -135 degree angle with the horizontal axis.
  - *upRight* (**hDirection**)
    - Returns a virtual button defined by the stick's projected position along a direction that has a 45 degree angle with the horizontal axis.
  - *downRight* (**hDirection**)
    - Returns a virtual button defined by the stick's projected position along a direction that has a -45 degree angle with the horizontal axis.
  - *leftUp* (**hDirection**)
    - Returns a virtual button defined by the stick's projected position along a direction that has a 135 degree angle with the horizontal axis.
  - *leftDown* (**hDirection**)
    - Returns a virtual button defined by the stick's projected position along a direction that has a -135 degree angle with the horizontal axis.
  - *rightUp* (**hDirection**)
    - Returns a virtual button defined by the stick's projected position along a direction that has a 45 degree angle with the horizontal axis.
  - *rightDown* (**hDirection**)
    - Returns a virtual button defined by the stick's projected position along a direction that has a -45 degree angle with the horizontal axis.
  - *position* (**Vector2**)
    - Returns the coordinates of the stick.
  - *positionRaw* (**Vector2**)
    - Returns the coordinates of the stick. The dead zone is not taken into account.
  - *horizontal* (**float**)
    - Returns the x coordinate of the stick.
-

- ***horizontalRaw (float)***
    - Returns the x coordinate of the stick. The dead zone is not taken into account.
  - ***vertical (float)***
    - Returns the y coordinate of the stick.
  - ***verticalRaw (float)***
    - Returns the y coordinate of the stick. The dead zone is not taken into account.
  - ***angle (float)***
    - Returns the value of the angle between the current position of the stick and the horizontal axis (In degrees : left=180, up=90, right=0, down=-90).
  - ***angleRaw (float)***
    - Returns the value of the angle between the current position of the stick and the horizontal axis (In degrees : left=180, up=90, right=0, down=-90). The dead zone is not taken into account.
  - ***distance (float)***
    - Returns the current distance of the stick to its origin.
  - ***distanceRaw (float)***
    - Returns the current distance of the stick to its origin. The dead zone is not taken into account.
  - ***inDeadZone (bool)***
    - Returns true if the current position of the stick is within a distance of `hSettings.stickDeadZone` of its origin. Returns false otherwise.
  - ***inPressedZone (bool)***
    - Returns true if the current position of the stick is beyond a distance of `hSettings.pressedZone` of its origin. Returns false otherwise.
  - ***worldPositionCamera (Vector3)***
    - Returns the coordinates of the stick as a Vector3 facing `hSettings.worldCamera`. The stick's horizontal and vertical axes are interpreted as the camera's right and up directions.
  - ***worldPositionCameraRaw (Vector3)***
    - Returns the coordinates of the stick as a Vector3 facing `hSettings.worldCamera`. The stick's horizontal and vertical axes are interpreted as the camera's right and up directions. The dead zone is not taken into account.
  - ***worldPositionFlat (Vector3)***
-

- Returns the coordinates of the stick as a Vector3 with a y value of 0. The stick's horizontal and vertical axes are interpreted as the absolute right and forward directions.
  
- *worldPositionFlatRaw* (**Vector3**)
  - Returns the coordinates of the stick as a Vector3 with a y value of 0. The stick's horizontal and vertical axes are interpreted as the absolute right and forward directions. The dead zone is not taken into account.

