Console Cursors

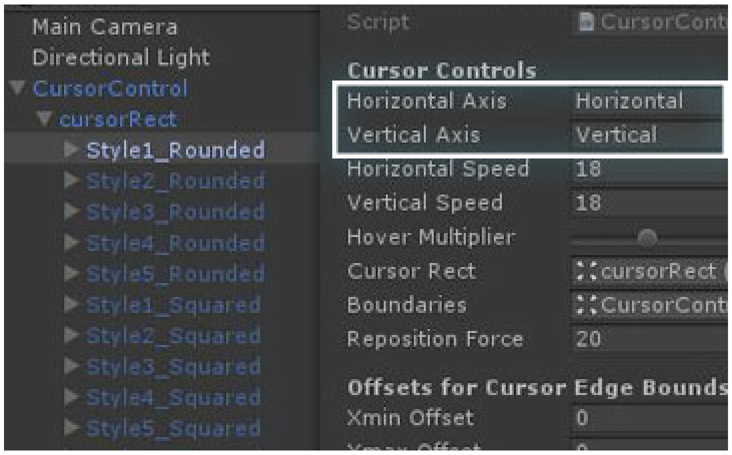
For latest documentation, visit slimui.com/cursor-console-documentation
For support, email: andrei@shulgach.com

INTRODUCTION

Console Cursors is a quick and easy out-of-the-box solution for anyone trying to achieve a modern controller cursor becoming popularized in recent AAA games. With a simple boundaries detection, 25 unique hover animations, 25 different cursor styles to choose from, 3 different button styles, and more, you can make your game have a AAA quality console/controller navigation in just a few minutes. Console Cursor can be used for UGUI Screen Space and World Space offering endless possibilities for your game or project.

1.1 SETTING UP THE CURSOR

To get started, add the **“CursorControl”** to a scene in your project. Assign the **Render Camera** variable to the camera functioning as your main camera in your scene. If you press play, the cursor is already controllable through the keyboard inputs for the **Horizontal** and **Vertical** axes as well as the 25 styles in the inspector, assuming the Input Manager is still using default settings for the axis'. For Input Manager configuration, go to section 5.1



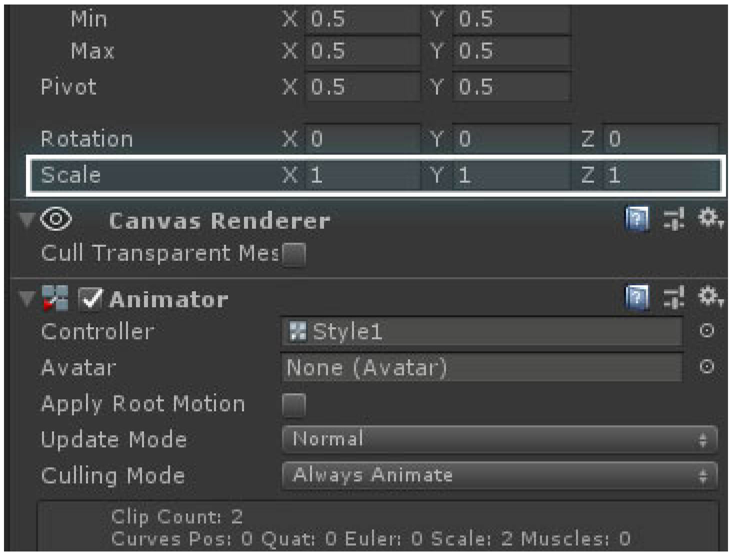
```
Main Camera                       Script                 CursorCont
Directional Light
▼ CursorControl                   Cursor Controls
  ▼ cursorRect                    Horizontal Axis        Horizontal
    ▶ Style1_Rounded              Vertical Axis          Vertical
    ▶ Style2_Rounded              Horizontal Speed       18
    ▶ Style3_Rounded              Vertical Speed         18
    ▶ Style4_Rounded              Hover Multiplier
    ▶ Style5_Rounded              Cursor Rect            cursorRect (
    ▶ Style1_Squared              Boundaries             CursorCont
    ▶ Style2_Squared              Reposition Force       20
    ▶ Style3_Squared
    ▶ Style4_Squared              Offsets for Cursor Edge Bounds
    ▶ Style5_Squared              Xmin Offset            0
                                  Ymax Offset            0
```

1.2 CUSTOMIZING THE CURSOR

For all 25 cursor styles, you can enable or disable any part that you don't want to use and the animation will still function properly. Each cursor has it's own animation and animator so modifying one will not affect the others.

