hinput : Documentation

hinput is a simple multi-OS gamepad input system for Unity.

A hiloqo (@hiloqoco) project from henri (@henriforshort).

Instruction on how to **get started** are available here : https://bit.ly/2JLXf4S

hinput

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

Static properties

- **anyGamepad** (hGamepad)
  - A virtual gamepad that returns the biggest absolute value for each control of all connected gamepads.
  - **Note :** If a given stick has equal and opposite values on different gamepads, the lowest value will be returned. For instance, if player 1 has their left stick pointing all the way left, and player 2 has theirs pointing all the way right, anyGamepad's left stick will point left.
- **gamepad** (hGamepad array)
  - An array of 8 gamepads, labelled 0 to 7.
  - The exact labelling of the gamepads depends on the order in which they were plugged in, as well as the usb slots and your os… Update coming soon.

hinputSettings

hinput class responsible for handling settings, as well as gamepad updates.

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.

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.
- **deadZone** (float, range 0 -- 1, default : 0.2)
  - The distance from the center beyond which stick and trigger inputs start being registered (except for raw inputs).
- **triggerZone** (float, range 0 -- 1, default : 0.5)
  - The distance from the end of the dead zone beyond which stick and trigger inputs are considered pushed or activated.
- **directionAngle** (float, range 45 -- 90, default : 90)
  - The size of the angle that defines a stick direction.
  - **Note :** 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).
- **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.
- **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 get the Camera 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.

hinputUpdater

hinput’s class responsible for updating gamepads.

It is automatically instantiated at runtime, or added to the gameobject with the hinputSettings component if you created one.

You don’t need to do anything about it.

hGamepad

hinput class representing a gamepad.

Properties

- **fullName** (string)
  - The full name of a gamepad, like “Linux_Gamepad4”.
  - **Note :** the number at the end of the gamepad’s name is the one used by Unity, not by hinput. It is NOT equal to (index), but to (index)+1.
- **index** (int)
  - The index of a gamepad in the (gamepad) array of hinput, like 3 for (hinput.gamepad[3].index).
  - **Note :** (hinput.anyGamepad.index) will return -1.
- **leftStick** (hStick)
  - The left stick of a gamepad.
- **rightStick** (hStick)
  - The right stick of a gamepad.
- **dPad** (hStick)
  - The D-pad of a gamepad.
- **sticks** (List<hStick>)
  - The list containing a gamepad’s sticks, in the following order : { leftStick, rightStick, dPad }
- **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.
    - **Note :** 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.

hAbstractPressable

hinput abstract class representing anything that can be pressed. It can be an actual button, a stick click, a trigger, or a stick or D-pad direction.

Implicit Cast

If no property of the hAbstractPressable is used, it will automatically be cast to a boolean with the value (pressed). For instance, (hinput.gamepad[0].A) will return (hinput.gamepad[0].A.pressed).

Abstract properties (overridden by hButton, hTrigger and hDirection)

- **pressed** (bool)
  - Returns true if the input is considered “pressed”. Returns false otherwise.
- **position** (float)
  - Returns the current position of the input (0 or 1 for a button, 0 to 1 for a trigger, and -1 to 1 for a stick direction).
- **positionRaw** (float)
  - Returns the current raw position of the input. Similar to (position) for buttons. Triggers and stick directions do not take the dead zone into account.
- **inDeadZone** (bool)
  - For a button, returns (pressed)
  - For a trigger, returns true if the trigger’s raw position is higher than (deadZone).
  - For a stick direction, returns true if the stick’s raw distance is higher than (deadZone)

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”
  - **Note :** the number at the end of the gamepad’s name is the one used by Unity, not by hinput. It is NOT equal to (index), but to (index)+1.
- **gamepadIndex** (int)
  - Returns the index of the gamepad this input is attached to.
- **gamepad** (hGamepad)
  - Returns the gamepad this input is attached to.
- **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 (hinput.doublePressDuration) seconds apart. Returns false otherwise.
- **doublePressJustPressed** (bool)
    - Returns true if the input is currently (justPressed), and the last two presses started less than (hinput.doublePressDuration) seconds apart. Returns false otherwise.
- **doublePressJustReleased** (bool)
    - Returns true if the input is currently (justReleased), and the last two presses started less than (hinput.doublePressDuration) seconds apart. Returns false otherwise.
- **lastPressWasDouble** (bool)
    - Returns true if the last two presses started less than (hinput.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 (hinput.longPressDuration) seconds. Returns false otherwise.
- **longPressJustReleased** (bool)
    - Returns true if the input is currently (justReleased), and the last press has lasted longer than (hinput.longPressDuration) seconds. Returns false otherwise.
- **lestPressWasLong** (bool)
    - Returns true if the last press has lasted longer than (hinput.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 : hAbstractPressable

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

Inherits hAbstractPressable and redefines the values of (pressed), (position), (positionRaw), and (inDeadZone).

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 not (pressed). Returns false otherwise.

hTrigger : hAbstractInput

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

Inherits hAbstractPressable and redefines the values of (pressed), (position), (positionRaw), and (inDeadZone).

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 (hinput.triggerZone). Returns false otherwise.
- **inDeadZone** (bool)
  - Returns true if if the position of the trigger is within (hinput.deadZone). Returns false otherwise.

hDirection : hAbstractPressable

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

Inherits hAbstractPressable 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 (stick) is (inTriggerZone), and within (hinput.directionAngle) degrees of (angle). Returns false otherwise.
- **inDeadZone** (bool)
  - Returns true if (stick) is (inDeadZone), or beyond (hinput.directionAngle) degrees of (angle). 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 return (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”
  - **Note :** the number at the end of the gamepad’s name is the one used by Unity, not by hinput. It is NOT equal to (index), but to (index)+1.
- **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 in the shape of a Vector2.
- **positionRaw** (Vector2)
    - Returns the coordinates of the stick in the shape of a Vector2. 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 (hinput.deadZone) of its origin. Returns false otherwise.
- **inTriggerZone** (bool)

    - Returns true if the current position of the stick is beyond a distance of (hinput.triggerZone) of its origin. Returns false otherwise.
- **worldPositionCamera** (Vector3)
    - Returns the coordinates of the stick as a Vector3 facing (hinput.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 (hinput.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.
- **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. The dead zone is not taken into account.