1.3 CURSOR SCALE

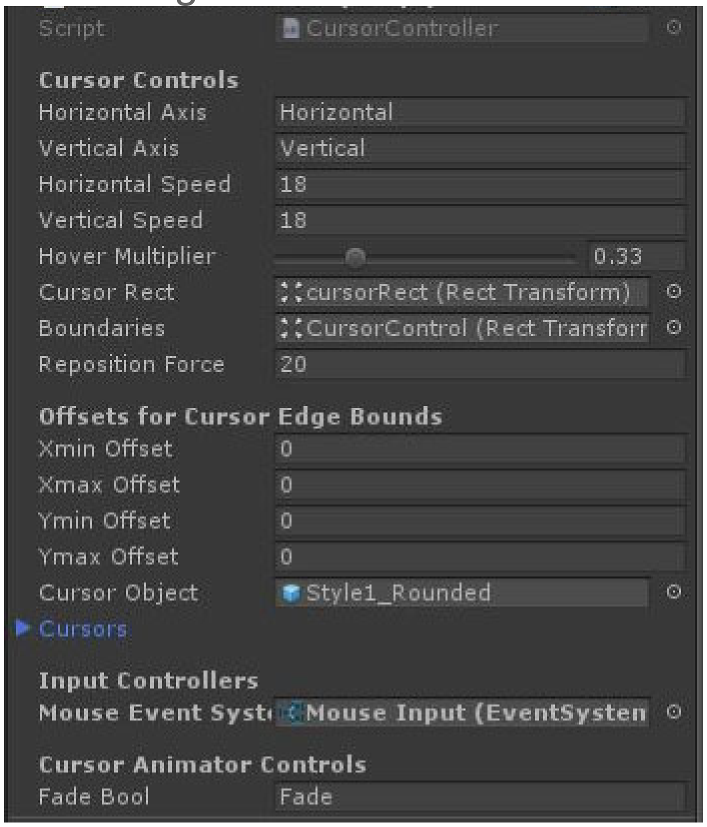
All the cursors have a 1-1 ratio in their scaling, and they all have a parent Rect Transform that can be modified. Simply adjust the scale values on the parent object of each style and it will adjust the scale without affect the animations.


Min X 0.5 Y 0.5
Max X 0.5 Y 0.5
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
Cull Transparent Mes
Animator
Controller Style1
Avatar None (Avatar)
Apply Root Motion
Update Mode Normal
Culling Mode Always Animate
Clip Count: 2
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 2 Muscles: 0

2.1 INSPECTOR WALKTHROUGH

Horizontal Axis – The name of the Axis in the Input Manager being used to control the horizontal movement of the cursor.
Vertical Axis – The name of the Axis in the Input Manager being used to control the vertical movement of the cursor.
Horizontal Speed – The horizontal movement speed of the cursor.
Vertical Speed – The vertical movement speed of the cursor.
Hover Multiplier – The speed multiplier for when the cursor is hovering over a button. 0 = 0% of original movement speed and 1 = 100% of original movement speed.
Cursor Rect – The UI Element with a Rect Transform that will function as the cursor.
Boundaries – The Rect Transform that will be used as the edges of the screen for detection.
Reposition Force – The amount of pixels that the cursor will jump in the opposite direction once attempting to move outside the boundaries of the screen.
Xmin Offset – The amount of pixels from the left side of the screen that the cursor will detect as a boundary.
Xmax Offset – The amount of pixels from the right side of the screen that the cursor will detect as a boundary.

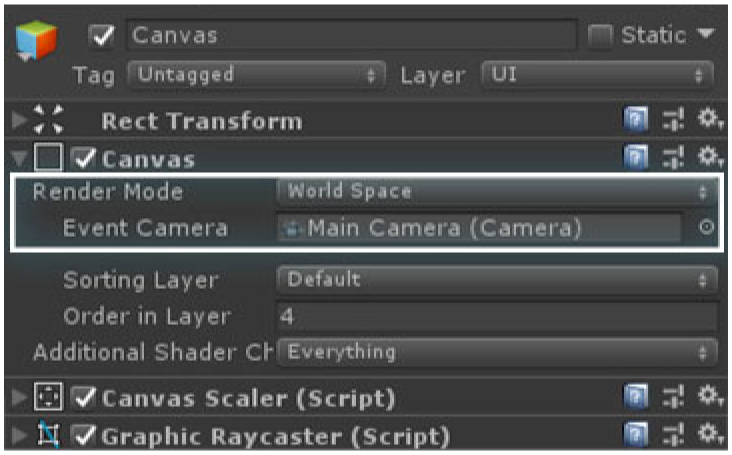
Ymin Offset – The amount of pixels from the bottom of the screen that the cursor will detect as a boundary.
Ymax Offset – The amount of pixels from the top of the screen that the cursor will detect as a boundary.
Cursor Object – The current active cursor style, and the origin point at which the button click detection will be based on.
Cursors – The list of cursor game objects available in a scene to be active.
Mouse Event System – The game object with the Standalone Input Module accessing mouse input.
Fade Bool – The name of the bool variable on the Animator controlling the transitions for the cursors animations.



Script	CursorController
Cursor Controls	
Horizontal Axis	Horizontal
Vertical Axis	Vertical
Horizontal Speed	18
Vertical Speed	18
Hover Multiplier	0.33
Cursor Rect	cursorRect (Rect Transform)
Boundaries	CursorControl (Rect Transforr
Reposition Force	20
Offsets for Cursor Edge Bounds	
Xmin Offset	0
Xmax Offset	0
Ymin Offset	0
Ymax Offset	0
Cursor Object	Style1_Rounded
▶ Cursors	
Input Controllers	
Mouse Event Syst	Mouse Input (EventSysten
Cursor Animator Controls	
Fade Bool	Fade

3.1 CANVAS CONFIGURATION

Make sure the canvas is set to **Screen Space – Camera** or **World Space**. This is the only way that the UI will be able to layer properly. **Order in Layer** can be adjusted to determine what Canvases appear on top. The higher the number, the more on top a canvas will be. Also, you can also just increase the order in layer of the **CursorControl** canvas if your project has specific ordering for canvases already set previously.


Canvas
Static
Tag Untagged
Layer UI
Rect Transform
Canvas
Render Mode World Space
Event Camera Main Camera (Camera)
Sorting Layer Default
Order in Layer 4
Additional Shader Ch Everything
Canvas Scaler (Script)
Graphic Raycaster (Script)

3.2 ADDING BUTTONS TO A SCENE

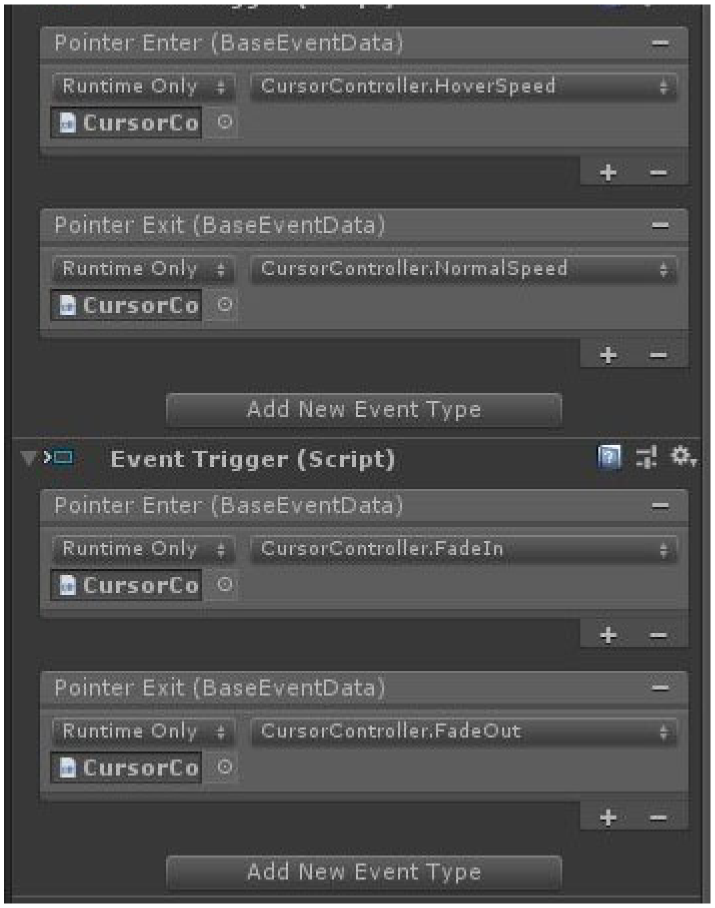
Adding buttons is extremely simple. For demonstration, we will use the button prefabs available with the Console Cursors package. Drag any of the button prefabs into a canvas (any canvas will work).

There are 2 empty **Event Trigger** components on each button. All you need to do to have the buttons properly respond to the cursor movements and inputs is to fill them in.

Fill all 4 object slots with the **CursorControl** object in your scene. Select the functions as the following:

Pointer Enter – CursorController.HoverSpeed
Pointer Exit – CursorController.NormalSpeed
Pointer Enter – CursorController.FadeIn
Pointer Exit – CursorController.FadeOut

And that’s it! The buttons should now be responding. You can copy and paste the Event Trigger components onto any button, including your own, and it will function the same.


Pointer Enter (BaseEventData)
Runtime Only | CursorController.HoverSpeed
CursorCo

Pointer Exit (BaseEventData)
Runtime Only | CursorController.NormalSpeed
CursorCo

Add New Event Type

Event Trigger (Script)
Pointer Enter (BaseEventData)
Runtime Only | CursorController.FadeIn
CursorCo

Pointer Exit (BaseEventData)
Runtime Only | CursorController.FadeOut
CursorCo

Add New Event Type

4.1 BREAKING DOWN FUNCTIONS

All the functions that will need to be called to have the cursor function properly technically and visually are in the list below. All other functions in the script are handled in the background and do not need to be modified.

HoverSpeed() – Calls a function that adjust the speed of the cursor to the multiplied value set on the **Hover Multiplier** slider. The slider has a value range of 0-1 with 0 equaling 0% of the original speed and 1 equaling 100% of the original speed.

NormalSpeed() – Calls a function that returns the speed of the cursor to the starting **Horizontal Speed** and **Vertical Speed**.

FadeIn() – Calls a function that sets the bool value of the string **Fade Bool** to **True**. **Fade Bool** can be modified in the inspector of the **CursorController** script. The script uses the Animator component on the cursorObject.

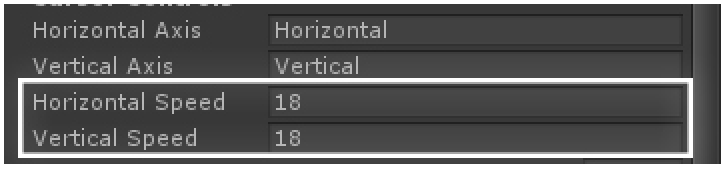
FadeOut() – Calls a function that sets the bool value of the string **Fade Bool** to **False**.
SwitchingToConsoleInput() – Calls a function that switches from mouse input to hand-held controller input. First, it will check to make sure you have a mouseEventSystem in the scene. Make sure to fill the variable **Mouse Event System** in the inspector for **CursorController** or the script will not function properly. The cursor will then become visible through an alpha channel switch (from 0-1). The cursor event system is enabled and the mouse event system is disabled. in Runtime, this will immediately switch the input Controls.
SwitchingToMouseInput() – Calls a function that switches from console hand-held input to mouse input controller. First, it will check to make sure you have a mouseEventSystem in the scene. Make sure to fill the variable **Mouse Event System** in the inspector for **CursorController** or the script will not function properly. The cursor will then become invisible through an alpha channel switch (from 1-0). The cursor event system is disabled and the mouse event system is enabled. in Runtime, this will immediately switch the input Controls.

5.1 CONFIGURING INPUT MANAGER

Console Cursors uses 3 Axes in the Input Manager: **Horizontal**, **Vertical**, and **Submit**. Horizontal and Vertical are modifiable through the inspector on the CursorController script. Submit is the default button for clicking a button.

5.2 CURSOR MOVEMENT

The sensitivity values on **Horizontal** and **Vertical** are set to 1 solely for consistency with keyboard controls. The keyboard input Axes are also named Horizontal and Vertical so Unity will read them both simultaneously. Keyboard input is from 0-1 while a joystick has small incremental values. You can adjust the **Sensitivity** down to increase the smoothness of the movement of the cursor, but make sure to also increase the **Horizontal Speed** and **Vertical Speed**.

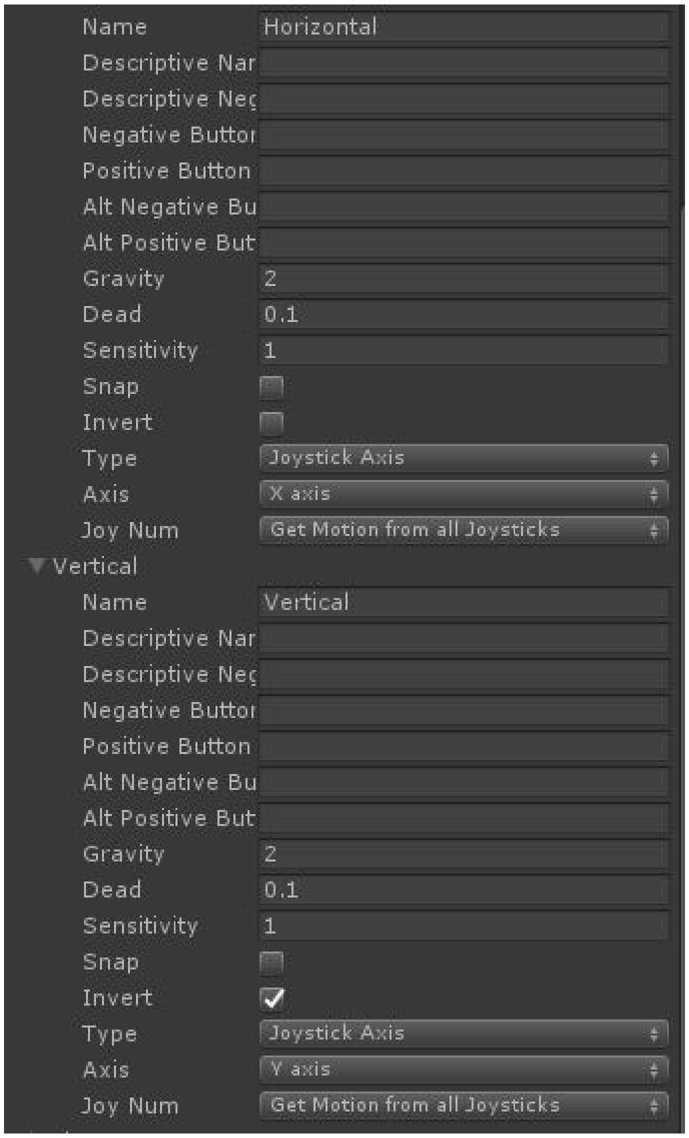


Horizontal Axis	Horizontal
Vertical Axis	Vertical
Horizontal Speed	18
Vertical Speed	18

5.3 ADJUSTING KEYBOARD INPUT

If you choose to control the cursor with keyboard input, the first two Axes in the Input Manager have Positive and Negative button values controlling the movement. Modify those values to change which keys will move the cursor.

You can use this same method to adjust the **Submit** Axis joystick and keyboard button.



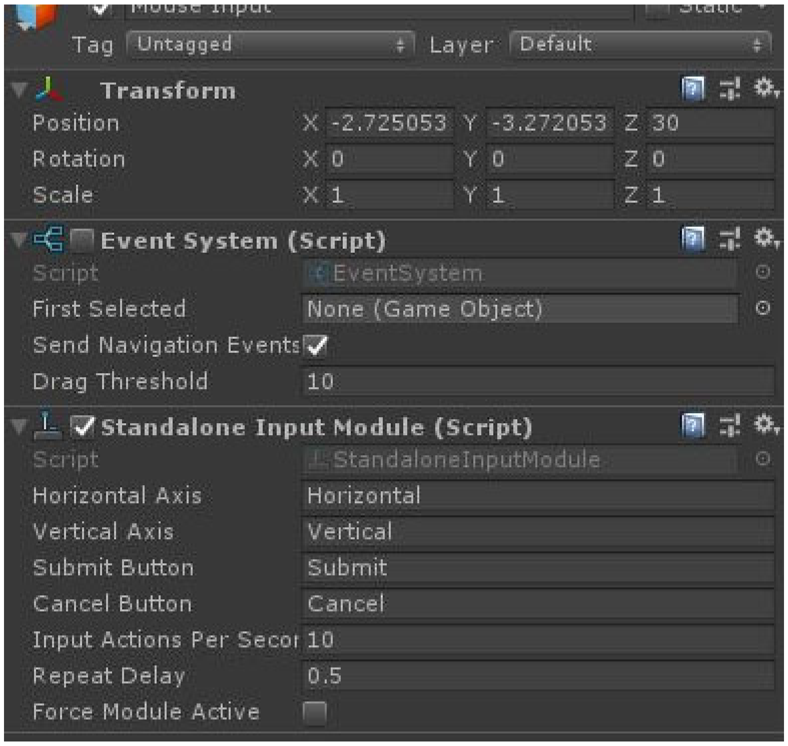
Name	Horizontal
Descriptive Nar	
Descriptive Neg	
Negative Buttor	
Positive Button	
Alt Negative Bu	
Alt Positive But	
Gravity	2
Dead	0.1
Sensitivity	1
Snap	☐
Invert	☐
Type	Joystick Axis
Axis	X axis
Joy Num	Get Motion from all Joysticks
▼ Vertical	
Name	Vertical
Descriptive Nar	
Descriptive Neg	
Negative Buttor	
Positive Button	
Alt Negative Bu	
Alt Positive But	
Gravity	2
Dead	0.1
Sensitivity	1
Snap	☐
Invert	✓
Type	Joystick Axis
Axis	Y axis
Joy Num	Get Motion from all Joysticks

6.1 CONFIGURING MOUSE INPUT

If you want to disable console controls and enable the mouse input, or start with the mouse input instead, simply add an Empty Game object and add the **Standalone Input Module** component. Assign the game object to the **Mouse Event System** variable slot on the **CursorController** script on **CursorControl**. Change the Alpha value on **CanvasGroup** from 1 to 0 to hide the console cursor.
Make sure the **Standalone Input Module** Event System is active and the **CursorControl** Event System is disabled!

6.2 SWITCHING INPUT CONTROLS

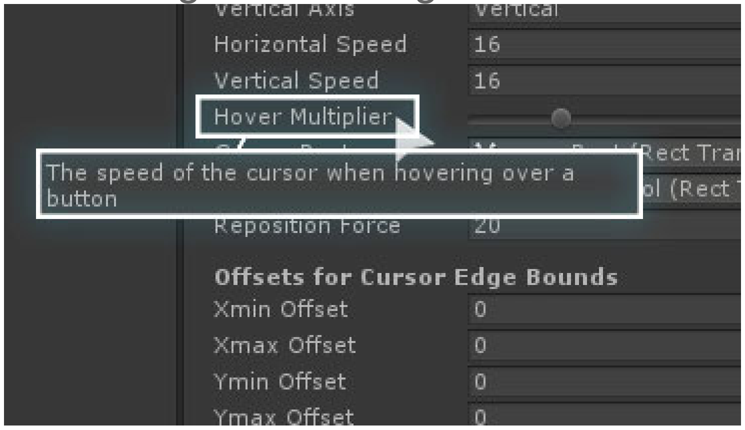
To switch from Hand-Held to Mouse and vice versa have any script call the Switching functions (Section 4.1) corresponding to the desired input controller. In an options menu, if you are pressing a button to change input, call CursorController.SWITCHFUNCTION() and the script will handle the rest.



```
Tag  Untagged                    Layer  Default

Transform
Position                  X -2.725053   Y -3.272053   Z 30
Rotation                  X 0           Y 0           Z 0
Scale                     X 1           Y 1           Z 1

Event System (Script)
Script                    EventSystem
First Selected            None (Game Object)
Send Navigation Events    ✓
Drag Threshold            10

Standalone Input Module (Script)
Script                    StandaloneInputModule
Horizontal Axis           Horizontal
Vertical Axis             Vertical
Submit Button             Submit
Cancel Button             Cancel
Input Actions Per Secor   10
Repeat Delay              0.5
Force Module Active       ☐
```

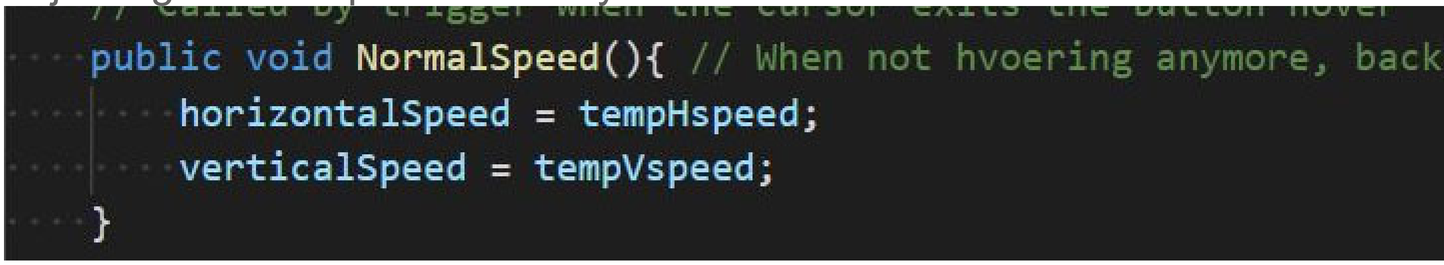
7.1 TIPS

Every variable has a **tooltip** describing it’s function, so you don’t have to guess what each variable is for!
Just hover over the variable to see it’s description in case something is confusing.



```
Horizontal Speed          16
Vertical Speed            16
Hover Multiplier
The speed of the cursor when hovering over a
button
Reposition Force          20

Offsets for Cursor Edge Bounds
Xmin Offset               0
Xmax Offset               0
Ymin Offset               0
Ymax Offset               0
```

For programmers, the code has been documented to ensure adjusting code is quick and easy.



```
    public void NormalSpeed(){ // When not hvoering anymore, back
        horizontalSpeed = tempHspeed;
        verticalSpeed = tempVspeed;
    }
```

NOTE* Each cursor style has its own Animations and Animator, so modifying one will not affect the others.
NOTE* Do NOT rename elements of the Cursor Styles as the Animator uses exact names to reference the key frame animations.
NOTE* Make sure at the start of the scene there is only 1 Event System active in the scene.